MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

Trabajo Fin de Máster

Data-Driven resource orchestration in sliced 5G Networks

Autor: Adrián Gallego Sánchez

Tutor: Isaías Martínez Yelmo y Carmen Guerrero López

2019

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

Máster Universitario en Ingeniería de Telecomunicación

Trabajo Fin de Máster

Data-Driven resource orchestration in sliced 5G Networks

Autor: Adrián Gallego Sánchez

Tutor/es: Isaías Martínez Yelmo y Carmen Guerrero López

TRIBUNAL:

Presidente: Juan Antonio Carral Pelayo

Vocal 1º: Jose Antonio Portilla Figueras

Vocal 2º: Isaías Martínez Yelmo

FECHA: 10 de septiembre de 2019

"Never have so many been manipulated,

so much,

by so few.

- Aldous Huxley,

Complete Essays: 1956-1963"

Abstract

In the past few years the fifth generation in mobile communications started to arise.

5G supposes a great change compared with the past mobile communication generations, it doesn't aim merely at improving bandwidth, reducing delay or upgrading spectral efficiency but at offering a wide range of services and applications, with huge different requirements, to a vast variety of users. These objectives are to be accomplished using new technologies such as: Network Function Virtualization, Software Defined Networks, Network Slicing, Mobile Edge Computing, etc.

The objective of this Master Thesis is to analyze the current support for end-to-end Network Slicing in a 5G Open Source environment and to develop an open source 5G Testbed with recent Software contributions in Network Slicing.

Key Words: *5G, Network Slicing, Virtual Network Function, Network Function Virtualization, Orchestration.*

Resumen

En los últimos años la quinta generación de comunicaciones móviles ha comenzado a desarrollarse.

El 5G supone un gran cambio si se compara con las anteriores generaciones de comunicaciones móviles, puesto que no se centra meramente en aumentar el ancho de banda, reducir la latencia o mejorar la eficiencia espectral, sino en ofrecer un amplio rango de servicios y aplicaciones, con requisitos muy dispares entre sí, a una gran variedad de tipos de usuario. Estos objetivos pretenden ser alcanzados empleando nuevas tecnologías: Network Function Virtualization, Software Defined Networks, Network Slicing, Mobile Edge Computing, etc.

El objetivo de este Trabajo de Fin de Máster es analizar el soporte actual de end-to-end Network Slicing en un entorno 5G Open Source y desarrollar una maqueta 5G con software que admita Network-slicing.

Palabras Clave: *5G, Network Slicing, Virtual Network Function, Network Function Virtualization, Orquestación.*

Table of Contents

List of figures

List of tables

List of Text Boxes

Abbreviations

3GPP	3rd Generation Partnership Project
5G	Fifth Generation (mobile/cellular networks)
5GPPP	5G Infrastructure Public Private Partnership
AADU	Active Antenna Distributed Unit
AI	Artificial Intelligence
AP	Access Point
API	Application Programming Interface
B2B	Business to Business
B2C	Business to Consumer
B2H	Business to Household
B2B2X	Business to Business to Everything
BBF	Broadband Forum
BC	Broadcast
CLI	Console Line Interface
CN	Core Network
CP	Connection Point
C-RAN	Cloud RAN
CSC	Communication Service Consumer
CSP	Communication Service Provider
CU	Centralized Unit
CU-C	CU Control Plane
CU-U	CU User Plane
DCSP	Datacenter Service Provider
DECOR	Dedicated Core Network
DM	Data Model
E2E	End-to-end
eDECOR	Enhanced Dedicated Core Network
eMBB	Enhanced Mobile Broadband
eNB	Enhanced Node B
ENI	Experiential Networked Intelligence
EPC	Enhanced Packet Core
ETSI	European Telecommunications Standards Institute
GST	Generic Slice Template
GUI	Graphical User Interface
HSS	Home Subscriber Server
HW	Hardware
IaaS	Infrastructure as a Service
IESG	Internet Engineering Steering Group
IM	Information Model
IETF	Internet Engineering Task Force

ISC	Intra-Slice Controller
ITU	International Telecommunication Union
KPI	Key Performance Indicator
LCM	Lifecycle Management
LSO	Lifecycle Services Orchestration
MEC	Mobile Edge Computing
MIB	Management Information Base
MIMO	Multiple Input Multiple Output
ML	Machine Learning
MME	Mobility Management Entity
mMTC	Massive Machine Type Communications
mmWave	Millimeter Wave
MTNSI	Mobile-Transport Network Slice Interface
NaaS	Networking as a Service
NBI	North-Bound interface
NEF	Network Exposure Function
NFV	Network Function Virtualization
NGC	New Generation 5G Core
NGMN	Next Generation Mobile Networks Alliance
NMRG	Network Management Research Group
NOP	Network Operator
NR	New Radio
NS	Network Slice
NSA	Non-Stand-Alone
NSaaS	Network Slice as a Service
NSD	Network Service Descriptor
NSE	Network Slicing Engine
NSI	Network Slice Instance
NSS	Network Slice Subnet
NSMF	Network Slice Management Function
NSSMF	Network Slice Subnet Management Function
NST	Network Slice Template
NWMO	Network Management and Orchestration
OAI	Open Air Interface
OASIS	Organization for the Advancement of Structured Information Standards
ONAP	Open Networking Automation Platform
OS	Operating System
OSM	Open Source MANO
PDU	Physical Device Unit
PLMN	Public Land Mobile Network
PRB	Physical Resource Block
PTM	Point to multipoint
PTP	Point to point

QoS	Quality of Service
QoE	Quality of Experience
RAN	Radio Access Network
RCS	Rich Communication Services
RFC	Request for Comment (Document)
RIA	Research and Innovation
SA	Stand-Alone
SC	Service Customer
SDN	Software Defined Networks
SDO	Standard Definition Organization
SG	Study Group
SLA	Service Level Agreement
SME	Small and Medium-Size Enterprise
SMI	Structure of Management Information
SoBI	South-Bound Interface
SP	Service Provider
SPGW	Serving-Packet Data Network Gateway
SPID	Service Profile Identifier
SW	Software
TN	Transport Network
ToR	Top of Rack
TOSCA	Topology and Orchestration Specification for Cloud Applications
TSG	Technical Specification Groups
uRLLC	Ultra-Reliable Low-Latency Communications
UC3M	Universidad Carlos III de Madrid
UE	User Equipment
UML	Unified Modelling Language
V2X	Vehicle to Everything
VDU	Virtual Device Unit
vEPC	Virtual Enhanced Packet Core
VISP	Virtualization Infrastructure Service Provider
VLC-gNB	Visible Light Communication-based gNB
VL	Virtual Link
VLD	Virtual Link Descriptor
VNF	Virtual Network Function
VNFD	Virtual Network Function Descriptor
VPN	Virtual Private Network
VR	Virtual Reality
WG	Working Group
WIM	WAN Infrastructure Manager
XSC	Cross-Slice Controller
YANG	Yet Another Generation (language)

Definitions

This document contains specific terms to identify elements and functions that are mandatory, strongly recommended or optional. These terms have been adopted for use them similarly to that in *IETF RFC2119*, and have the following definitions:

- **MUST**: This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
- **MUST NOT:** This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
- **SHOULD:** This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in specific circumstances to ignore an item, but the full implications must be understood and carefully weighed before choosing a different course.
- **SHOULD NOT:** This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- **MAY:** This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include an option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include an option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides).

1 Introduction

1.1 Extended Summary

Advances in cloud computing, Artificial Intelligence (AI), hardware (HW) and software (SW) technologies have led to a rapid evolution and huge transformation of the telecommunications network, simplifying lots of management processes, operations, provisioning… And, besides, to the possibility of creating and developing new and highly innovative mobile communications technologies, such as 5G.

Although what 5G involves and what it exactly means to be, is still to be defined, it is clear that this new mobile communications generation is going to be characterized by a change in the way cellular connectivity is provided and conceived. Its technology has the aim at offering a huge range of services and applications, the widest in the history of mobile and wireless communications according to [1]. These services can be classified into three main groups: enhanced mobile broadband (eMBB), ultra-reliable and low-latency communications (uRLLC) and massive machine type communications (mMTC). Each of them has different requirements, for example: mMTC are characterized by transmitting low volumes of information, very low delay requirements and low energy consumption; on the other hand, uRLLC, requires high availability and a very low delay (<5ms); eMBB will also require high data rates (+50 Mbps) and extensive coverage (+10 Tbps/Km$_2$). In order to respond to the demand of each service, 5G technology needs to provide a flexible platform throughout the usage of innovative technologies in every layer of its technology: from massive multiple-input and multiple-output (MIMO) and the reduction of the size of the cells (femtocells) located at the Radio Access Network (RAN) of the 5G system; to the usage of multiple virtualization technologies at the Next Generation Core (NGC), such as Software Defined Networks (SDN) and Network Function Virtualization (NFV).

A key feature attached to these requirements and environment, is the so-called concept of Network Slicing (NS) [2], it provides an end-to-end perspective into the 5G framework allowing a flexible architecture and spanning through different domains (core, transport, network; and even through different mobile network operators).

As mentioned above, although it is a promising technology, it is still in development and several Standard Definition Organizations (SDOs) are in charge of looking over the progress and definition of its standards, protocols, architecture, etc. Since the liberation of the third Generation Partnership Project's (3GPP) Non-Stand-Alone (NSA) New Radio (NR), specifications for 5G and the completion, in 2018, of the 3GPP's Rel.15 with an initial set of 5G specifications; other SDOs did take this input and started to develop their own standards: 5G Infrastructure Public Private Partnership (5GPPP), European Telecommunications Standards Institute (ETSI), International Telecommunication Union (ITU), Internet Engineering Task Force (IETF) and others (later on in this document a detailed explanation of each SDO contribution will be given).

All these facts suppose a grand motivation to carry out this Master Thesis, due to the high level of innovation related to them, the huge number of new technologies involved and because of the challenge and advantages that it supposes.

This project aims at filling the lack of practical and experimental validation of truly end-to-end (E2E) NS through the creation of a complete NSA 5G open source sliced testbed with SW such as: OpenStack[1], Open Source MANO (OSM)[2] and Open-Air Interface (OAI)[3].

Finally, it is worth mentioning that this Master Thesis is being developed under the H2020 Research and Innovation (RIA) project 5G Verticals Innovation and Infrastructure (5G-VINNI[4]) and that the results of this work will help and contribute to build a European E2E 5G testbed for future experimentation and validation of 5G Key Performance Indicators (KPIs). 5G-VINNI is part of the 5GPPP phase 3 infrastructure projects (two more projects are inside this research area: 5G-EVE and 5Genesis), it started on July 1st, 2018 and it has a 3 year lifespan. Its main purpose is to provide a set of E2E 5G facility sites which demonstrates 5GPPP network KPIs and that can be accessed and used by vertical industries to launch experiments for innovative use cases, further supporting large-scale pre-commercial 5G technology pilots. Moreover, the infrastructure research carried out by this project will be used as an input for many SDOs to generate new specifications and standards.

1 https://www.openstack.org/
2 https://osm.etsi.org/
3 https://www.openairinterface.org/
4 www.5g-vinni.eu

1.2 Objectives

This section details the main objectives of this Master Thesis and splits them in the following milestones:

a. **Objective 1:** Study of the current standardization process in 5G-NS and identification of the main development and implementation challenges due to differences between standards definitions. This objective is going to achieved carrying out an extensive research throughout several SDOs, 5G architecture and NS definitions; and information and data models.

b. **Objective 2:** Deployment of an open source testbed with recent software contributions in NS. In order to complete this objective various task will be developed:
    a. Investigate, study and select the proper SW for the testbed.
    b. Generate different scenarios that support the deployment of the testbed (or at least part of it). This is expected to end at a production-level scenario.

c. **Objective 3:** Practical Experimentation and validation of the current testbed. At least one NS SHOULD be implemented. To fulfill this objective, multiple tests are going to be developed, from a simple PING-PONG trial to a fully automated virtual Enhanced Packet Core (vEPC).

1.3 Document Structure

Past this introduction chapter, the contents of this Master Thesis are structured as follows:

- **Chapter 2 - State of Art:** Contains an overview of the current state of art of several of the 5G technology Key features. The subsections of this chapter provide detailed information of different related topics E.g. SDOs, standards, architecture, information models (IMs), NS concept. Besides, it also identifies the available Open Source SW and presents it.
- **Chapter 3 - Technical Development:** Contains the technical deployment of the different developed scenarios and details their HW/SW requirements and configuration.
- **Chapter 4 - Test Design, Evaluation and Results:** Focuses on testing the abovementioned scenarios, evaluating them and obtaining results. Several tests are shown.
- **Chapter 5 - Conclusions:** Addresses the conclusions of this Master Thesis.
- **Chapter 6 - Future Work:** Aims at defining future objectives, research and work once this Master Thesis is done.
- **Chapter 7 - Planification and Budget:** Focuses at the planification and costs aspects of this project.
- **Chapter 8 - Specifications:** Covers the specifications and requirements needed to fulfill and carry out this project.

2 State of Art

This chapter offers an overview of several relevant 5G related SDOs, components and SW. It is a summarized view of the 5G ecosystem's current state of evolution and development. Its main objective is to give the reader an overview and introduction to the 5G technology's terminology, architectures and functionalities; as well as trying to depict the most important research areas and giving a slight touch over the future path of the 5G technology.

2.1 Industry Fora and SDOs

This section focuses at giving a brief explanation of the contributions to standards and functionality in areas associated with 5G's architectural development and concepts. A more detailed explanation will be given for the major SDOs and a shorter one to the smallest (see "Others" subsection).

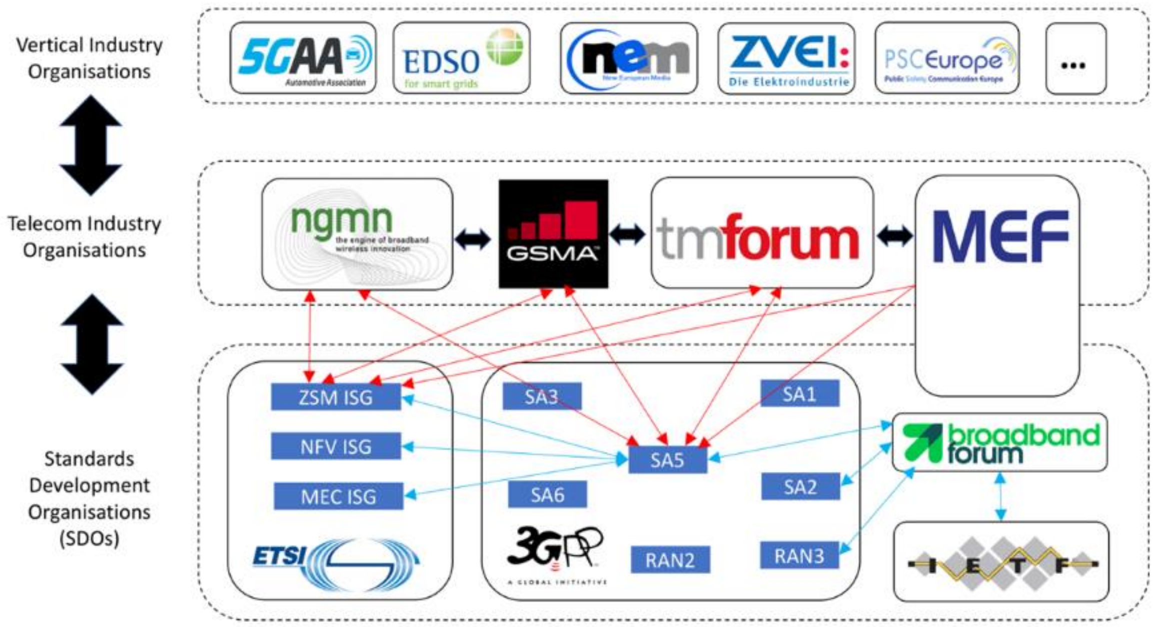

Vertical Industry Organisations
5GAA
EDSO for smart grids
nem
ZVEI: Die Elektroindustrie
PSCEurope
...
Telecom Industry Organisations
ngmn
GSMA
tmforum
MEF
Standards Development Organisations (SDOs)
ZSM ISG
NFV ISG
MEC ISG
ETSI
SA3
SA1
SA5
SA6
SA2
3GPP
RAN2
RAN3
broadband forum
IETF

Figure 2-1. Relevant SDOs for 5G-VINNI project example. [3]

2.1.1 NGMN

The Next Generation Mobile Networks Alliance (NGMN) is a composed industry alliance which supports SDOs through various projects as, for instance, the NGMN Network Management and Orchestration (NWMO) project which specifies the requirements for 5G Network and Service Management and Orchestration [4]. Furthermore, it was also one of the first alliances that provided the basis for some of the 5G technology basics, as it can be appreciated in [2] and [5]. Thanks to these papers Network Slicing concept was brought as a new paradigm for verticals, as technology and service innovation were understood from a business point of view and, last but not least, a 3-layer architecture for NS was provided, which later was taken as a reference by multiple SDOs including: ETSI, IETF, 3GPP and Broadband Forum (BBF).

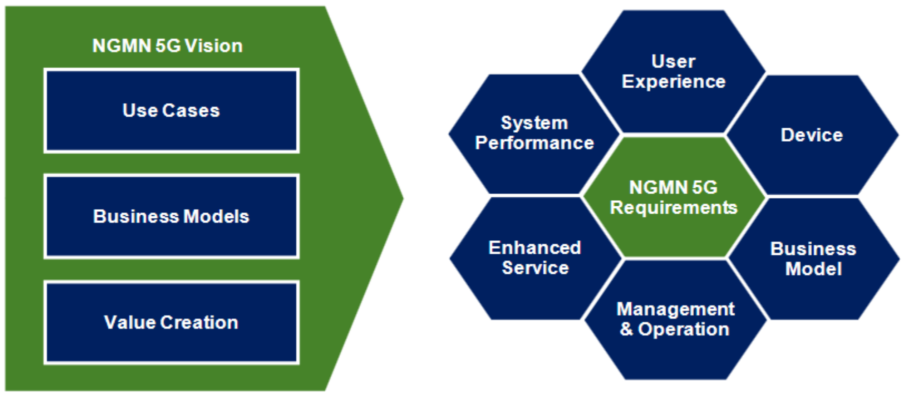


Figure 2-2 - NGMN 5G Requirements. [5]

A second version of the abovementioned 5G architecture framework is envisioned and defined in [6]. Moreover, in this very same paper the following concepts are enlightened:

- Network Slicing concept related to user and business provision
- Orchestration Architecture scenarios. First definition of "Federated Orchestration" (see page 25. [6])
- Identity Management and Quality of Service (QoS)
- Security aspects, business and network related
- Conceptual relationships between several entities. E.g. Tenants and NS

Other whitepapers have been released covering a wide range of customer-facing issues such as "Test&Validation" for pre-commercial network trials and use-cases [7] and secure exposure to 3rd parties [8]. In the last-mentioned whitepaper a series of scenarios, from slice authentication to even the integrated monitoring security functions, are analyzed, from a passive and active exposure, in order to obtain a clear view on the exposure of security capabilities to 3rd parties.

2.1.2 GSMA

GSM Alliance is a trade body that represents the interests of mobile operators worldwide, it is also composed by 400 companies from device makers, to equipment providers and SW developers. This alliance leads the organization of the Mobile World Congress (MWC) and the Mobile 360 series events. It was formally born as the GSM MoU Association with the objective of supporting and promoting mobile operators using GSM.

Currently, GSMA manages three "industry programmes": *Future Networks* (Rich Communication Services - RCS; and 5G), *Identity* and *Internet of Things*. The GSMA has released several whitepapers related to 5G innovative topics, e.g. [9] [10] . On the one hand, whitepaper [9] provides a comprehensive overview about the 5 possible configurations for connecting to an Enhanced Packet Core (EPC) as 3GPP has specified, talks about the benefits of using a Stand-Alone (SA) implementation with a NGC and reviews each deployment option giving slight considerations. Whitepaper [10] focuses on providing detailed guidelines for implementing 5G using 3GPP *Option 3* taking into account technological, spectrum and regulatory considerations throughout the deployment. Additionally, it gives a brief view over Testing Items and Supported Features.

On the other hand, GSMA is highly concerned about the definition of NS and it has presented its own vision of it in [11] (a detailed explanation of this whitepaper and its concept of NS will be giving in the section 2.3. Network Slicing). Furthermore, leaded by the Future Network Programme, the Network Slicing Taskforce (NEST) was born after the publication of the last-mentioned whitepaper; this Taskforce is in charge of detecting and identifying parameter and functionality requirements in NS, NS types and defining a formal and common NS definition.

NEST also studies specific use cases for business and industries. Below, a list with some of these use cases is shown:

- Virtual Reality (VR)
- Automotive – Vehicle to Everything (V2X)
- Disaster Management
- eHealth
- Industry 4.0
- Smart Cities
- Drones
- Aviation
- Public Safety

2.1.3 3GPP

The Third Generation Partnership Project has been, through the past mobile communication generations, one of the main responsible (if not the main) of defining and managing a vast range of standards as can be seen in the Figure 2-3. Core Network Evolution.:

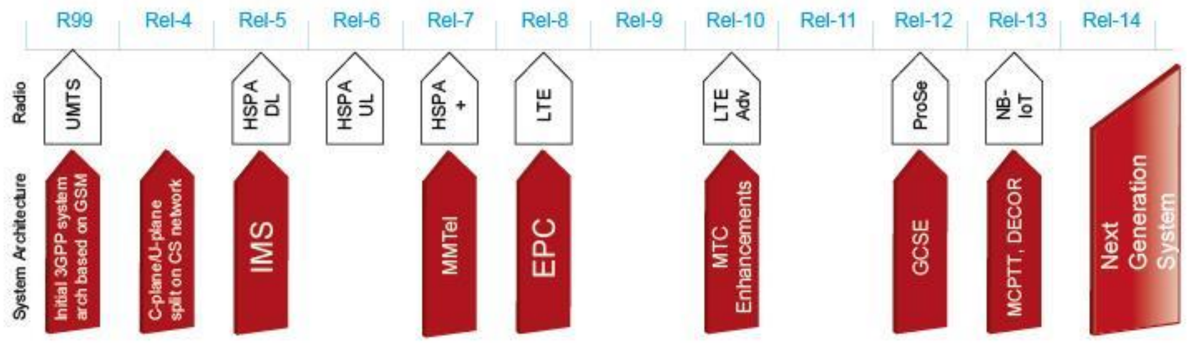


Figure 2-3. Core Network Evolution.

3GPP uses a three-stage approach for the definition of standards where Stage 1, Stage 2 and Stage 3 are focused on service requirements, architecture and detailed interface specification, respectively. Besides, maintains an organized structure based in the so-called *Technical Specification Groups (TSGs):*

- **TSG GERAN** (closed)
- **TSG T** (closed)
- **TSG CN** (closed)
- **TSG RAN** (Radio Access Networks)
- **TSG CT** (Core Network & Terminals)
- **TSG SA** (Service & Systems Aspects)

Each of them is further subdivided in various Working Groups, some insights of the most relevant WG are given in

TSG	WG	Functions
SA	2	Its main responsibility is to develop 3GPP Network Stage 2, takes as an input SA WG1 main functions and network entities, how they are inter-related and how they interchange data. Core, transport and User Equipment – core network layer 3 are also a responsibility of this WG. E.g. This WG wrote down the 3GPP Technical Specification (TS) 23.501 which describes the System Architecture for 5G systems [12] and the TS 23.502 which describes the procedures and operations in 5G systems [13].

SA	5	Contains the MANO aspects and it defines the architecture and requirements for managing and provisioning the whole network and its services. E.g. TS 28.530 defines MANO requirements, concepts and use cases [14].
RAN	1	Responsible for the physical layer of the radio interface.
RAN	2	Oversees the Radio Interface architecture and protocols.
RAN	3	Looks after the overall UTRAN/E-UTRAN protocol and architecture specifications.

Table 2-1. 3GPP TS WG explanation.:

TSG	WG	Functions
SA	2	Its main responsibility is to develop 3GPP Network Stage 2, takes as an input SA WG1 main functions and network entities, how they are inter-related and how they interchange data. Core, transport and User Equipment – core network layer 3 are also a responsibility of this WG. E.g. This WG wrote down the 3GPP Technical Specification (TS) 23.501 which describes the System Architecture for 5G systems [12] and the TS 23.502 which describes the procedures and operations in 5G systems [13].
SA	5	Contains the MANO aspects and it defines the architecture and requirements for managing and provisioning the whole network and its services. E.g. TS 28.530 defines MANO requirements, concepts and use cases [14].
RAN	1	Responsible for the physical layer of the radio interface.
RAN	2	Oversees the Radio Interface architecture and protocols.
RAN	3	Looks after the overall UTRAN/E-UTRAN protocol and architecture specifications.

Table 2-1. 3GPP TS WG explanation.

As it can be observed, 3GPP can be considered one of the most prominent SDOs for Radio and Core-Architecture Standards. Almost all its standards are used by other SDOs to guarantee a certain degree of interoperability.

2.1.4 5GPPP

The Fifth Generation Infrastructure Public Private Partnership (5G PPP) is a composed European initiative between both, the European Commission and the European ICT industry. 5G PPP aims at creating and delivering solutions, architectures and standards for the next generation upcoming technologies and it is willing to reinforce the European Industry to compete in global and innovative markets. They maintain strict KPI requirements for these next generation networks and technologies (see Figure 2-4. 5G PPP KPIs.).

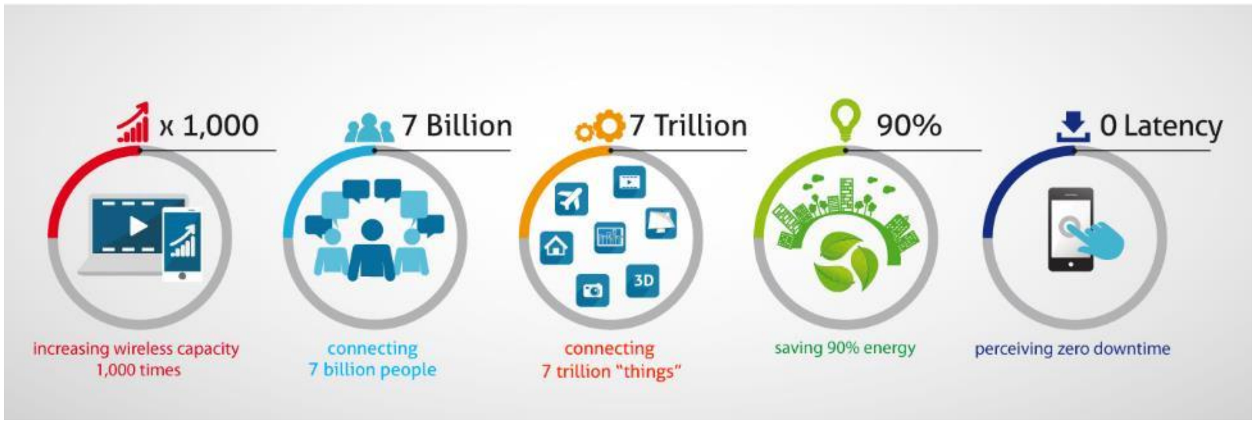

x 1,000
increasing wireless capacity 1,000 times
7 Billion
connecting 7 billion people
7 Trillion
connecting 7 trillion "things"
90%
saving 90% energy
0 Latency
perceiving zero downtime

Figure 2-4. 5G PPP KPIs.

It is structured in two branches: 5G Infrastructure Association (represents the private side) and projects (divided in three phases). On the other hand, it is also composed by WGs that can originate either from 5G IA or from any of the projects, bellow a small introduction to these WGs

Name	Origin	Description
Pre-Standardization	5G IA	Its purpose is to identify SDOs to align with and develop a roadmap of relevant standardization topics relevant for 5G.
Spectrum	5G IA	Promotes research results in the spectrum are obtained by 5G PPP/H2020 projects.
Architecture	5G PPP Projects	It serves as a common platform between 5G PPP projects that develop architecture solutions and builds consensus upon the 5G architecture.
SW Networks	5G PPP Projects	Analyses and addresses key research topics related to SW networking including IoT and Services, networked Clouds, etc.
Security	5G IA	Common platform for security related issues to be addressed by 5G PPP projects.
5G Automotive	5G PPP Projects	Focuses on connected and automated mobility, such as V2X or Vehicle-as-Infrastructure topics.
Test, Measurement and KPIs validation	5G PPP Projects	Looks after automation of tests over E2E NFV infrastructures.

is given:

Table 2-2. 5G PPP WGs.

To end, 5G PPP has released a couple of really interesting whitepapers[5] related to a wide range of 5G topics. [15] [16] [17] are considered the most valuable ones, in fact, [15] will be used as the main input for the following *5G Architecture section* on this document.

2.1.5 ETSI

The European Telecommunications Standards Institute is involved in standardizing conforming components of the 5G system, requirements specifications, multiple use cases and, besides, ETSI also provides feedback to other standardization groups such as 3GPP. E.g. Thanks to the Mission Critical Services Plugtests[6], a platform was created based on 3GPP Mission Critical Services, that allowed to demonstrate the interoperability between many different implementations in various scenarios.

ETSI is structured in: Industry Specification Groups (ISGs), Technical Committees (TCs) and projects. It is the SDO that has taken the lead upon 5G key aspects and technologies, with a high impact on the 5G architecture development, e.g. NFV, MANO and Mobile Edge Computing (MEC). In *Table 2-3. ETSI relevant ISGs, TCs and projects*., a summary of the most relevant ISGs, TCs and projects, for this Master Thesis, is given.

5 https://5g-ppp.eu/white-papers/

6 https://www.etsi.org/events/plugtests

Name	Type	Description
Zero Touch network and Service management (ZSM)[7]	ISG	Focused on the investigation and specification of horizontal and vertical E2E network management architecture to enable full automation of future network and services. [18]
NFV[8]	ISG	Primarily focused on defining an architecture and requirements to enable virtualization of network functions. Papers from this ISG include detailed specifications, proofs of concept and pre-standardization studies. [19] [20]
MEC[9]	ISG	Aims to standardize a multi-vendor MEC platform environment, that allows clear app integration from vendors, service providers and 3rd party-vendors. [21]
Experiential Networked Intelligence (ENI)	ISG	Looks for an improvement in the operator experience defining a Cognitive Network Management Architecture using closed-loop AI.
Open Source MANO (OSM)	Project	Provides an open source MANO SW stack aligned with ETSI NFV IMs.
Satellite Earth Stations and Systems (SES)	TC	Focuses on communication systems, satellite navigation systems and services, earth station equipment…

Table 2-3. ETSI relevant ISGs, TCs and projects.

7 https://www.etsi.org/technologies/zero-touch-network-service-management
8 https://www.etsi.org/technologies/nfv
9 https://www.etsi.org/technologies/multi-access-edge-computing

2.1.6 ITU

The International Telecommunication Union oversees defining global standards and ensuring universal access. Its main activities can be split in three sectors:

- **Standardization** (ITU-T)
- **Radiocommunications** (ITU-R)
- **Development** (ITU-D)

Each sector includes multiple Study Groups (SGs) that generate international standards known as ITU-X Recommendations. Following the scope of this Master Thesis, most interesting SGs include:

Name	Description
ITU-T SG13[10]	Defines requirements for 5G, cloud computing and trusted network infrastructures (IMT-2020).
ITU-T SG15[11]	Takes special consideration towards the changes required to achieve, in telecommunication networks, the advantages of futures networks.

Table 2-4. ITU relevant SGs.

2.1.7 IETF

The Internet Engineering Task Force (IETF) is an open SDO which pools together a large community of network designers, operators, vendors and researchers in the area of Internet architecture and Internet-related technologies. IETF Mission Statement is defined and documented in one of hist best-known Request for Comment documents, *RFC 3935*. WGs are used to perform the technical work, they are organized by topics or areas and managed by Area Directors (Ads), whom are members of the Internet Engineering Steering Group (IESG).

The main areas which IETF focus on 5G-related topics are: NS[12] [13], Mobile Edge Computing (MEC), machine learning (ML) at network level and Low Power IoT Networking (LPWA). The Common Control and Measurement Plane WG looks after the standardization of a common

10 https://www.itu.int/en/ITU-T/studygroups/2013-2016/13/Pages/default.aspx
11 https://www.itu.int/en/ITU-T/studygroups/2017-2020/15/Pages/default.aspx
12 https://tools.ietf.org/id/draft-flinck-slicing-management-00.html
13 https://tools.ietf.org/id/draft-defoy-netslices-3gpp-network-slicing-02.html

control plane for non-packet technologies (microwave links, optical cross-connects, etc.) found in the Internet and in telecom service provider networks.

2.1.8 Others

Apart from the abovementioned SDOs, there are a couple more of them that SHOULD be mentioned because they represent interests of industry groups that MAY have enough weight in the future or in other projects or contexts:

- **Open Networking Foundation (ONF):** Non-profit user driven SDO focused on the adoption of SDN through the development of open standards.
- **Metro Ethernet Forum (MEF):** It is an industry association that recently developed the MEF 3.0 transformational global services framework for defining, delivering and certifying agile, assured, and orchestrated services across a global ecosystem of automated networks. MEF is also developing Lifecycle Services Orchestration (LSO) specifications using Open Application Programming Interfaces (APIs), as said in [22] *"LSO enables service providers to transition from a silo-structured BSS/OSS approach towards flexible end-to-end orchestration that unleashes the value of SDN and NFV. Standardized LSO APIs are critical for enabling agile, assured, and orchestrated services over automated, virtualized, and interconnected networks worldwide."*

2.2 5G Architecture

The main objective of this section is to briefly describe the current state of the 5G Architecture. As this topic could lead to the creation of an entire Master Thesis, due to the enormous amount of related information, the author of this document has decided to focus this section on one of the many papers, standards and specifications that define the 5G Architecture. After a long comparison, reading and investigation, the selected document has been *"5GPPP 5G Architecture White Paper v.3.0."* [15]; mainly because it provides a clear and up-to-date vision of the 5G Architecture and all of its components taking into account the latest 3GPP Rel.15 specifications and, more importantly, the feedback from 5GPPP's Phase 2 and Phase 3 projects for this topic and their current impact in future specifications such as 3GPP Rel.16 (available at the end of 2019).

2.2.1 Overall Architecture

A whole new interaction model between service providers, manufacturers, HW suppliers, Small and Medium-Sized Enterprises (SMEs) arises in the 5G ecosystem. Due to this fact, it is important to clearly define Stakeholder Roles in the 5G environment. 3GPP offers a great description of these roles, but exclusively from an operator point of view. In [23] section 1.3, 3GPP and NGMN stakeholder roles are used to define a more extensive view of the stakeholder roles. In *Table 2-5. Stakeholder roles*. they are summarized:

Name	Description
Virtualization Infrastructure SP (VISP)	Operates and provides several virtualized infrastructures.
Network Operator (NOP)	Uses aggregated virtualized infrastructure from virtualized infrastructure providers to develop network services which are offered to SPs.
Service Provider (SP)	Designs, builds and operates services using aggregated network services. Three sub-roles are contemplated: • **Communication SP:** Traditional Telecommunication Services. • **Digital SP:** IoT or eMMB services. • **NS as a Service (NSaaS)** provider: Offers a NS along with its services.
Service Customer (SC)	User of the services offered by SPs (vertical industries are the main SC).

Data Center SP (DCSP)	It differs from VISPs because they offer "raw" resources in centralized locations, rather than aggregating multiple technology domains and making them accessible through an API.

Table 2-5. Stakeholder roles.

5G is being designed to support a wide range of services, vertical industries needs and to meet the requirements of a society where machine type and human-centric applications will coexist, resulting in a new and complex technology with highly diverse functional and performance requirements.

[15] proposes the following recursive structure:

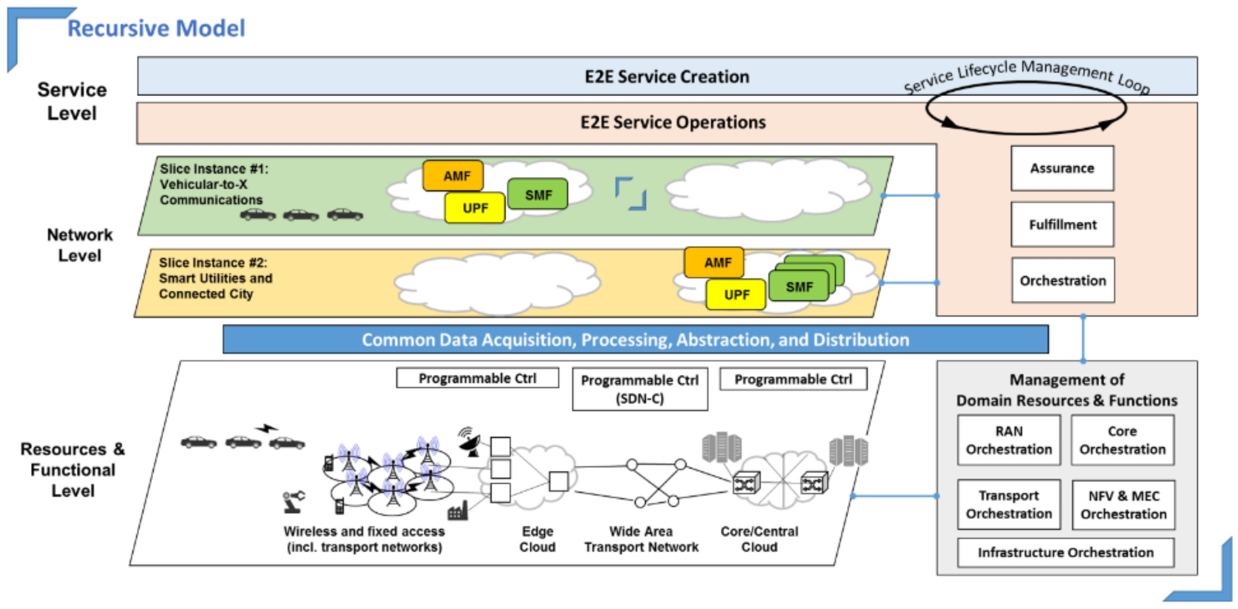

Recursive Model
Service Level
E2E Service Creation
Service Lifecycle Management Loop
E2E Service Operations
Network Level
Slice Instance #1: Vehicular-to-X Communications
AMF
UPF
SMF
Slice Instance #2: Smart Utilities and Connected City
Assurance
Fulfillment
Orchestration
Common Data Acquisition, Processing, Abstraction, and Distribution
Resources & Functional Level
Programmable Ctrl
Programmable Ctrl (SDN-C)
Programmable Ctrl
Wireless and fixed access (incl. transport networks)
Edge Cloud
Wide Area Transport Network
Core/Central Cloud
Management of Domain Resources & Functions
RAN Orchestration
Core Orchestration
Transport Orchestration
NFV & MEC Orchestration
Infrastructure Orchestration

Figure 2-5. 5GPPP Overall Architecture.

As it can be seen in the previous figure, the proposed model shows three architecture levels: Service Level, Network Level (where NS instances are allocated) and Resources & Functional Level (where the physical and de NFV Infrastructure are located). On the first level, Service Level, the E2E Service Creation and Service Operations Framework has a high level of NS lifecycle management (LCM) automation in order to cover the increase of customer (service customers, users, verticals…) service requests. LCM automation SHOULD be achieved by closed-loop Service Assurance, Service Fulfilment and Service Orchestration functions that cover all the lifecycle phases[14]. If a precise spatial and temporal level policy and rule programming is required, SDN controllers at the Resources & Functional Level can be programmed to efficiently carry on these actions.

A recursive architecture, as proposed here, holds the ability to create a service from existing services, even from another instance of the same service; improving scalability and allowing a slice instance to run over the NFVI provided by another NS Instance (NSI) e.g. Each existing tenant can own its own MANO system. In order to be able to support this recursive architecture, a layer of abstraction for the management and controlling of each slice SHOULD be provided by a set of common APIs, allowing each different tenant to request the slice provisioning through them and so, making the underlying infrastructure resources and SW transparent to the tenants.

2.2.2 RAN Architecture

[15] proposes in section 3.1 an overall RAN architecture based on 3GPP *TS 38.300, TS 38.401* and the consensus from 5GPPP Phase 1 projects latest specifications.

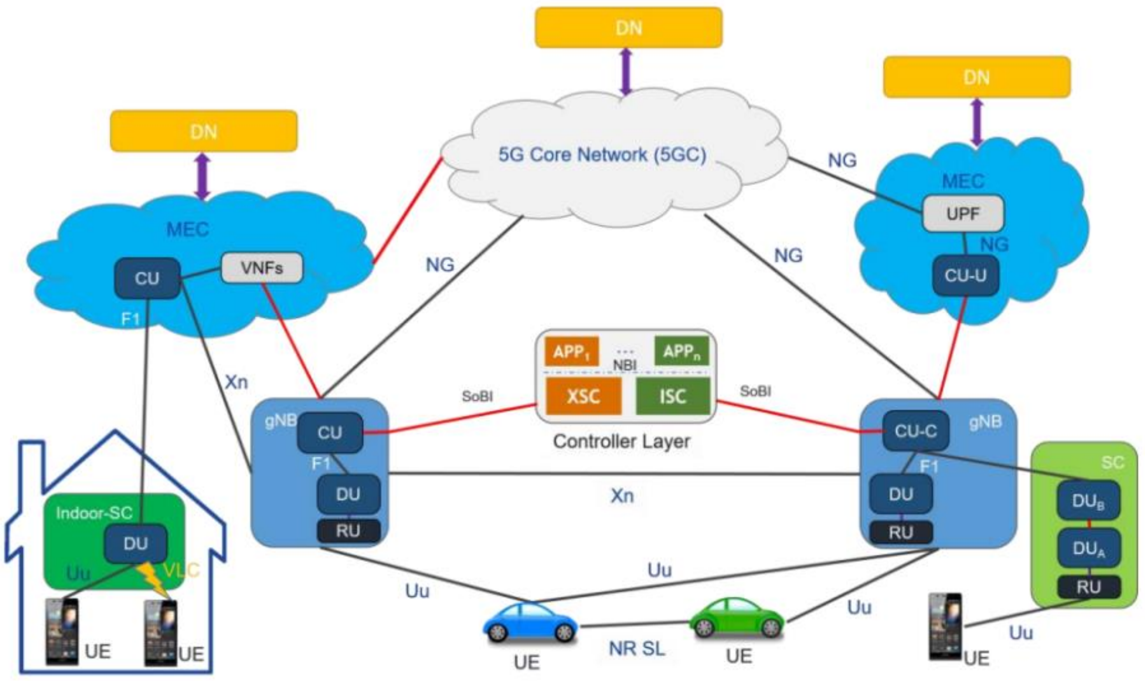


Figure 2-6. 5GPPP RAN Architecture.

14 Preparation, instantiation, configuration, activation and decommissioning.

The most important features of the above RAN Architecture are shown in the following list:

- **Small Cell coverage As a Service** for multiple operators.
- **Two-Tier Architecture:**
    - **First Tier:** Distributed, in charge of low-latency services.
    - **Second Tier:** Centralized, in charge of compute-intensive services and network applications.
- **Increased versatility** through virtualization techniques for:
    - **Data Isolation**
    - **Latency reduction**
    - **Resource usage efficiency**
    - **VNF placement and live migration**
- **Centralized Unit (CU) MAY be split in[15]:**
    - **Control Plane (CU-C)**
    - **User Plane (CU-U)**
- **Controller Layer:** Enables RAN programmability for RAN control functions.
- **Multiple links cooperation:** Applications can run over the North-Bound Interface (NBI) over the cross-Slice Controller (XSC) and Intra-Slice Controller (ISC), maintaining the communication with the RAN through the South-Bound Interface (SoBI).
- **NFV technology usage:** Allows to deploy, with no signaling cost to the 5G NGC, multiple Small Cells as one Small Cell with a VNF deployed at the MEC.

It is important to mention that support for NS in the RAN is currently provided by 3GPP specifications, more concretely in 3GPP *Tdoc RP 182554/Tdoc RP-182850.* Basic NS support is achieved in 3GPP Rel.15, but further and enhanced support (relevant signaling changes, implementation-dependent algorithms…) for an efficient management of the NS shared resources is expected to be achieved in 3GPP Rel.16. NS is one of the main tools in 5G environments, consequently, as 5G pools a wide variety of radio technologies (LTE, 5G NR, Wi-Fi…), virtualization is required to enable it. The simplest way of virtualizing a wireless interface is to share it among a set of tenants e.g. In LTE, a Public Land Mobile Network ID can be instantiated for each tenant on the same carrier, to differentiate between the tenants. An implementation of Wi-Fi virtualization is detailed in section 3.2.1.2 of [15].

Another important functionality that should be mentioned is the introduction of Broadcast and Multicast functionalities in the RAN level. The prime objective of this functionality is to generate a NR mixed mode with multicast capabilities and a new 5G-NR air interface based on the terrestrial broadcast mode provided by the LTE enTV MBMS. Both modes are detailed bellow:

15 Enables the possibility of deploying CU-C and CU-U in different locations

- **NR Mixed Mode:** Facilitates seamless switching between Point-to-Point (PTP) and Point-to-Multipoint (PTM) transmissions. It aims at enhancing the compatibility of the NR air interface for PTP and adding required features for PTM (new DCI format, channel acquisition for user-groups through a common Group Radio Network Temporal Identifier).
- **Terrestrial BC Mode:** Lets the possibility of receiving TV and radio services to users without uplink capabilities, exist, in mobile or SA BC networks. It is oriented to transmit over large areas in High-Power high-Tower networks with a single cell, multi-frequency and single-frequency network configurations.

Moreover, from an implementation and deployment point of view, there are several proposed RAN deployment options. A key aspect of the 5G ecosystem is to maintain the compatibility and integration with the existing mobile communications technologies such as 4G LTE. As mentioned in earlier sections, 3GPP has proposed and defined a set of NSA/SA architectures options; besides, [9] does an extensive analysis of each of the options. Here, in this Master Thesis, only the Options 3, 3a and 3x NSA are going to be mentioned because they are the most probable options to be implemented at 5G-VINNI Spanish facility and, so on, they are the most probable options that the author will have to implement[16] at some point in the project. On the other hand, as later will be explained, Option 3, 3a and 3x will probably remain as a practical deployment need to maintain the coexistence of both, 5G and 4G networks.

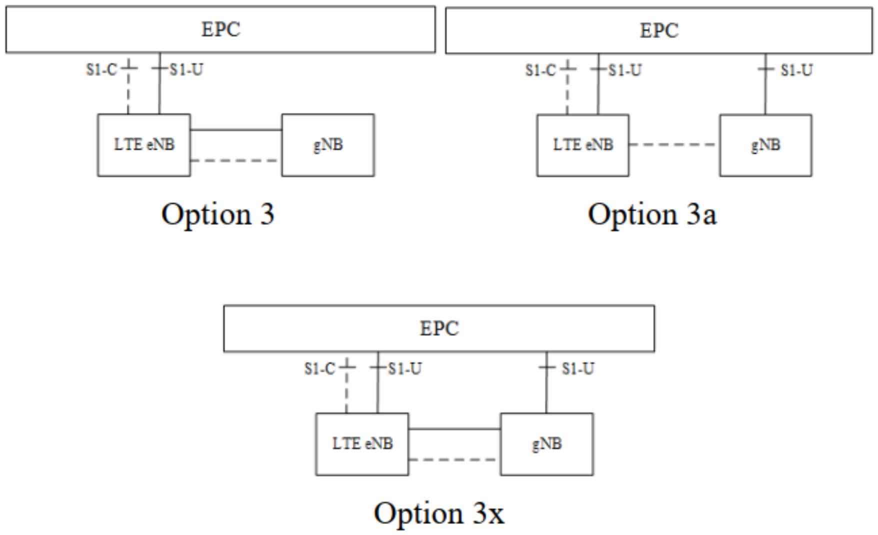


Figure 2-7. 5G-VINNI rel.0 RAN deploying options.

16 Option 3 is the one to be implemented.

As it can be seen, all these options are NSA and, accordingly, all of them reuse the LTE EPC and eNB connecting them through NSA NR. In Option 3 the NR user plane connection is done through the eNB, in the case of Option 3a, this is done directly; finally, in Option 3x the solid between eNB and gNB indicates the transmission of user plane data that terminates at the gNB.

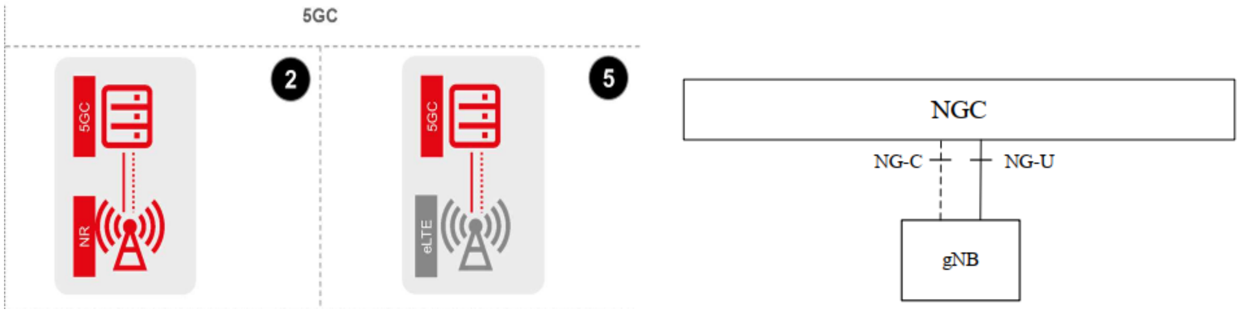


Figure 2-8. Option 2 and Option 5 SA RAN deploying options.

The evolution of these deployment options towards a SA option such as the depicted Option 2 and Option 5, is still being questioned and investigated due to the high dependency of the NSA options over the LTE eNB and EPC. Except that Option 2 and Option 5 coverage exceeds LTE coverage, there will be a need to enable fallback from 5G to 4G. *Table 2-6. Practical factors towards Option 2 deployments.*, exposes the factors to be considered to analyze the feasibility of this evolution:

Condition	Description
4G radio deployed infrastructure	If the operator has already deployed 4G radio, NSA options are easier to introduce and less expensive. Besides, the placement of this 4G infrastructure could be critical for the location of the new 5G radio antennas.
Area Covered	Far edge, aggregated edge, regional and central types are considered.
Frequency Availability and carrier aggregation	The radio coverage of 5G radio frequencies and the combinations with carrier aggregation of several 5G/4G frequencies could lead to robust deployments with high bandwidth scenarios.
MEC new applications	Supports new KPIs and monitoring through Capabilities Exposure Functions.

Table 2-6. Practical factors towards Option 2 deployments.

Furthermore, two more architecture functionalities should be mentioned: *"Visible light communication-based gNB"* and *"Baseband processing in active antenna distributed unit"*.

- **Visible light communication-based gNB (VLC-gNB):** It is mainly composed by two subsystems: RAN subsystem and the networking and service subsystem. VLC-gNB aims at generating an indoor 5G small cell, the first subsystem uses mmWave bands (60GHz unlicensed or 40GHz licensed) and the VLC module to release the radio resources. Thanks to this subsystem VLC-gNB can achieve Gbps data rates and sub-meter location accuracy in indoor environments. On the other hand, the networking and service subsystem uses SDN and VNF technologies and pools it with an Intelligent Home IP Gateway (IHIPGW). In consequence, VLC-gNB can deploy UE's location server with the beforementioned sub-meter accuracy and hold support for add-on services. The main advantages of VLC-gNB are that it can be deployed in a wide variety of indoor environments from a small office to a museum or a supermarket and that it provides better QoS/QoE for UE as [24] demonstrates.

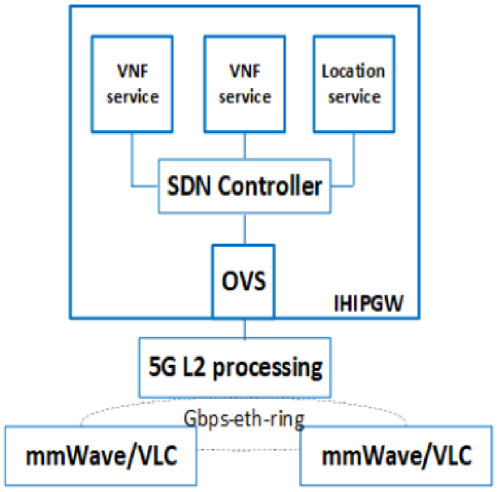


Figure 2-9. 5G RAN VLC-gNB small cell.

- **Baseband processing in active antenna distributed unit (AADU)**[17]**:** It turns to be a completely necessary lower-split implementation due to the extremely high data rates required on the Common Public Radio Interface (CPRI) Fronthaul and the derived infeasibility of CPRI-based C-RAN architectures (hundreds of MIMO antennas will be required). AADU mitigates these facts by including some parts of the Digital Signal Processing (DSP) of the baseband directly into the remotely deployed radio units; the Remote Radio Head and the distributed unit become a single entity (includes analogue and Radio frequency processing).

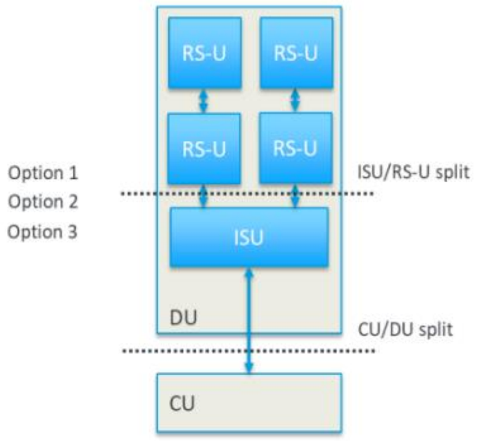


Figure 2-10. AADU overall architecture.

17 See AADU functional split options in section 3.3.3.1 of [15].

2.2.3 Core and Transport Architecture

In the 5G architecture, two clear networks are defined by 3GPP: Core Network (CN) and one or more access networks such as the RAN. CN is composed by NFs, NF services and the interaction between them. On the other hand, a network is required to provide the proper connectivity between the respective Access Points (APs) and the CN, this is the Transport Network (TN). As 5G envisions to provide new services, the TN also needs to adapt to these new requirements. For example, the Cloud-RAN (C-RAN) concept will require connectivity between CUs and Distributes units, overcoming traditional RAN limitations but also introducing the requirement at the transport level of new operational and controlling network services to meet these challenges.

Bellow, in *Table 2-7. CN interfaces and protocols.*, an overview of the 5G's CN most important interfaces is shown, explaining its functionality and transport protocols used.

NAME	FUNCTION	TRANSPORT PROTOCOL
SERVICE BASED INTERFACE	Offering services from a control NF to another control NF.	HTTP
N4	Communication between control and user CN's planes.	IP/UDP/PFCP and IP/UDP/GTP-U
N3 AND N9	User plane data	GTP-U
N2	Control Communications between CN and the current Access Network.	IP/SCTP/NGAP

Table 2-7. CN interfaces and protocols.

As in the RAN Architecture, the CN Architecture is also prepared to support BC and multicast, in this case being aligned with the 3GPP TS 23.501 "*System Architecture for the 5G System*". Several solutions and proposals are given in [1] [12] [15], 3GPP TS 29.500 ,3GPP TS 29.116 and, more concretely, in 3GPP TS 26.246 "*Multimedia Broadcast/multicast Service (MBMS); Protocols and Codecs*". Additionally, a data analytics framework is required in the CN, composed by several data analytics functions (Big Data And MANO, application function level, UE/RAN-Data Analytic Function…) to enable cross-layer optimization. Currently, this is a vast research area due to the huge variables to have into account[18]: data analytics characterization (predicted or expected parameters), granularity of analytics (real-time, on demand, non-real-time…) and the type of the analytics (descriptive, diagnosis, prescriptive…).

[18] See [15] sections 4.2.2.1 and 4.2.2.2 for detailed information.

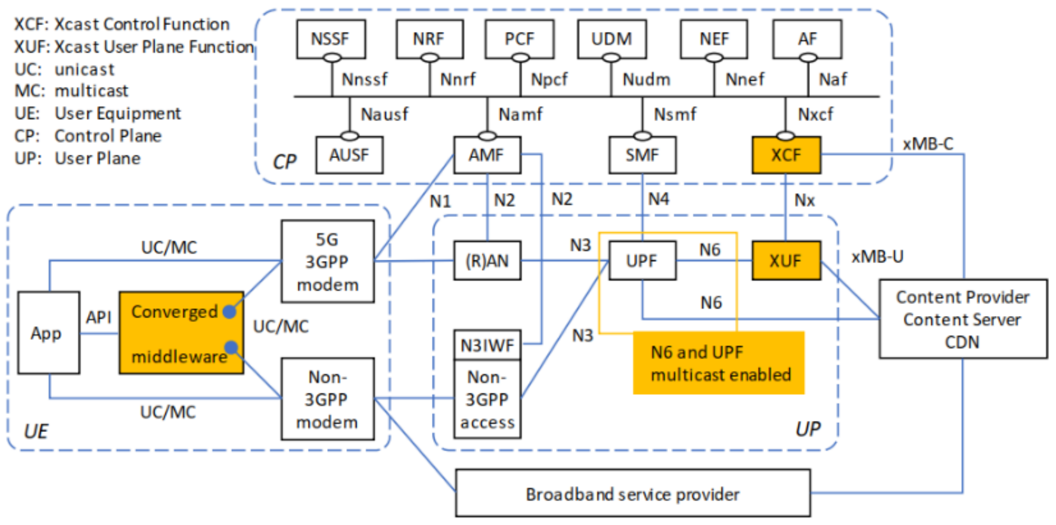


Figure 2-11. Enhanced 5G system Architecture for BC&Multicast.

As it has been mentioned in this section before, the TN faces several challenges, one of the most important ones is to provide proper connectivity between DUs and CUs using commonly digitalized formats (CPRI, eCPR…) or even analogue Fronthaul solutions such as Radio over Fiber (RoF). In summary, new transport technologies SHOULD be used in the TN in order to guarantee high levels of flexibility and resource and energy efficiency. In the following list, some of these technologies are mentioned:

- **Programmable Elastic frame-based Optical Transport**
- **Ethernet**
    - **Eth. Over Multiprotocol Label Switching**
    - **Eth. Over SONET/SDH**
    - **Eth. Over DWDM**
    - **Eth. Over Optical TN**
    - **Flex-E and X-Ethernet**
- **Programmable Metro Network with disaggregated Edge Node**
- **Space Division Multiplexing**
- **Millimeter Wave**
- **Multi-tenant Small Cells with Integrated Access and Backhaul**
- **Satellite Backhaul**
- **Fiber-Wireless (FiWi) PTM**

2.2.4 MANO Architecture

5G's MANO Architecture MUST be analyzed from a high-level perspective, in some cases the SDOs even called it "meta-architecture". This high-level architecture is not a minor research topic in the 5G ecosystem as it has directly related to it, the possibility of reducing the time-to-market of the new features, reduce general costs, the ability of adding greater flexibility and versatility and to leverage an operator's infrastructure for new business models. One of the core issues in this Architecture is the necessity of managing the Life-Cycle processes of the "softwarized" components of the network (VNFs, NFs, NSs…), this issue, has been named as *"Management and Operations"* paradigm. There is an existing reference Architecture addressed for this paradigm, which has been raised by ETSI's WGs and has received some additional inputs from 3GPP and some 5GPPP Phase 2 and Phase 3 projects.

ETSI NFV and ETSI MANO have been the most important SDOs in this Architecture as they have defined the basics of NFV, VNF, NFVI and key interfaces. Besides, ETSI ZSM, ETSI MEC, ETSI ENI and 3GPP SA2 and SA5 WGs have contributed greatly to stablish a common structure and a consensus over the MANO Architecture, better called MANO meta-architecture as it is still not enough precise to be implementable. Some of the key features of the common meta-architecture are mentioned bellow:

- **Controlling individual NFs.**
- **Implementation of SFC** to achieve the conversion of individual NFs into services.
- **Device Heterogeneity:** It has to be capable of using different execution environments, different virtualization techniques (VMs, containers or even just plain processes) in clusters of various sizes (DCs or a single CPU machine) over a wide range of HW types (FPGAs, ASICs, general purpose HW…) in different networking domains or technologies (wireless, wired, optics…).
- **Role Heterogeneity:** Several network operators, companies with different business models and with different "organization domains" are to be present in this architecture. It is a matter of roles; a single company can assume multiple roles or even a single role can be split-out in multiple business or administrative domains.
- **Application Heterogeneity:** In this environment applications will need very different resource, deployment, management and orchestration resources. This feature is sometimes referred to as "application domains".
- **Network Slicing.**

Again, defining core roles is a key for the correct development of this meta-architecture. Based on the abovementioned common features, seven different roles can be expected:

- **End User**
- **Function Developer**
- **Application Developer**
- **Validation/Testing Entity**

- **Tenant** (application owner)
- **Operator** (further differentiation between communications operator, slicing operator, etc. SHOULD be made)
- **Infrastructure provider** (further subdivisions COULD be made network infrastructure provider, cloud infrastructure provider, etc.)

Even with these stablished roles overlapping exists, for example, between function and application developer (both concepts “function” and “application” in the MANO architecture and environment SHOULD be better defined), or between tenant and validation entity. This slight un-definition of these roles leads to scenarios where an operator can act as a tenant and run its own validation formulas. Furthermore, it has be mentioned that the abovementioned roles belong to different phases in the lifecycle of NFs and the NFVI.

In order to bring some feasibility to the MANO (meta)Architecture, most of the proposals and inputs begin considering the ETSI NFV Reference Architectural Framework and its components (see Figure 2-12. ETSI NFV reference architectural framework.).

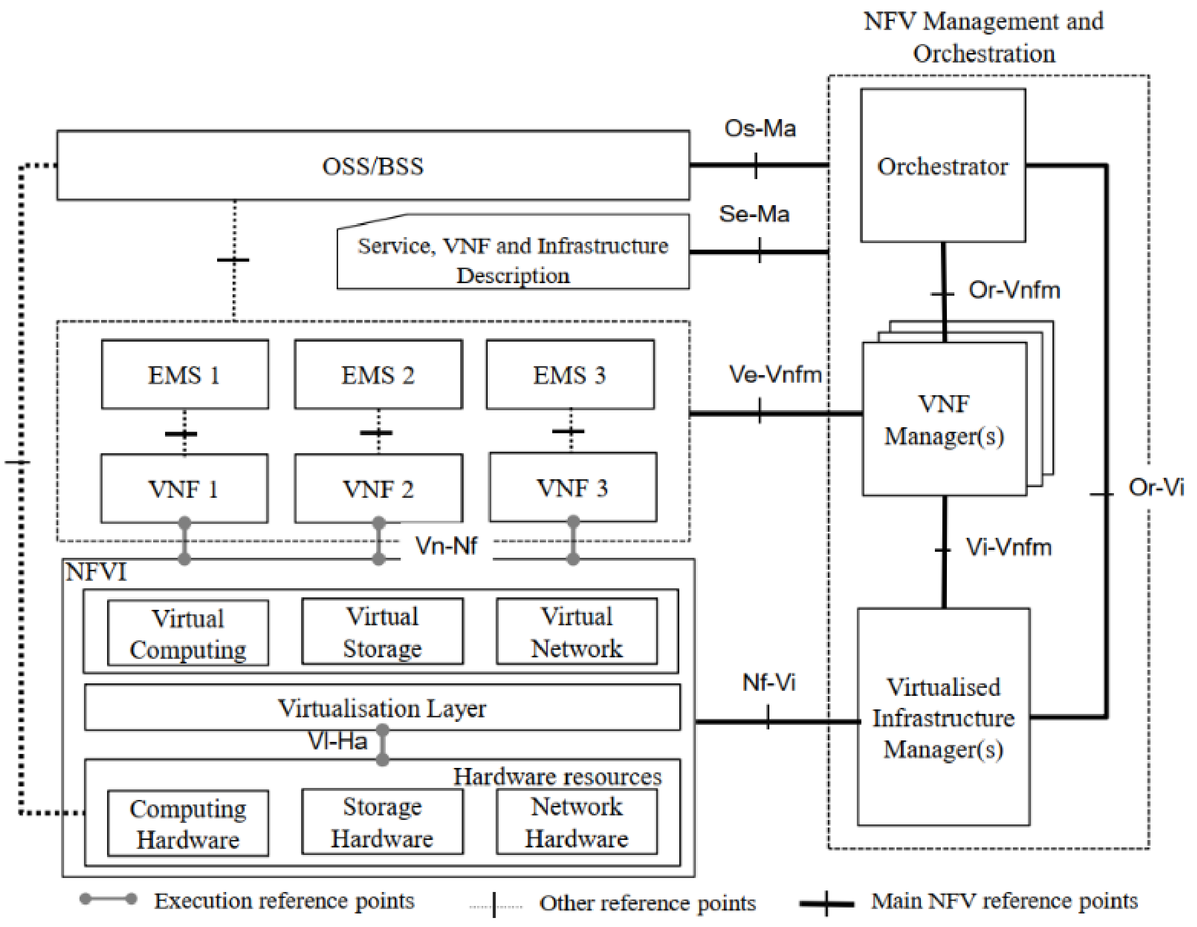


Figure 2-12. ETSI NFV reference architectural framework.

Consequently, any NFV service platform is made-up by four[19] main modules:

- **Orchestrator (NFVO):** In charge of realizing network services allocated in the NFVI and the MANO of the NFVI and SW resources.
- **VNF Manager (VNFM):** Responsible of the VNF LCM. Multiple VNFMs MAY be deployed, one for each VNF or one for various VNFs.
- **Virtualized Infrastructure Manager (VIM):** From a NFVs point of view, it comprises the functionalities which are used to control and stablish the interaction between the final VNF and its associated resources (computing, storage, network…). Multiple VIMs with various hypervisors MAY be deployed.
- **NFVI:** As defined in [19], NFVI is "The totality of all HW and SW components which build up the environment in which VNFs are deployed, managed and executed. NFVI can span across several locations (…)".

Other than this common consensus (meta)architecture, some other options or extensions have been proposed as still has room for interpretation among the real implementations of the system/architecture. Bellow, the most important and remarkable extensions/paradigms, are mentioned[20]:

- **Integrated vs. Segregated Orchestration**
- **Flat vs. Hierarchical Orchestration**
- **Orchestration vs. Slicing**

[19] Operation Support Systems, Business Support Systems (OSS/BSS) and the Element Management System (EMS) are not directly required as they can be derived from the main modules.

[20] See [15] section 5.3 for deeper information.

2.3 Service Modelling

The scope of this section is to present and describe the importance of service modelling towards model-driven service management; and show the differences between Information Models (IMs) and Data Models (DMs). Finally, relevant NS models from several SDOs are summarized.

Basing the services of a given management system in the specific information given in a model is the basis of "model-driven service management". Two kind of models are to be mentioned: IMs and DMs; the main standard reference that specifies the differences between both models is *IETF RFC 3444* [25]. Summarizing section 2 of [25], the following differences can be extracted:

INFORMATION MODEL	**DATA MODEL**
Protocol-agnostic	Protocol-specific
Conceptual model	Detailed model
Hides implementation details	Focuses on implementation details
Gives information about the components of the service	Gives information about mapping the conceptual model to data structures, protocols, etc.
Defines the boundaries of the DM	Takes as an input the respective IM

Table 2-8. Main differences between IMs and DMs.

2.3.1 Information Models

IMs aim at giving conceptual/abstract definitions of the managed components of a service, as well as the relationships between these components and the ways they interact. As described in [25]:

"IMs are primarily useful for designers to describe the managed environment, for operators to understand the modeled objects, and for implementors as a guide to the functionality that must be described and coded in the DMs. The terms "conceptual models" and "abstract models", which are often used in the literature, relate to IMs. IMs can be implemented in different ways and mapped on different protocols. They are protocol neutral."

As mentioned in the above description IMs can be implemented using various description languages from natural languages such as English to formal languages such as the Unified Modelling Language (UML). The first option, natural languages, is used mainly when the scope is to define an informal IM; on the contrary, when a formal IM is scoped, UML's class diagrams are usually utilized. In fact, several SDOs use UML thanks to the language simplicity, visual readability and easy usage. The following table has been adapted from [26] table C-2, Annex C:

ENTITY	DESCRIPTION LANGUAGE	INFORMATION MODEL
3GPP	UML	5G Network Resource Model
IETF	English	RFC 3290
NGNM	English	5G NS Concept
ETSI NFV	UML	NFV IM
ONF	UML	Core IM

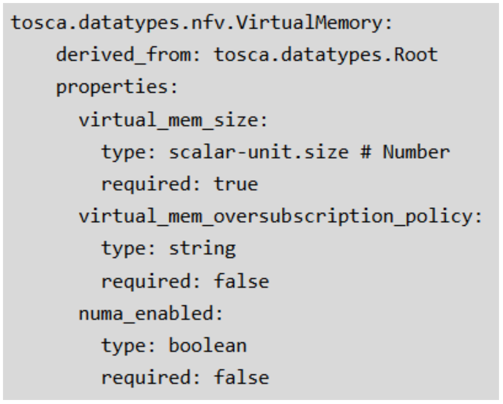
Table 2-9. IM examples.

2.3.2 Data Models

In comparison with IMs, DMs are aimed at a lower level of abstraction, more focused on implementation and protocol-low-level details e.g. Data structure, operations and integrity rules. Similarly to IMs, DMs require a description language and, although there are a wide variety available there are three of them which are vastly used in networking topics:

- **SMIng:** Derived from the Structure of Management Information (SMI) language versions SMIv1 and SMIv2; and defined by the Network Management Research Group (NMRG). It eliminates the semantic and ambiguousness from these languages and arises as a new Management Information Base (MIB) module language; one of its main objectives was to stop the development of different DM languages inside the IETF, harmonizing the existing models.

- **Topology and Orchestration Specification for Cloud applications (TOSCA):** Created by the Organization for the Advancement of Structured Information Standards (OASIS), TOSCA has a special focus on orchestration of cloud-based applications, that is what makes this language specially appropriate for web services rather than for describing network services or functionalities. An adaptation for NFV was developed [27] following the standardized fields of ETSI NFV's IM.



```
tosca.datatypes.nfv.VirtualMemory:
    derived_from: tosca.datatypes.Root
    properties:
      virtual_mem_size:
        type: scalar-unit.size # Number
        required: true
      virtual_mem_oversubscription_policy:
        type: string
        required: false
      numa_enabled:
        type: boolean
        required: false
```

Figure 2-13. TOSCA NFV DM VirtualMemory Datatype.

- **Yet Another Next Generation (YANG):** Developed by the IETF, its main scope is the configuration of running network devices, apps and state information. It supports translation to various schema/tag languages such as YAML, JSON or XML. See IETF RFC 6020[21] for further information.

2.3.3 Network Slicing relevant Models

As demonstrated in the *Section 2.1 Industry Fora and SDOs* of this document, there are several standardization bodies and industry associations developing specifications and standards related to the 5G concept. In this section, these entities are re-examined in order to identify which of them has a heavy role in the development of IMs/DMs directly related with 5G NS concept. Below a table summarizing the scope of these entities towards this topic is shown:

ENTITY	MODEL	RELEVANCE
NGNM	IM	Considered the first industry association to generate a logical view over NS. NGNM proposed a 3-layer NS IM which is widely used by SDOs as an input for further DMs.
GSMA	IM	Looks over business cases and vertical use cases for NS. GSMA has defined the IM for a Generic Slice Template (GST).
TFM	IM/DM	Focus its efforts on simplifying the OSS/BSS system. Lately, it has been developing an Open Digital Architecture with a highly detailed IM and Open APIs.
3GPP	IM	Aims at standardizing the definition of NS, NS instance (NSI) and NS Subnet (NSS). It has developed several IMs.
ETSI	IM/DM	Focused on defining the NFV framework and its interfaces. Besides, ETSI proposes a mapping between its IMs and NGNM NS IM.

Table 2-10. NS IMs/DMs developers.

[21] https://tools.ietf.org/html/rfc6020

2.4 Network Slicing

This section focuses on creating a self-definition for the NS concept, pooling together various SDOs definitions and adapting them to the objective of this project and the 5G-VINNI scope (generating an E2E NS). Additionally, the basic requirements, interfaces and use cases will also be exposed.

2.4.1 Definition

Network slicing is a completely new concept/technology that aims at fulfilling 5G environmental KPIs and at supporting communication services from different verticals and tenants in a cost-effective and efficient manner. So, what is Network Slicing?

"Network Slicing is a technology that splits the E2E common/shared underlying network infrastructure, physical or virtual, into mutually isolated, optimized (NFs, topology, resources, management…) with independent control and management, logical networks that can be created on-demand to satisfy sets of given requirements and a negotiated service quality."

NS is a potential key feature in 5G ecosystems, because it is going to be the technology that enables the possibility of developing a 5G mobile communications system capable of providing all the features and functionalities mentioned in the above definition. Its concept is built around seven principles, as mentioned in [28]:

1. **Automation:** Avoids the need of manual intervention, enabling on-demand dynamic configuration of NS.
2. **Isolation:** Assures security and performance over services used by different tenants, even if they use NS with conflicting performance requirements.
3. **Customization:** Related to *Isolation*, assures that any tenant service requirements are completely fulfilled in an efficient way. This is done at several levels:
   I. **Network Level** (Topology, data plane, control plane…)
   II. **Data Plane Level** (Dedicated NFs, data forwarding and SFC)
   III. **Control Plane Level** (Programmable policies, protocols and data analytics)
   IV. **Context Awareness**
4. **Elasticity:** Key feature that assures the Service Level Agreement (SLA) even when resources and network conditions vary.
5. **Programmability:** Allows the control of the allocated resources, for a NS, to be taken by 3rd parties.
6. **E2E:** Basic and implicit property of the NS technology that guarantees the service delivery from the service provider to the end-customer. It is important to remark that this property MAY imply crossing different business and administrative domains; and MAY imply crossing various technologies and network types.

7. **Hierarchical Abstraction:** Resources linked to a tenant on a particular NS, can be shared among other parties which relates to the NS tenant, generating another NS Service above the prior one. In *Figure 2-14. NGNM NS concept proposal*. this property is clearly depicted on NSI3.

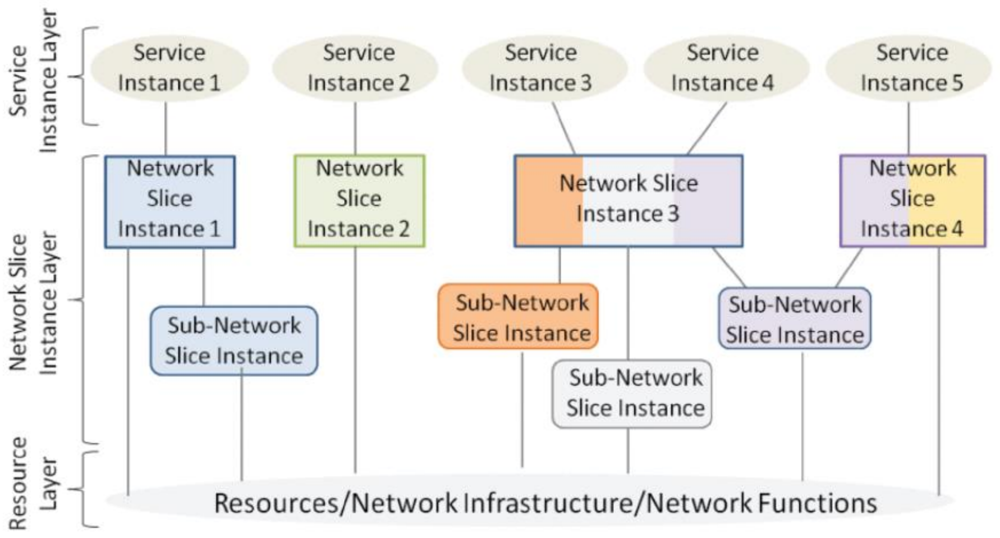


Figure 2-14. NGNM NS concept proposal.

When a NS is deployed in the current infrastructure it is called Network Slice Instance. A NSI has the following characteristics:

- It is the result of instantiating a NS over a certain infrastructure so that multiple NSIs can be instantiated from a single NS.
- Same NSIs are all executed in parallel over a single, shared E2E network infrastructure, but they are isolated from each other (performance, management and security).

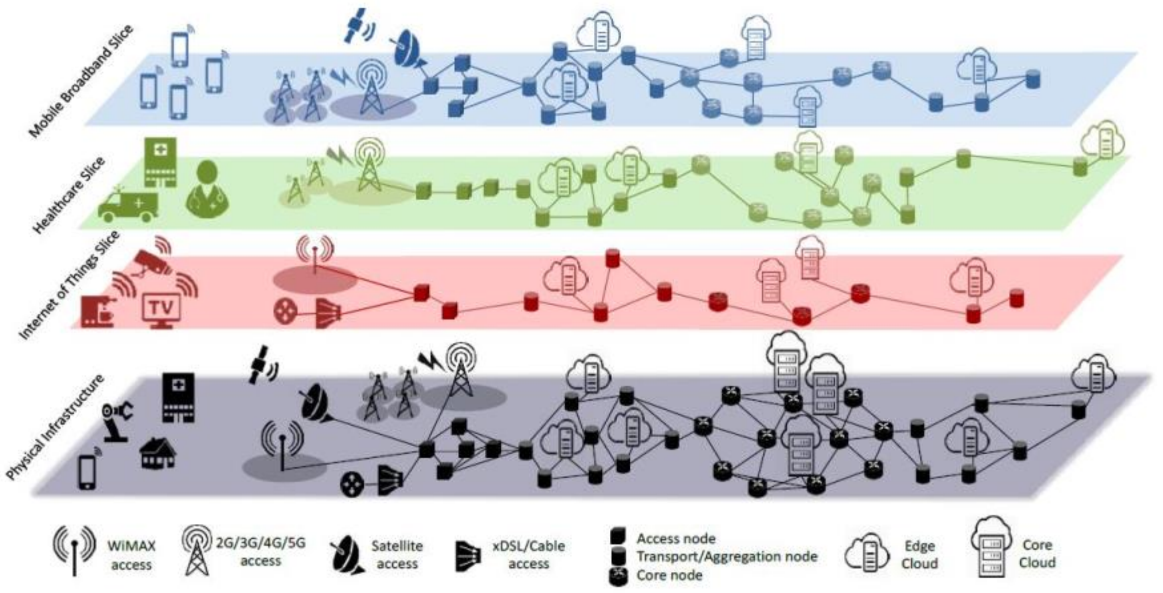


Figure 2-15. NSS/NSI concepts. [29]

Figure 2-15. NSS/NSI concepts. is highly illustrative as it makes a clear picture of the NS concept and how it can manage different services, with a wide variety of requirements, over a shared network. Three NSs are shown on top of a common infrastructure. Each NS: IoT, MBB and uRLLC (healthcare); has one NSI, which is the represented instantiation over the physical infrastructure.

2.4.2 Network Slicing Roadmap

5G E2E NS is an on-going technology, this means that it is still in a developing phase, far away from a full-defined and implemented state. [30] presents an interesting roadmap which clearly pictures the evolution of the NS technology, it identifies four phases:

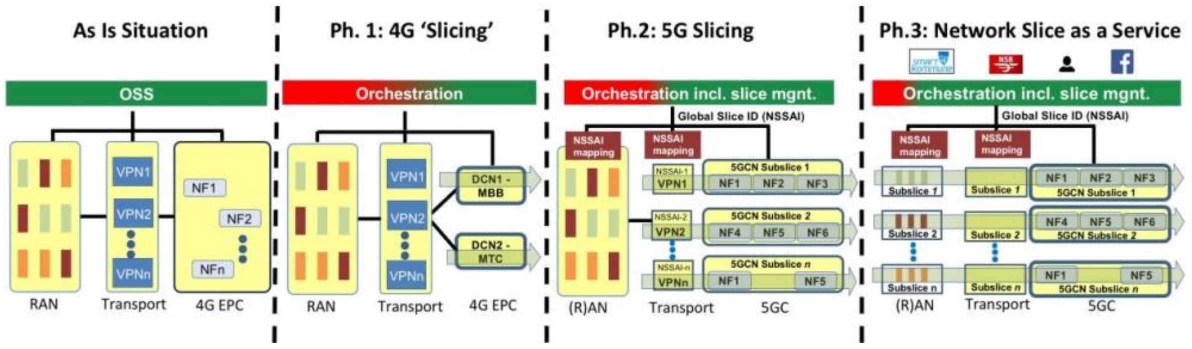


Figure 2-16. NS roadmap.

Each phase represents different states in the evolution of the technology, from the current phase *"As in situation"* where there is no NS and isolation is provided by VPNs and a single CN is present; to the final phase *"NSaaS"*, where 5G NS has been upgraded to a new concept know as Network Slice as a Service[22].

It is important to remark that this Master Thesis aims at achieving some proof-of-concept in between *Phase 1: 4G Slicing* and *Phase 2: 5G Slicing*, mainly because of the inexistence of an Open Source 5G Core.

2.4.3 Network Slice Lifecycle Management

Once again, there are several points of view from which this topic can be addressed, in this section the author is going to be focused on describing NS LCM from a network management perspective. Even though this has been specified before, there are several models to adhere to; for instance, if an innovative and current model is searched, [31] has developed a proof-of-concept model that looks towards automating LCM.

The figure on the right shows the before-mentioned model and the phases that it comprises.

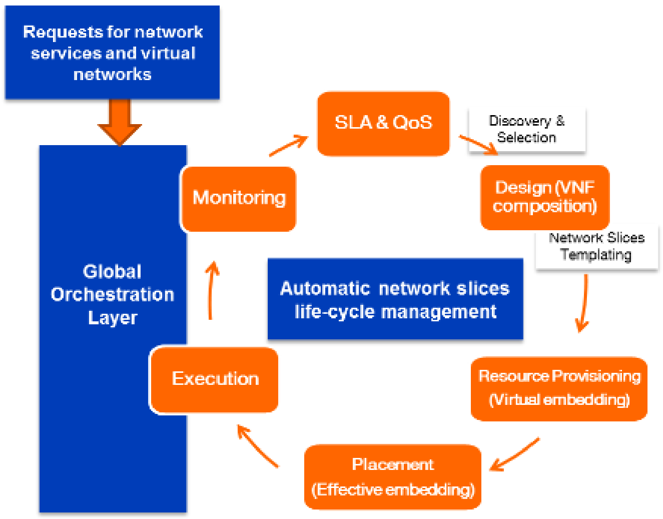

Requests for network services and virtual networks
Global Orchestration Layer
SLA & QoS
Discovery & Selection
Monitoring
Design (VNF composition)
Network Slices Templating
Automatic network slices life-cycle management
Execution
Resource Provisioning (Virtual embedding)
Placement (Effective embedding)

Figure 2-17. Automated and innovative NS LCM model.

Nonetheless, this section is centered on a more standardized and referenced model, this is the 3GPP NSI Lyfecycle[23].

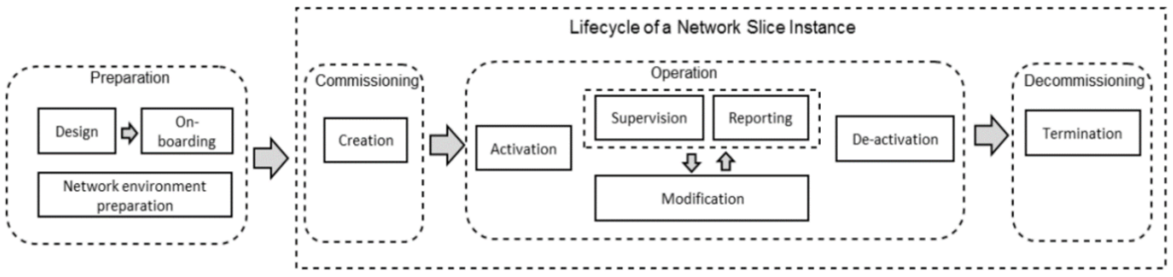


Figure 2-18. 3GPP NSI LifeCycle.

22 In Section *2.4.5 Network Slice as a Service* this concept is explained.
23 See 3GPP TS 28.530.

According to this figure, the lifecycle of an NSI is composed by four phases:

- **Preparation:** Focused on the creation of the NSI, in this phase the NSI does not exist yet. It is composed by the design of the NS Template, resource-planning and requirements' evaluation.
- **Commissioning:** In this phase the NSI's creation takes place, this means that the needed resources and the environment are prepared and allocated to hold the NSI. Various types of NSIs can be created from this phase: completely isolated NSI, NSI with shared NSSIs, etc.
- **Operation:** This phase includes several sub-phases; it is the most complex one. In summary, it involves the activation, modification and de-activation of the NSI.
    - **Activation:** NSI is ready to support communication services.
    - **Supervision & Reporting:** Generates modifications policies because of the monitoring over the resources usage. This sub-phase MAY include Data Analytics functionalities.
    - **Modification:** Refers to changes of the NSI constituents, for instance, when new NS requirements are received through *"Supervision & Reporting"* operations.
    - **De-activation:** Stops communication services execution over the NSI.
- **Decommissioning:** NSI is terminated, obliterated and after this phase it does not exist anymore.

2.4.4 Network Slicing Requirements

As mentioned in section *"2.4.1 Definition"*, 5G NS aims at creating isolated E2E logical networks over a common/shared infrastructure with specific requirements defined by one or more SLAs. To achieve these objectives, several general characteristics are needed inside the 5G ecosystem. This section is going to be focused on the other part, those requirements related with the architecture components and that have a direct impact on the network behavior and KPI fulfillment.

2.4.4.1 Radio Access Network

Till the date, requirements for this "part" of the 5G Architecture are still being explored and evolving. Actually, there is support for NS for LTE/NR NSA implementations based on two slice identifiers:

- **Public Land Mobile Network** (PLMN)
- **Service Profile Identifier** (SPID)

The Radio Resource NF oversees the configuration of sets of predefined shared radio resources (slices), these partitions are distinguished by the PLMN for NOPs sharing the RAN and by the SPID for specific groups of UEs.

On the other hand, NS for 5G Core & NR SA based RAN architectures will allow. through the gNB, to handle differentiated groups of users (each NS MAY contain users with different QoS/QoE requirements); and RAN resources MAY be shared between slices or they MAY NOT. Bellow, key components for NS support on RAN NR-SA architectures are mentioned:

- **Slice-aware CN instance selection:** In charge of choosing the proper CN instance for a specific UE. This functionality can be done using several options:
    - **PLMN ID:** As in LTE traditional network sharing.
    - **Dedicated Core Network (DECOR):** CN selection is carried out using HSS information (UE Usage Type), different SPIDs are used for different RAN Policies.
    - **Enhanced DECOR (eDECOR):** 3GPP Rel.14 UEs are required for this option. In this case, CN selection is done based on UE DCN-ID.
    - **5G Slicing:** Allows connectivity to various slices simultaneously and the slice selection (CN instance selection) is done based on the UE parameter "*Single Network Slice Selection Assistance Information*" (S-NSSAI) which is the NS identificatory in this option. In this parameter, further information is carried as, for instance, the Slice Service Type (SST-1=eMBB, SST-2=URLLC, SST-3=mIoT).
- **Slice-aware performance monitoring:** Related to SLA management; SLA observability will be required at PLMN and S-NSSAI levels to achieve RAN E2E SLAs' adjustments. SLA reporting will be carried out in parallel to SLA observability operations by measuring defined specific slice KPIs between Packet Data Convergence Protocol's reference points.
- **Slice-aware resource management:** Takes care of handling resource isolation/pooling for different groups of users.
- **Slice LCM**

It must be remembered that the 5G Architecture is comprised by one CN and multiple Access Networks; these Access Networks can be RAN or Wireline Access Networks. For further information on this topic see 3GPP TR.716 [32] and for a detailed view on the impact of NS over 5G RAN see [33].

2.4.4.2 5G Core

Once again, there are no clearly defined specifications for NS support over the 5G Core component in the 5G Architecture. Bellow, a table with some guidelines and features has been collected in order to achieve a multi-domain and multi-tenant E2E CN split instance (slice):

Table 2-11. 5G Core NS implementation guidelines.

FEATURE	DESCRIPTION
VIRTUALIZATION	Key feature in order to support the deployment of NSs dynamically. Depending on the service requirements a trade-off between VNFs and Physical NFs (PNFs) SHOULD be considered.
CONFIGURABILITY	As the range of use cases for the 5G network is going to be huge, the 5G CN components SHOULD be highly configurable not just in parameter configuration, but in supporting subscribers with different QoS/QoE requirements.
PROGRAMMABILITY	The 5G CN needs to be able to adapt quickly to changes, allowing fast reactions when network events are triggered. This includes the implementation of component-specific load and state transfer procedures.
SCALABILITY	Any 5G CN NS SHOULD be able to scale and adapt its resources depending on the subscribers varying requirements and on the predicted network usage. Especially relevant for mIoT slices due to the high number of devices to be managed and controlled.
INFRASTRUCTURE AGNOSTIC	5G CN SHOULD be capable of deploying NS scenario using virtual resources, regardless of the underlying HW resources and without a major penalty in performance or usage.
ISOLATION	Guarantees the separation of NSIs that belong to different NSs, as well as information (privacy) isolation. The owner of a slice SHOULD view its slice as a complete dedicated and independent network.
SLICE STICHING	Key feature regarding the process of enabling inter-slice communication and slice-merging.
MANO	Slice management 5G CN components must expose the proper northbound interfaces in order to let MANO solutions to access, manage and influence the slice deployment and configurations.
MONITORING	Allows the automation of the 5G CN, it requires from the CN components to advertise runtime metrics or certain states. Key for NS Management Automation through Machine Learning (ML) techniques.
NF EXPOSURE	Carried out by the 5G CN Network Exposure Function (NEF), allows certain functionalities of the NS to be available to external applications or 3rd party users.
AUTHENTICATION & AUTHORIZATION	Every slice SHOULD implement certain procedures or processes that allow limited access to a group of resources depending on "who" the user is identified as or the role it has.
RELIABILITY	The NS given service SHOULD remain reliable towards subscribers regardless of the infrastructure conditions.

AUTOMATION	Automation operations could be driven over several CN components and processes: fault, performance and security operations; and LCM or low-level policy changes.

2.4.4.3 Transport Network

This part of the 5G Architecture is out of the 3GPP scope, but it represents a crucial component in a NS, especially on a E2E NS, as it is in charge of providing the backhaul and fronthaul infrastructure of the 5G Architecture. The 3GPP management system can identify requirements on different domains (RAN, CN and non-3GPP slice parts – TN) and derive them to the current management system thus, it must be coordinated with the non-3GPP parts management systems in order to prepare a NS.

The BBF plays a key role on the development of the TN NS as it is focused on an interface between the 3GPP NS Management Function and TN Slice Management. It also splits the Network Slice Subnet Management Function (NSSMF) in three entities:

- **Access Network Slice Management:** Looks over the access NSSI's LCM.
- **Core Network Slice Management:** Looks over the core NSSI's LCM.
- **Transport Network Slice Management:** Looks over the transport NSSI's LCM and provides capability exposure of the TN to the NSMF through the Mobile-Transport Network Slice Interface (MTNSI), which is also in charge of passing slice request received from the NSMF and of indicating parameters such as latency, loss ratio, bit rates, etc.

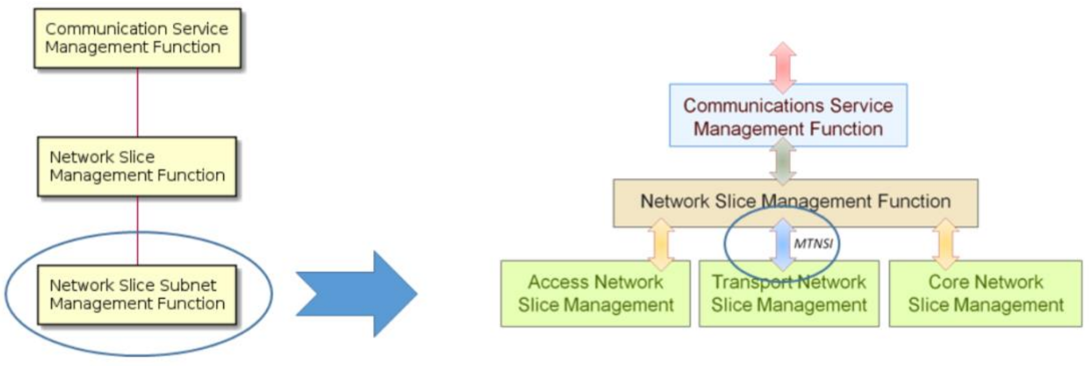

Communication Service Management Function
Network Slice Management Function
Network Slice Subnet Management Function
Communications Service Management Function
Network Slice Management Function
MTNSI
Access Network Slice Management
Transport Network Slice Management
Core Network Slice Management

Figure 2-19. BBF focus on NSMFs.

2.4.4.4 Others

Although it is not directly related to the objectives of this Master Thesis, it is important to mention that NS is also done in other levels of the 5G Architecture, such as:

- **Satellite Transport (Fronthaul/backhaul)**
- **Data Network**
- **User Equipment**

2.4.5 Network Slice as a Service

NSaaS is an innovative concept that consists on, rather than using NS as a technology to support other communication services over the 5G network, providing it directly to customers with its own service description and SLAs. This concept is especially useful for Communications Service Providers (CSPs), four relations can be stablished between CSPs and slice consumers talking about NSaaS:

1. **Business to Business (B2B):** CSP provides the slice to another business-level entity.
2. **Business to Consumer (B2C):** CSP provides the slice to individual users.
3. **Business to Household (B2H):** CPS provides the slice to a group of individual users that have common characteristics.
4. **Business to Business to Everything (B2B2X):** Similar to B2B but the second business-level entity uses the given slice to give a service to its customers.

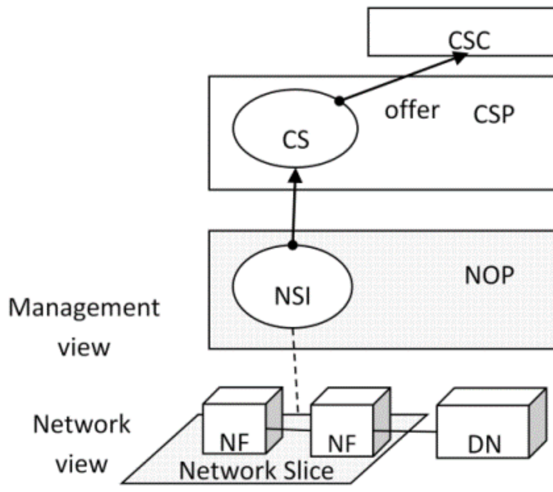


Figure 2-20. NSaaS B2B, B2C and B2H schema.

The above figure, extracted from 3GPP TS 28.530, shows how a CSP can offer a service to a Communication Service Consumer (CSC) which, in this case, could be a Business (B2B), an individual (B2C) or a group of consumers (B2H); as a NS.

2.5 5GPPP Phase 2 Projects

This section aims at mentioning the most relevant 5GPPP Phase 2 projects[24] in terms of standardization, specification and concepts contribution. Furthermore, it aims at giving the reader a general idea of the relevance and importance of these projects and their continuity over the development of the 5G technology in Europe. In *Table 2-12. 5GPPP Phase 2 Projects' contributions*. (extracted from [15]), a complete list of the total number of contributions, by these projects, to standards, white papers, proof of concepts, etc. Per 5G Architectural area is shown:

AREA	NUMBER OF CONTRIBUTIONS
OVERALL ARCHITECTURE	70
RADIO AND EDGE	41
CORE AND TRANSPORT	50
MANO	58
TOTAL	**219**

Table 2-12. 5GPPP Phase 2 Projects' contributions.

2.5.1 5GinFIRE

Started on January 2017 (36-month lifespan), it is a European Commission H2020 project which pursuits building an Open, Extensible 5G NFV-based ecosystem aligned with the on-going standardization bodies. Moreover, 5GinFIRE[25] intends to provide an experimental testbed provisioned with different toolsets and functionality, for industry verticals. Two main outputs from 5GinFIRE should be retrieved:

- **New sets of roles:**
  - **Virtual Vertical/Network Function Developer:** Responsible of uploading VNFDs and NSDs to the 5GinFIRE Catalogue. Verticals take this role when they bring to the 5GinFIRE environment their own VNFD/NSDs.
  - **Experimenter:** Those users that instantiate NSDs as an experiment.
  - **Testbed Provider:** Facility/infrastructure operators.
  - **Services Administrator:** Maintains 5GinFIRE services.
- **5GinFIRE Portal[26]:** Gives access to 5GinFIRE testbed resources for verticals and experimenters.

[24] Full list of 5GPPP Phase 2 projects: https://5g-ppp.eu/5g-ppp-phase-2-projects/
[25] https://5ginfire.eu/
[26] https://portal.5ginfire.eu/#!/

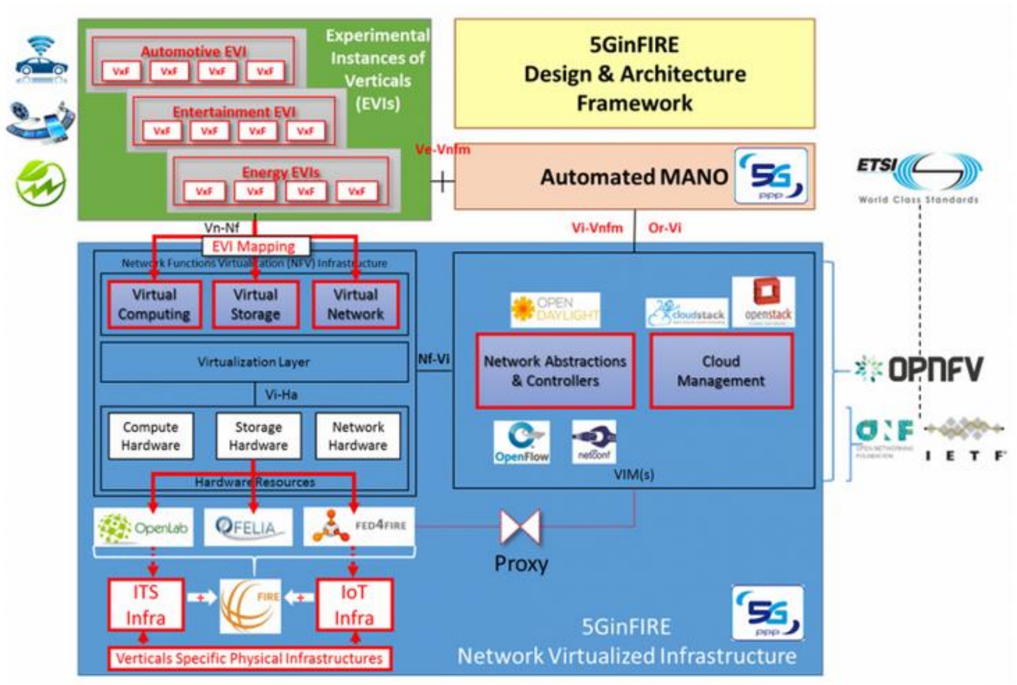


Figure 2-21. 5GinFIRE reference model architecture.

2.5.2 5G-TRANSFORMER

Begun on June 2017 (30-month lifespan), it is a European Commission H2020 project which is especially relevant in relation with the concept of NS. 5G-TRANSFORMER[27] is focused on transforming nowadays concept of a mobile TN into an enriched service platform, to achieve this objective it defines three novel blocks inside its architecture:

1. **Vertical Slicer:** Logical entry point for verticals, it is aware of the business requirements and SLA requirements. It includes a complete catalogue of vertical service Blueprints and maps customer requirements into resource-facing requirements.
2. **Service Orchestrator:** Manages and orchestrates the federation of computing and transport resources into slices. It takes the Vertical Slicer requirements and details and use them to create instances of network services.
3. **Mobile transport and computing platform:** Manages the virtual resources used by VNF/PNF instances over the infrastructure.

27 http://5g-transformer.eu/

5G-TRANSFORMER system architecture is aligned with the 3GPP and ETSI NFV concepts NS concepts, it has contributed with the following features to the NS concept:

a. **Definition of an IM for NS description**
b. **Definition of an IM for vertical service description with NS support**
c. **NS operation and proof of concept across federated administrative domains**

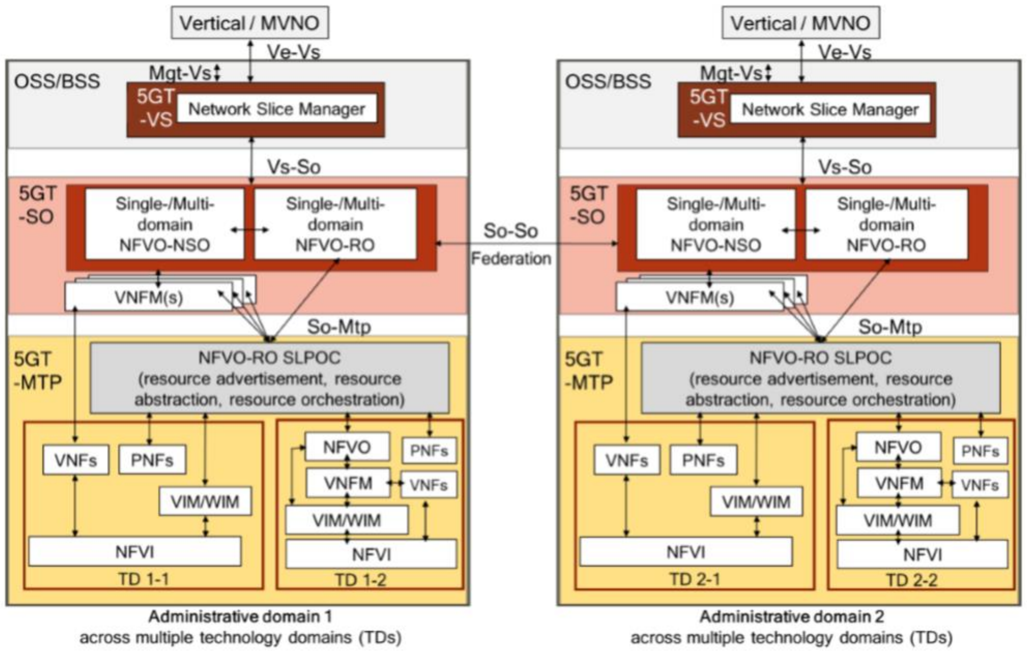

Vertical / MVNO
Ve-Vs
OSS/BSS
Mgt-Vs
5GT -VS
Network Slice Manager
Vs-So
5GT -SO
Single-/Multi-domain NFVO-NSO
Single-/Multi-domain NFVO-RO
So-So Federation
VNFM(s)
So-Mtp
5GT -MTP
NFVO-RO SLPOC (resource advertisement, resource abstraction, resource orchestration)
VNFs
PNFs
VIM/WIM
NFVI
NFVO
VNFM
TD 1-1
TD 1-2
TD 2-1
TD 2-2
Administrative domain 1 across multiple technology domains (TDs)
Administrative domain 2 across multiple technology domains (TDs)

Figure 2-22. 5G-TRANSFORMER system architecture.

2.5.3 5G-TANGO

Started on June 2017 (30-month lifespan), it is a European H2020 project that took as an input some of the concepts and architectural concepts built by 5GPPP Phase 1 SONATA[28] project to develop a multi-modal NFV DevOps approach between several actors, facilitating the implementation of new services and optimizing the design and run-time NFV stages. 5G-TANGO[29] system architecture, as shown in ***Error! Reference source not found.*** is composed by t hree subsystems:

1. **Service Development Kit**: Set of tools that helps the service developer to validate, verify and test NFV services in an efficient and rapid manner.
2. **Verification and Validation Platform**: Sandbox environment for testing activities over designed and defined VNFs.
3. **Service Platform**: Operational environment for the deployment, management, and orchestration of VNFs and Network Services.

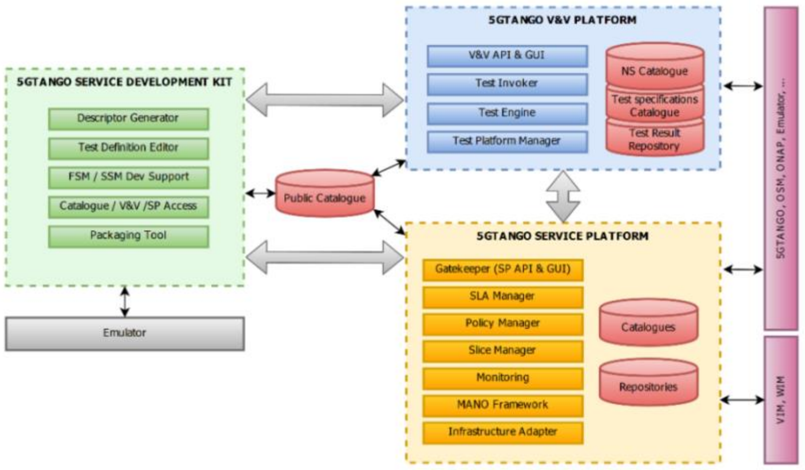


Figure 2-23. 5G-TANGO system architecture.

In this project, as in 5G-TRANSFORMER, NS is a major feature and, consequently, it has introduced the following capabilities for NS support:

a. **Preliminary Network Slice Template (NST) model definition**
b. **NSM architectural definition**
c. **NSM logic implementation**

In fact, the Network Slice Manager (NSM) defined by 5G-TANGO has been integrated as part of the OSM Release FIVE being ETSI compliant.

28 http://www.sonata-nfv.eu/
29 https://5gtango.eu/

2.5.4 SLICENET

Started on June 2017 (36-month lifespan), it is a European Commission H2020 project focused on developing MANO for multi-layered systems in SDN/NFV ecosystems using cognitive techniques and AI. One of its main objectives is to integrate NS technology and provide NSaaS for vertical customers; to achieve these objectives, the SLICENET[30] system architecture is divided into three planes: Management plane, control plane and data plane. Besides, the Management plane is further divided in three layers: Service Management Layer, Slice Management Layer and Resource Management Layer. Finally, these planes and layers contain the three main features that allow a complete NSaaS-AI ecosystem:

1. **One-Stop APIs:** Single entry-point that enables the system reachability for verticals, it pools the resource-facing services into selectable features that can be chosen when the creation of a new NS is required.
2. **Model-Driven Service Orchestration (NSaaS):** Equivalent to the Vertical Slicer feature of the 5G-TRANSFORMER 5GPPP project, the only difference is that, in this case, direct interaction with the verticals is not allowed (it is done through the One-Stop APIs)
3. **Plug&Play Control:** Each NSI has an instance of this feature, it exposes the NSI capabilities to the corresponding vertical and enables NSaaS, as it allows the verticals to have a high degree of control and operation over the given NSI.

2.5.5 SaT5G

Started on June 2017 (30-months lifespan), it is a European Commission H2020 project that pursuits, as described in the 5GPPP-SaT5G[31] web: *"(…) to develop a cost effective "plug and play" satellite communication solution for 5G to enable telcos and network vendors to accelerate 5G deployment in all geographies and at the same time create new and growing market opportunities for satcom industry stakeholders."*

In order to achieve its goals, SaT5G[32] has to develop an ETSI-compliant orchestration solution that is able to operate over 5G core functions, Satellite TN and to perform LCM over SatCom VNFs. SaT5G supports also E2E NS through its orchestrator, the main characteristics of its slices are:

- Dynamic and flexible E2E multi-domain SLA-based slices deployed over satellite resources.
- Dynamic provisioning and instantiation.

30 https://slicenet.eu/
31 https://5g-ppp.eu/sat5g/
32 https://www.sat5g-project.eu/

- Multi-domain management and orchestration operations over virtualized satellite and terrestrial functions.

For detailed information of this solution see “TALENT: Terrestrial Satellite Resource Orchestrator” [34].

2.6 Open Source Platforms

The scope of this section is to provide an overview of the current and more developed Open Source SW around the 5G topic. Three sub-sections are exposed, one for each of the main components needed to build-up a 5G NSA scenario.

2.6.1 Cloud

This first sub-section is centered on cloud-based Open Source solutions. Just one SW platform is described, OpenStack, as it is the only platform that fulfilled all the requirements for this project.

2.6.1.1 OpenStack

OpenStack[33] is a free Open Source SW platform for cloud-computing that allows to control sets of compute, storage and networking resources. It is composed by several modules that enable the SW to give an Infrastructure as a Service (IaaS) functionality through a set of RESTful APIs and command-line tools. Besides this IaaS features, it provides a dashboard that allows administrators to have graphical control over their infrastructure and empowers users to provision resources through a web interface; extra-modules are available to provide bonus services such as: orchestration, fault and service management, monitoring, LCM, billing, etc.

It was a joint project originated from NASA and Rackspace Hosting launched in July 2010, by 2016 the OpenStack Foundation oversaw the project management and since them, more than 500 companies have joined the project.

As OpenStack was born with a different idea than the normal clouds which are operated and designed by a single organization (AWS, AZURE…), this led to some specific considerations when it was generated:

- Interoperability
- Bidirectional Compatibility
- High scalability
- Built-in reliability and durability
- Customizable integration
- HW virtualization

33 https://www.openstack.org/

- Plays well with other PaaS, serverless compute platforms, Container Orchestration engines…
- Allows basic physical Data Center Management

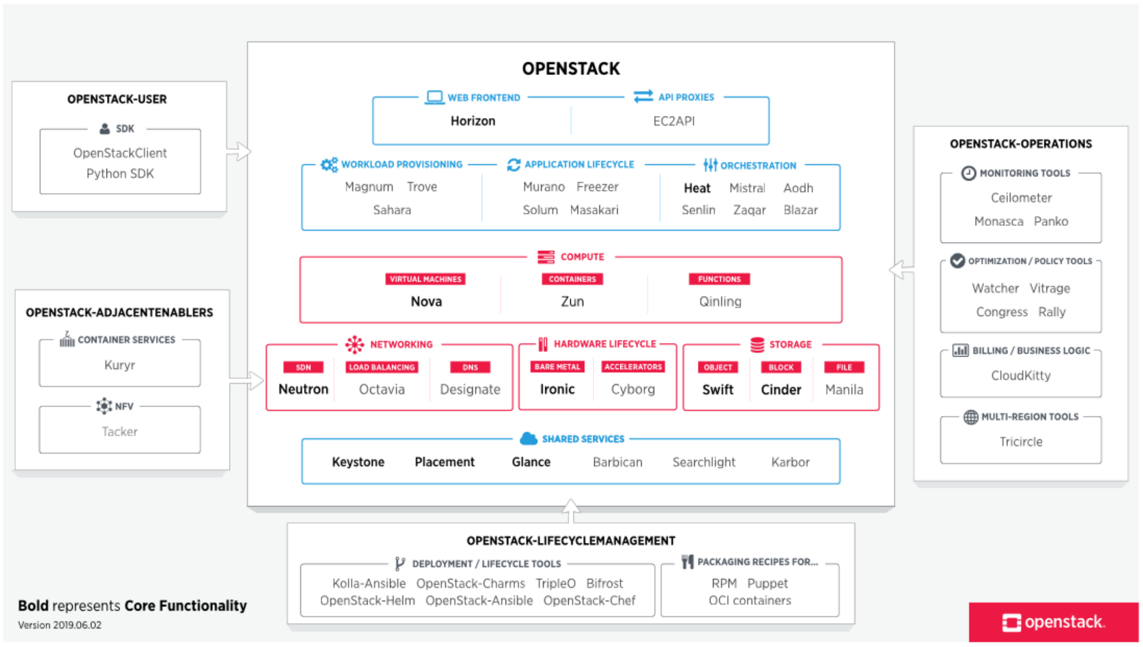


Figure 2-24. Openstack environment.[34]

As it can be seen in *Figure 2-24. Openstack environment*. OpenStack is divided in several modules, each of them with a specific functionality that works as a whole due to the implementation of several APIs. The OpenStack administrator can decide which modules to install (starting from core functionality), in the following table the core modules are described:

ICON	NAME	DESCRIPTION
🐢	KEYSTONE	Provides service/client authentication through API endpoints, service discovery and oversees the management of OpenStack users, tenants, roles and projects. It supports several protocols: LDAP, OAuth, OpenID Connect, SAML and SQL.

34 https://www.openstack.org/software/

GLANCE	Includes various services such as discovering, registering and retrieving of VM images and disks. This VM images can be stored in a variety of locations: object-storage system (Swift) or a simple filesystem.
PLACEMENT	Provides an HTTP API for tracking cloud resource inventories and usages to help other services effectively manage and allocate their resources.
NOVA	Allows direct access and operations to the compute resources of the infrastructure (bare metal, VMs and containers). It is composed by four main elements: nova-compute, nova-scheduler, nova-conductor and nova-api.
NEUTRON	SDN module in charge of providing networking as a service (NaaS) in virtual compute environments. Enables control over the creation of virtual/hybrid networks, virtual routers, IP allocation, VPN creation, port forwarding, SFC, etc.
HORIZON	Canonical implementation of OpenStack's dashboard, it uses the APIs to communicate with other services and is completely customizable.
CINDER	Block-storage service that manages virtualized block storage devices using a self-service API allowing users to request those devices without knowing where they are actually deployed.
SWIFT	Distributed blob/object-store service that allows the storage of massive amounts of data in an efficient, safe and cheap manner.

Table 2-13. OpenStack core Services.

2.6.2 MANO

This sub-section has its scope on presenting a couple of MANO platforms which could be used to develop this Master Thesis.

2.6.2.1 Open Network Automation Platform

The Open Network Automation Platform (ONAP)[35] is an Open Source project that offers a common automation platform for telecommunication, cloud service providers and solution providers that allows to automate various lifecycle processes, to deploy and operate network instances and to create/orchestrate/manage VNFs in a virtual environment. Moreover, ONAP has released several versions, been the most used the "*Casablanca*" version and the most recently released "*Dublin*" *(4.0.0-2019)* version.

ONAP has developed its platform in accordance with the following principles:

- Full-service lifecycle orchestration
- Metadata and policy-driven architecture that enables sourcing best-in-class components
- Elastic and scalable (horizontal and vertical scalability) architecture

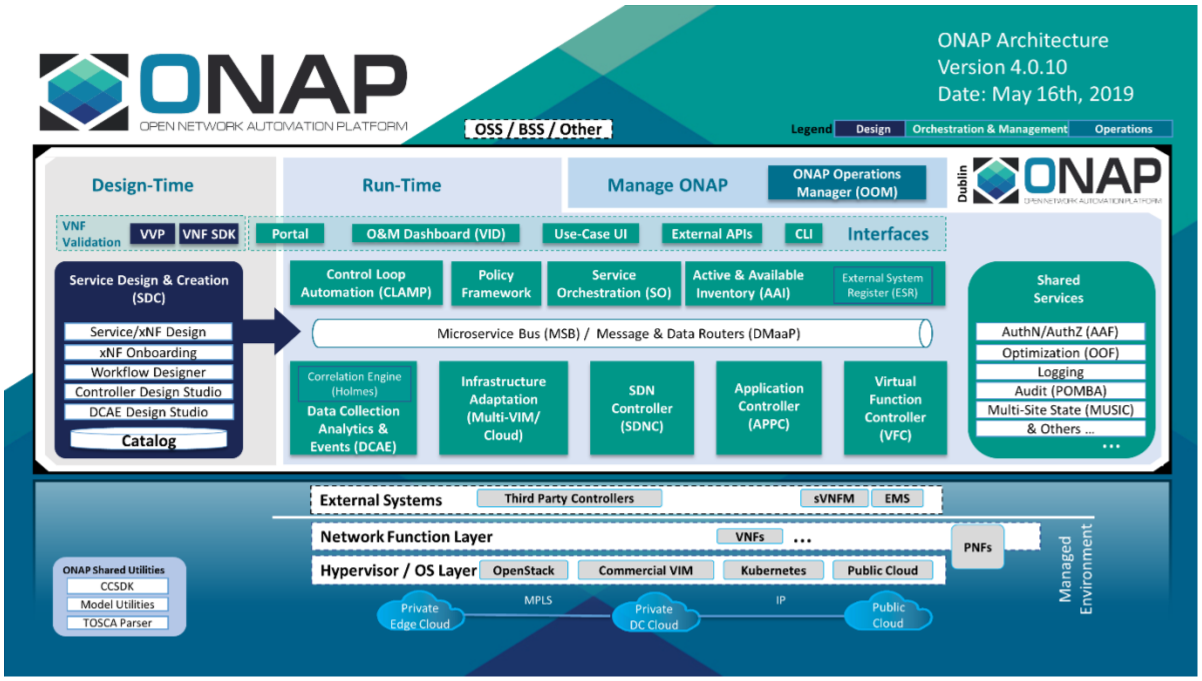


Figure 2-25. ONAP "Dublin's" architecture.

35 https://www.onap.org/

One of the most important features since the release of ONAP Casablanca is the support for NS management, vital for this Master Thesis, which is done by extending the cloud sharing resources to share their NFs and services. Slices have the following features in this SW:

- **Architecture Integrated Design**
- **Support for:**
    - Service Aggregation
    - SFC
    - Service Modification Capability
- **Monitoring of complex nested slice segments, NSIs and Slice services**
- **E2E slicing** and slicing services
- **Enhanced "Active and Available Inventory"** for storing slice segments (RAN, transport and core) and instances states
- **Enhanced "Service Design and Creation"** to define slice segments and mobility services

2.6.2.2 SONATA

SONATA is an Open Source framework born with the SONATA 5GPPP Phase 1 project and extended in the 5G-TANGO 5GPPP Phase 2 project (see *section 2.5.3 5G-TANGO*). The framework is composed by the three sub-systems presented in *section 2.5.3 5G-TANGO*: SDK, SP and V&V. The most important sub-system in the current section is the SP as it enables to manage and orchestrate Network Services. Currently, SONATA Release 4.0 is available with the following features:

- **Catalogue-driven:** Allows to storage VNFs and NSs in order to be used in later phases of the LCM.
- **Multi-VIM/WIM:** Supports multiple VIMs/WIMs (OpenStack, VTN…).
- **Policy-Driven:** Enables autonomous management through policies based on rules and monitoring events.
- **SLAs Assurance**
- **MANO basic:** Supports MANO basic functionality (LCM of VNFs/NSs by different roles).
- **Management Portal**
- **RESTful APIs:** ETSI NFV aligned, exposes SP's capabilities through these APIs.

As it can be observed NS support is not available in SONATA Release 4.0.0, this feature was supposed to be available with SONATA Release 5.0 in July 2019 but, by the time the author is writing this paragraph (august 2019) this release continues to be unavailable. The major features SONATA Release 5.0 is expected to have are:

- Support for NS, NSIs and NSTs.
- Deployment Flavors

- Kubernetes as a VIM
- QoS management and control over NS and network services
- Ingress and Egress endpoint
- Licensing
- Portal Enhancements

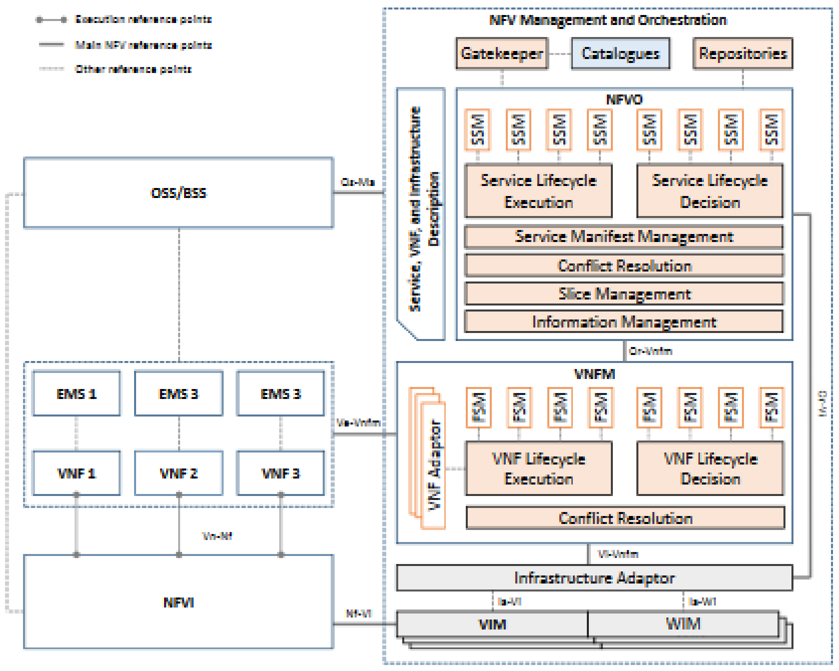


Figure 2-26. SONATA architecture mapping to ETSI reference NFV architecture.

Although SONATA is a promising SW, it was rapidly discarded for the development of this Master Thesis as it wasn't clear when SONATA Release 5.0 was going to be truly released and because of the lack of support for NS in the current release.

2.6.2.3 Open Source MANO

ETSI-OSM[36] is an ETSI-hosted dedicated initiative that aims at the generation of an E2E Network service Orchestrator aligned with ETSI-NFV specifications. To achieve this objective OSM has four key features:

- **ETSI-NFV aligned IM:** Infrastructure agnostic and capable of modelling and automating the LCM process for NFs, Network services and NSIs.
- **Unified NBI:** NFV SOL005 aligned, allows LCM of NSs and NSIs.
- **Network Service extended concept:** Multi-domain NS spanning is enabled.
- **Full LCM support**

Since OSM release FOUR, several upgrades have been added such as support for 5G NS through NSTs (still not aligned directly with the definition of ETSI-NFV Network Services completely), new micro-service-based architecture, orchestration for PNFs and hybrid elements is enabled. Furthermore, in July 2019 OSM Release SIX was released with a lighter orchestrator framework with Network Services and Slicing capabilities, enhanced GUI with a service composer, expansion of the cloud-native micro-service architecture with kafka bus asynchronous communications and performance, fault and policy management enhanced features, as well as WAN Infrastructure Manager (WIM) support. The main problem of this latest release is that it has not been tested enough yet and some issues are arising.

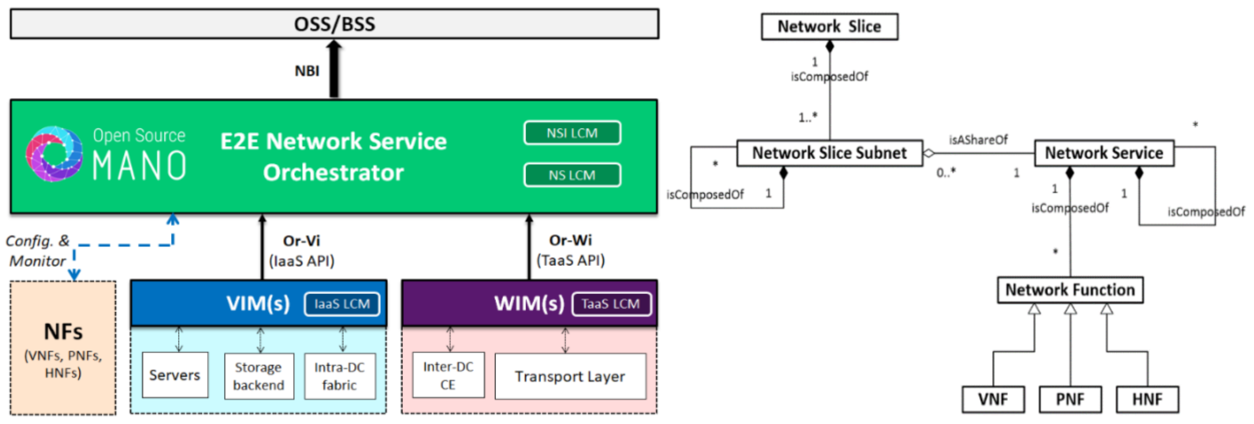


Figure 2-27.OSM service platform view and relation with ETSI-NFV network service concept.

36 https://osm.etsi.org

2.6.2.4 Open Baton

Open Baton[37] is an Open Source project developed jointly between Fraunhofer FOKUS and TU Berlin, it aims at developing an extensible and customizable framework following a modular approach, as an implementation of the ETSI-NFV MANO specification, with an NFV orchestrator, multiple VIM drivers (OpenStack, docker…) and a generic VNF manager. One of the main differentiating features of this platform is that it allows to generate network service descriptions through both: ETSI-NFV and TOSCA models.

Aside from the generic MANO functionality, it incorporates the so called Network Slicing Engine (NSE) which allows the instantiation of rules over physical networks for a determined bandwidth per Network service requirements, it also ensures QoS and it is run as an external component that communicates with the NFVO via Open Baton's SDK.

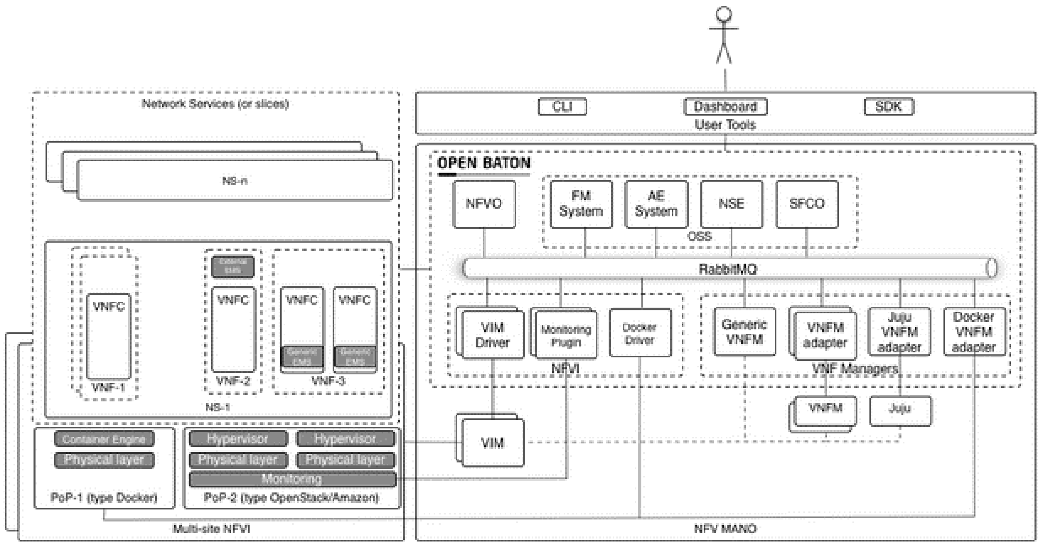


Figure 2-28. Open Baton architecture.

37 https://openbaton.github.io/

2.6.3 Radio

This *Chapter 2 - State of Art*'s final sub-section is dedicated to present the current Open Source Radio available SW. Two platforms are compared:

- **srsLTE[38]:** Is a free and Open Source 4G LTE SW suite developed by Software Radio Systems (SRS)[39] that allows the customer to build an entire E2E SW radio mobile network. srsLTE suite is composed by four main components:
  - **srsUE**
  - **srsENB**
  - **srsEPC:** Includes MME, HSS and SP-GW modules.
  - **Highly modular set of libraries:** For PHY, MAC, RLC, PDCP, RRC, NAS, S1AP and GW layers.

  Release 19.06 has brought several upgrades compared to previous versions; from a complete suite LTE Release 10 compliant to a new webpage (previous versions were hosted on a GitHub webpage) with a community, package repositories for Ubuntu, and organized and clear documents and manuals.

- **Open Air Interface (OAI)[40]:** Is a free Open Source flexible platform created by the OAI Software Alliance (OSA) that aims at developing a LTE/5G ecosystem that follows the full protocol stack of 3GPP standards both, E-UTRAN and EPC. It is composed by the same modules than srsLTE with the difference that this platform incorporates an extra WG called OAI NR that work on evolving the platform towards 5G NR standards and specifications. Although its webpage and GitHub repositories are not the best content-organized webpages, it is a highly customizable SW and it is looking to incorporate 5G features apart from the implemented 4G LTE ones.

The following two figures illustrate both platforms schemas/block diagrams:

38 https://www.srslte.com/

39 http://www.softwareradiosystems.com/

40 https://www.openairinterface.org/

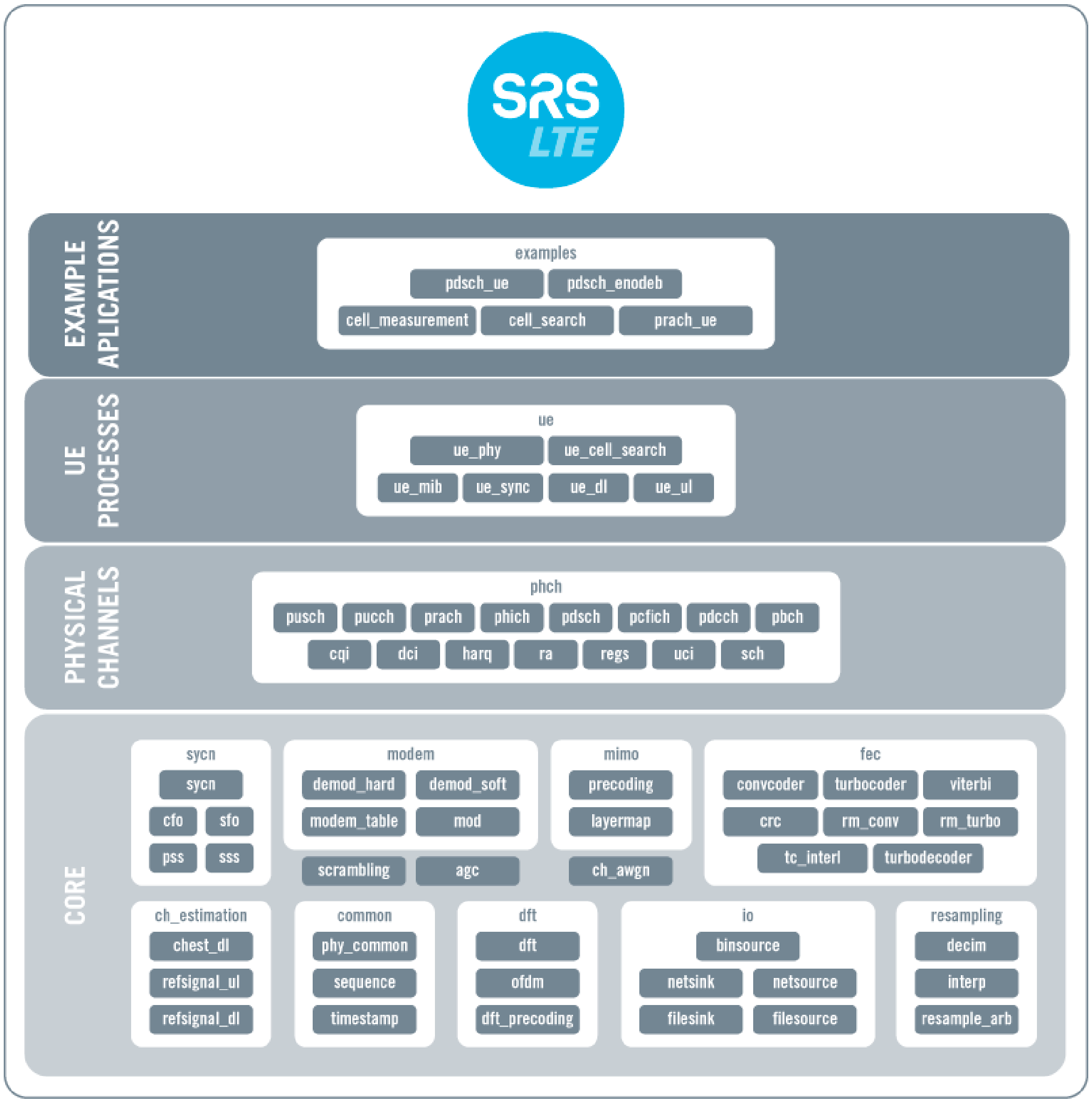


Figure 2-29. srsLTE Block Diagram.

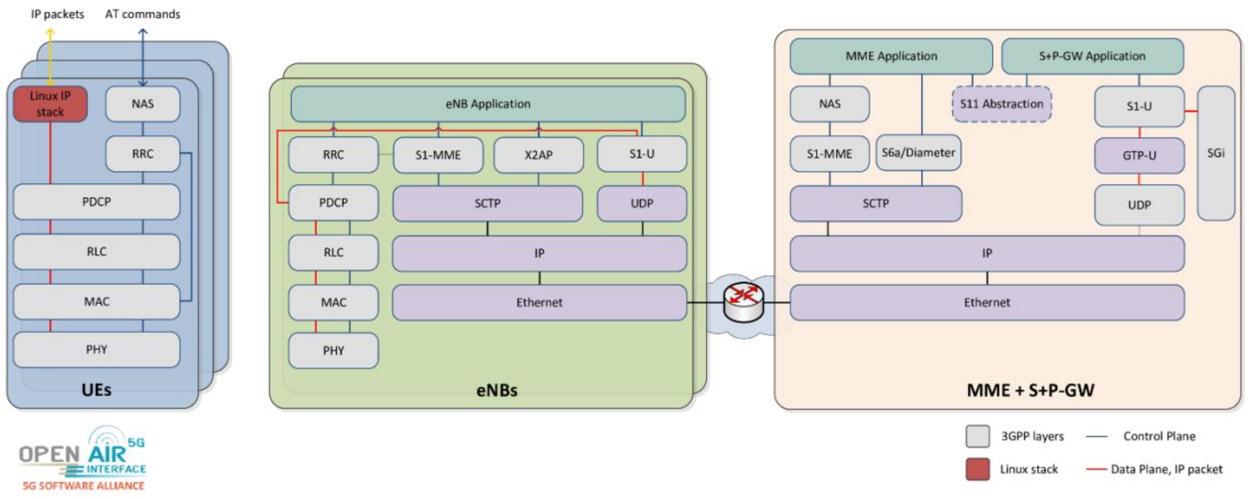


Figure 2-30. OAI Block Diagram.

Finally, a table comparing both platforms features is presented:

	SRSLTE (v19.06)	OAI (latest)
LTE RELEASE	10	8.6 with a subset of release 10 features
FDD AND TDD CONFIGURATIONS	√	√
TRANSMISSION MODES	1-4	1 (SISO), 2, 3, 4, 5, 6 and 7 (MIMO)
DL CHANNELS	PSS, SSS, PBCH, PCFICH, PHICH, PDCCH, PDSCH	PSS, SSS, PBCH, PCFICH, PHICH, PDCCH, PDSCH, PMCH, MPDCCH
UL CHANNELS	PRACH, PUSCH, PUCCH, SRS	PRACH, PUSCH, PUCCH (format 1/1a/1b), SRS, DRS
BANDWIDTH	1.4, 3, 5, 10 and 20 MHz	5, 10 and 20 MHz
CARRIER AGGREGATION	√	√
UE QOS	√	x
MIMO	2x2	4x4

Table 2-14. Radio platforms feature comparison.

3 Technical Development

In this chapter it is going to be explained and justified, in first place, the final SW platforms that have been used to develop this project after the "Research&Analysis" stage[41] was finished and, besides, another section is going to be dedicated to give a detailed explanation about the four generated scenarios, covering their characteristics, requirements and configuration.

3.1 Platforms and Tools

As in every project, before commencing the development, it is necessary to carry out an extensive research about the topic/field that it is going to be implemented gathering information and knowledge in such a manner that, once this stage is over, enough knowledge is acquired to take certain and efficient decisions about the HW and SW involved in the future development.

Aside from all the information given in the previous section about the existing Open Source Tools and platforms, this section aims at justifying the selection of the final set of tools/platforms that has been utilized and to give extended information in some of the cases e.g. OpenStack extended modules, Devstack platform and OSM performance and fault management stacks. On the following sub-sections, the functionality and justification of these platforms is shown.

3.1.1 Cloud Infrastructure

After an extensive research and comparison period, it was decided to use OpenStack as the Open Source cloud computing platform for this Master Thesis, the main reasons for this decision are listed below:

- Other stable cloud solutions such as Azure or AWS apply billing to the consumed services, they are not Open Source and they are run by a 3rd party entity. Besides, high levels of control over the owned infrastructure are required so, avoiding second-control steps is preferred.
- OpenStack has one of the most implicated and vast communities and this has a major consequence: quick upgrades and issue resolving; and good documentation.
- OpenStack supports a lot of customizable features that MAY be implemented installing the proper module, in most of the cases, independently of the rest of the modules.
- It is used by several of the 5GPPP previous projects as the NFVI infrastructure manager and by several of the partners in the 5G-VINNI project. Thereupon, as the scenarios and

[41] See *Annex A Planification.*

tests developed in this Master Thesis are going to be used as an input or even, in the case of production scenarios, as the base infrastructure for the Spanish Facility Site of the 5G-VINNI project; it is important to have into account the previous experience of these projects and the path taken by the other partners (at least for the production scenarios).

- Compared to other Open Source cloud computing platforms such as: OpenShift[42], Cloudify[43] or Tsuru[44]; OpenStack offers much more customizable, programmable and configurable options. Moreover, it supports more plugins than its "rivals".

Once the decision was taken, the next step was to stablish the requirements of the desired scenarios and relate those needs to the available modules of the OpenStack platform. Three scenarios were contemplated at the beginning, but four were created at the end due to some issues encountered in the first production-oriented scenario. There is a dedicated section in this chapter where the requirements of each scenario are described, for now, it is sufficient to say that for the first two scenarios the computing requirements were low and, in consequence, instead of using an OpenStack full stack, DevStack[45] was used. Furthermore, for the production-oriented scenarios, more computing resources were required and even more functionality had to be added to the standard OpenStack modules described in section *2.6.1.1 OpenStack*.

In *Table 3-1. Extended OpenStack final features*. DevStack characteristics and the new modules functionalities are described.

Table 3-1. Extended OpenStack final features.

ICON	NAME	DESCRIPTION
	DEVSTACK	Set of scripts that allows the cloud administrator to quickly deploy an OpenStack environment from git source trees. It is a project that aims at prototyping, developing, coming to know the OpenStack environment and deploying light versions of an OpenStack full stack. Used only on the Mock and Testbed scenarios.
	HEAT	Resource orchestrator module based on text-file templates and that provides both, an OpenStack-native REST API and a CloudFormation compatible API. At first, this module was not planned to be implemented but, in the middle of the development of this project the testing partners of the 5G-VINNI project (EANTC and

42 https://www.openshift.com/

43 https://cloudify.co/

44 https://tsuru.io

45 https://docs.openstack.org/devstack/latest/

		KeySight) announced that they were going to use this module to test the NFVI/NFVO instead of the orchestrator owned by each facility. So, therefore, it had to be installed in both production-oriented scenarios
	CEILOMETER	Metering and Data collection module, whose main goal is to collect and efficiently translate data provided by the different OpenStack modules and allow various representations of this data through 3rd parties SW e.g. Grafana. It is required by OSM if Performance and Fault Management components are installed with it (used in production-oriented scenarios).

3.1.2 MANO

Once again, one the most determining factors at the time of choosing the MANO platform was to be sure that it has support for NS and, for the production-oriented scenarios, consider the other main partners options. Due to these factors SONATA platform was the first to be discarded due to the lack of NS support; afterwards, the decision between the three remaining platforms was highly complex because of the following reasons:

- OSM was at its Rel. FOUR when the author of this Master Thesis started to write the draft of this project and, consequently, OSM did not had support for NS, but it was used by one of the main partners in the 5G-VINNI project (Patras University).
- Open Baton had support for NS and allowed high flexibility when it came to define a VNFD/NSD, but it was not used by any of the partners in the 5G-VINNI project.
- ONAP "Casablanca" version, as OSM Rel. FOUR lacked support for NS and it had been used by previous 5GPPP projects but no partners in the 5G-VINNI projects.

Fortunately, by the time the author of this Master Thesis ended its draft, OSM Rel. FIVE was released with a completely renovated micro-service architecture (the lightest of the three options), full NS support and a huge and complete IM for VNFDs, NSDs and NSTs. Therefore, in order to maintain uniformity in all the scenarios avoiding the use of different orchestration platforms through the different scenarios, following the advises and experience of the 5GinFIRE project and because of potential direct compatibility with Patras University in the 5G-VINNI project; OSM Rel. FIVE was selected as the orchestration platform.

OPEN SOURCE PLATFORM	**OSM REL. FIVE**	**SONATA REL. 4**	**OPEN BATON**	**ONAP "CASABLANCA"**
CAPABILITY EXPOSURE	√	x	√	√

METRICS COLLECTION	√	x	x	x
E2E NS SUPPORT	√	x	√	√
ETSI ALIGNED/RELATED	√	√	x	√
SFC SUPPORT	√	x	x	√
MULTI-VIM	√	√	√	√
WIM	x	x	√	√
PORTAL	√	√	√	√
MICROSERVICE ARCHITECTURE	√	x	x	x

Table 3-2. Open Source Orchestration Platforms comparison.

As it can be seen in the above table, aside from the before-mentioned arguments, OSM Rel. FIVE was the best option compared with the features of the other platforms. Besides, WIM support was to be added (and it was) in the former OSM Rel. SIX.

3.1.3 Radio

As it has been shown in *Table 2-14. Radio platforms feature comparison.*, the differences between both radio platforms are minimal, but the main point here is that these differences are between the current versions of both platforms (srsLTE v.19.06 was released on July 23 which means, it was released almost in the end of the estimated lifespan of this project). srsLTE previous version was far away, from a feature point of view, from the current release and, although, the SW is, by far, much easier to configure, set-up and run, its capabilities were insufficient for the project. Furthermore, despite the complex steps that are required to configure OAI correctly and the awful documentation it has, these platforms offers two main points that srsLTE lacks: high programmability and configurability, and a current branch researching on 5G-NR.

Because of the exposed reasons, the selected radio platform is OAI.

3.2 Scenario Design

In this section all the developed scenarios are detailed, beginning from its HW and SW requirements, to its configuration. At first, three scenarios were expected:

1. **Initial Scenario:** Mock scenario oriented to allow the author to know the MANO and cloud platforms and how they interact between them, as well as how to configure them to run as desired.
2. **Testbed Scenario:** Formal scenario with more computing resources which allows to develop more complex experiments.
3. **Production-Level Scenario:** Final scenario, complete capabilities are required, it is going to host the 5G NSA trial. This scenario will also be used as part of the 5G-VINNI Spanish Facility infrastructure.

However, the first design of the last scenario had some important issues and one more production-level design was carried out.

One sub-section is dedicated for each scenario.

3.2.1 Initial Scenario

The main objective of this scenario is to get in contact with the selected cloud and MANO platforms, designing a light-dependent resource computing scenario where the main functionalities and features of both platforms can be checked, tested and "played with" in order to obtain the basic knowledge needed to operate more complex scenarios. At first, this scenario was attempted to be implemented on a relatively old laptop with 8GB of RAM but, due to the minimum HW requirements of both platforms (explained bellow) it was impossible to run this scenario in that computer. Finally, the Laptop mentioned in *Section 9.2.1 Hardware* was used to build up this scenario.

Furthermore, another objective of this scenario was to obtain a completely virtualized environment and study the overall its performance to see if was implementable in a production-level scenario.

In *Figure 3-1. Initial Scenario Network Structure*. a schema of the scenario's network structure and components is pictured.

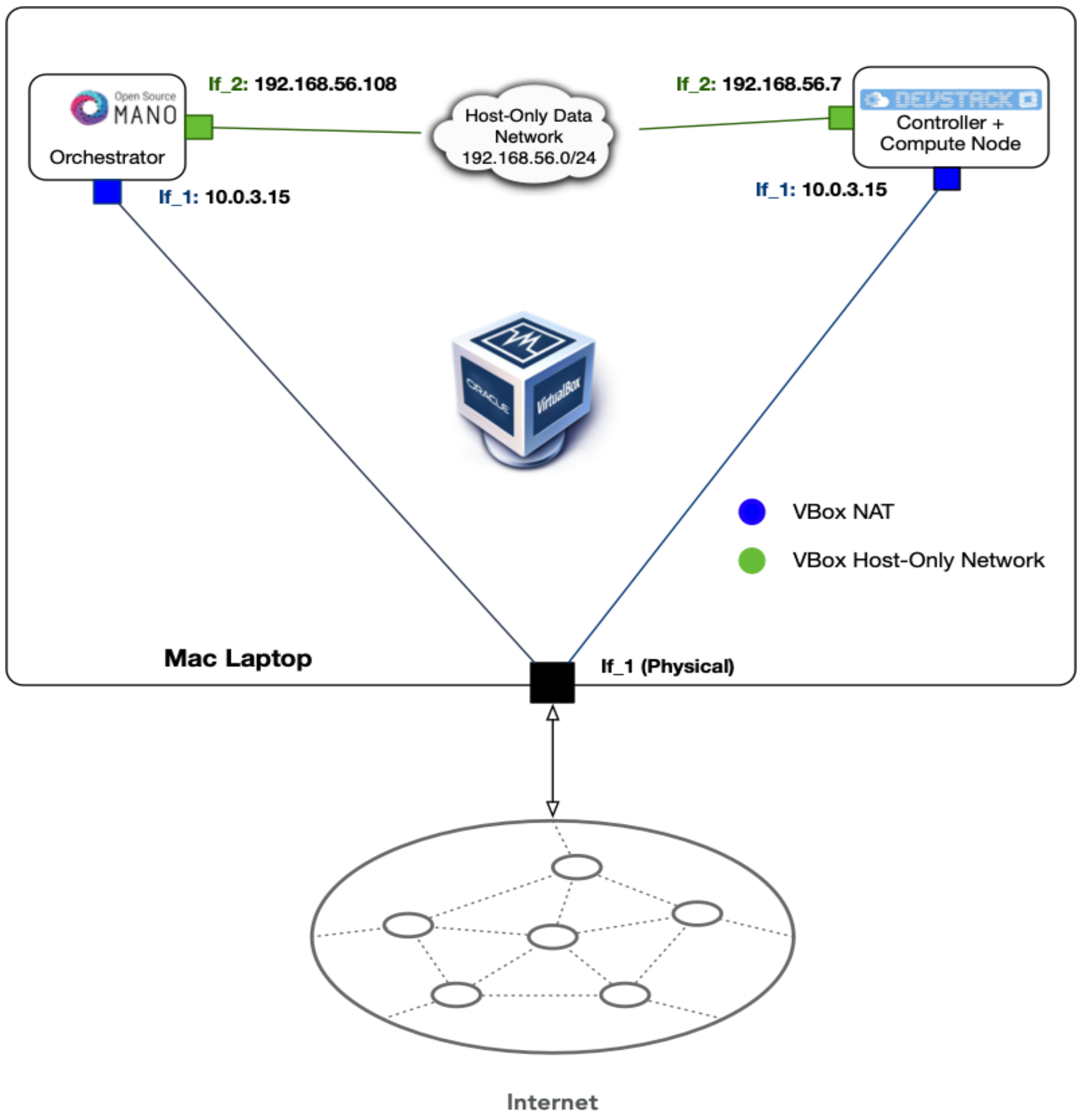


Figure 3-1. Initial Scenario Network Structure.

As it can be seen, two VMs were generated, both hosted inside the Laptop device:

- **OSM VM:** Acts as the orchestrator of the cloud computing platform – DevStack.
- **DevStack:** Acts as a development Stack of the OpenStack environment.

3.2.1.1 HW Requirements

Although this is supposed to be the lightest scenario, the minimum requirements needed by both platforms cannot be ignored even in a computer with 16GB of RAM. In this section, a table where the minimum required HW and the current dedicated HW of each of the VMs/platforms is shown:

	DEVSTACK MIN. REQUIREMENTS	DEVSTACK VM HW	OSM MIN. REQUIREMENTS	OSM VM HW
RAM (GB)	8	6	4	6
CPUS	1	1	2	2
DISK (GB)	40	80	20	40

Table 3-3. Initial Scenario HW.

As it has been mentioned, this scenario is ran over the Laptop device (16GB RAM, 4 cores and 1TB disk), which means that there was a limited amount of RAM and CPUs to be shared between the VMs and the host. After several trials, it was discovered that the host system crashed if less than 4GB of RAM were left for the host system and, besides, although it can be striking that the DevStack VM uses less RAM than the minimum recommended, it was discovered that the overall performance of the VM and the platform was not affected by this change (except for the fact that less VMs could be generated by DevStack) and the performance of the OSM VM was increased severally (with 4GB of RAM even trying to write in the console was a titanic task) if this 2GB of RAM were given to it.

3.2.1.2 SW Requirements

For this scenario it was decided to use VirtualBox as the hypervisor as it is an easy solution previously known by the author, it can be launched over Mac and it comes with enough features to cover the requirements of this scenario[46]. Furthermore, instead of implementing a full OpenStack Stack, as this was a first approach to the platform, it was decided to use DevStack directly to ease the configuration process.

Each platform has certain OS requirements to carry out its installation, thus avoiding problems derived from the versions of the packages:

OSM	DEVSTACK
Ubuntu 16.04 Xenial	Ubuntu 18.04 Bionic

Table 3-4. Required OSs.

These OSs have been used as the base OSs for the respective VMs in this scenario.

46 If a disadvantage must be pointed out, it should be remarked that OpenStack/DevStack uses KVM/QEMU as VIM and therefore, as VirtualBox does not support nested virtualization with KVM, the second level of virtualization is carried out with QEMU which does not support HW acceleration, resulting on a slight performance degradation.

3.2.1.3 Configuration

3.2.1.3.1 Network Configuration

Focusing on the objective of obtaining a full virtualized environment, all the network configuration has been done through the hypervisor. Both VMs have two network interfaces:

- **Management Network Interface**: Used for managing VM functionalities, upgrading packages and maintaining an Internet connection. "NAT VirtualBox network connection" was configured for this interface as it isolates the management connection from the other VMs and host; and gives access to the Internet virtually "nating" traffic through the host's physical interface.
- **Internal Data Network Interface:** Used to hold the VM2VM and the VM2Host communication. Implemented generating a "Host-Only" network in VirtualBox and assigning an interface in each VM to it. It is vital in order to have access to the GUIs/portals offered by each VM via web.

3.2.1.3.2 OSM Configuration

Configuring and installing OSM Rel. FIVE is a straightforward process, very few steps are required. Bellow, the commands needed to install OSM Rel. FIVE through bash are specified[47]:

```
$ wget https://osm-download.etsi.org/ftp/osm-5.0-five/install_osm.sh
$ chmod +x install_osm.sh
$ ./install_osm.sh -t v.5.0.5 2>&1 | tee osm_install_log.txt
```

Text Box 3-1. Basic OSM installation commands.

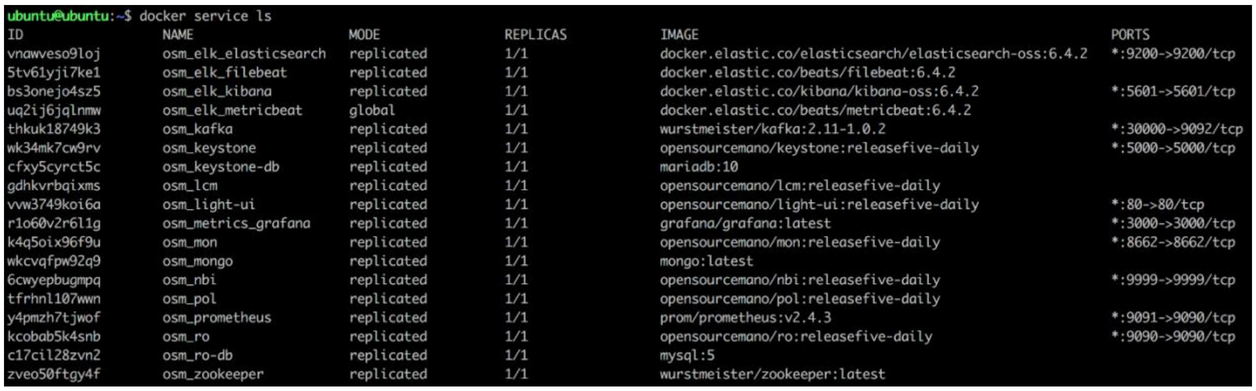
Basically, these commands download the proper installing script, giving to it executing permissions and install the platform leaving a log to debug potential errors. During the installation process some dialog messages will appear: LXD bridge must be configured, an IPv4 subnet must be configured with the appearing default values and an IPv6 subnet MUST NOT be configured. The installation will last between 20-40min depending on the assigned resources, once it is done, the following commands MAY be used to check correct behavior:

```
$ docker stack ps osm |grep -i running
$ docker service ls
$ docker service logs osm_lcm
$ docker logs $(docker ps -aqf "name=osm_lcm" -n 1)
```

Text Box 3-2. Checking OSM installation commands.

[47] Henceforth, when bash code is shown "$" will mean that the code can be executed by any user and if "#" at the beginning of the sentence, the code MUST be executed only by the root user.

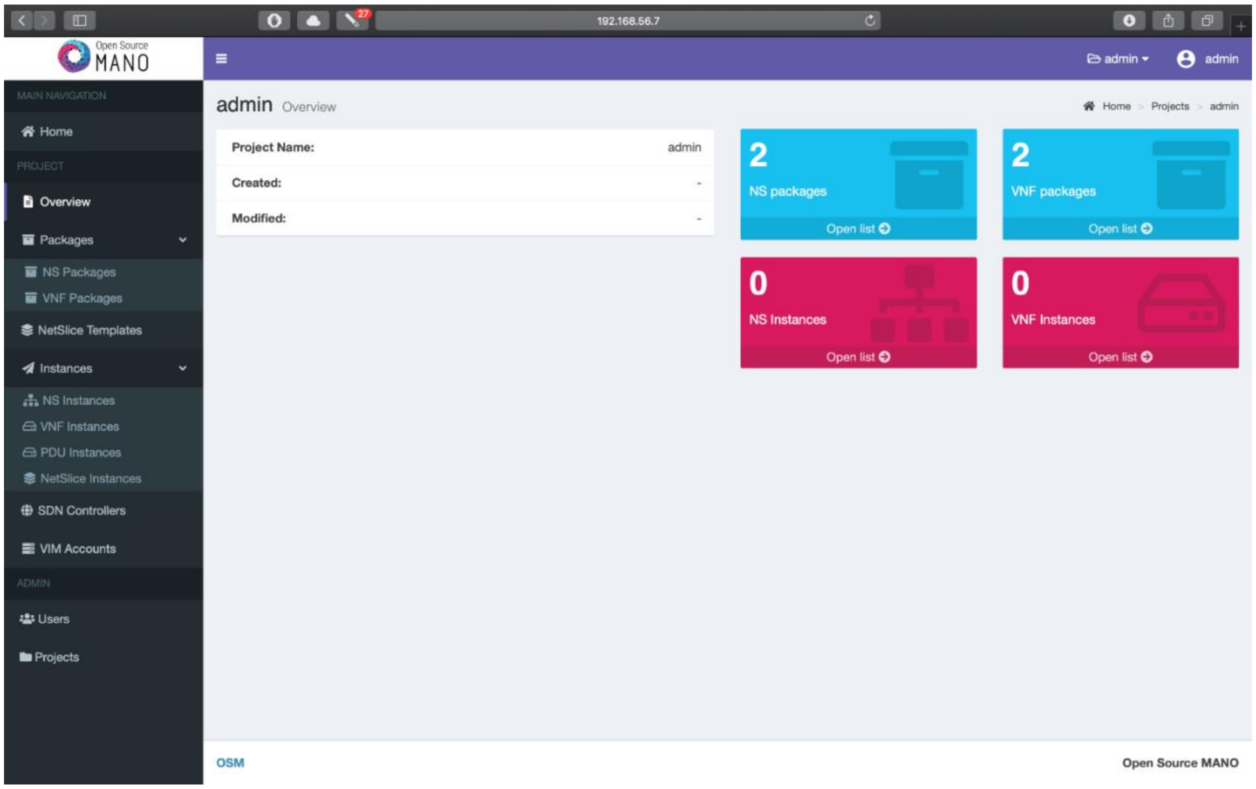
If everything went OK, the following output SHOULD be obtained when executing the second of the above commands:



```
ubuntu@ubuntu:~$ docker service ls
ID             NAME                      MODE         REPLICAS    IMAGE                                                    PORTS
vnawveso9loj   osm_elk_elasticsearch     replicated   1/1         docker.elastic.co/elasticsearch/elasticsearch-oss:6.4.2  *:9200->9200/tcp
5tv61yji7ke1   osm_elk_filebeat          replicated   1/1         docker.elastic.co/beats/filebeat:6.4.2
bs3onejo4sz5   osm_elk_kibana            replicated   1/1         docker.elastic.co/kibana/kibana-oss:6.4.2                *:5601->5601/tcp
uq2ij6jqlnmw   osm_elk_metricbeat        global       1/1         docker.elastic.co/beats/metricbeat:6.4.2
thkuk18749k3   osm_kafka                 replicated   1/1         wurstmeister/kafka:2.11-1.0.2                            *:30000->9092/tcp
wk34mk7cw9rv   osm_keystone              replicated   1/1         opensourcemano/keystone:releasefive-daily                *:5000->5000/tcp
cfxy5cyrct5c   osm_keystone-db           replicated   1/1         mariadb:10
gdhkvrbqixms   osm_lcm                   replicated   1/1         opensourcemano/lcm:releasefive-daily
vvw3749koi6a   osm_light-ui              replicated   1/1         opensourcemano/light-ui:releasefive-daily                *:80->80/tcp
r1o60v2r6l1g   osm_metrics_grafana       replicated   1/1         grafana/grafana:latest                                   *:3000->3000/tcp
k4q5oix96f9u   osm_mon                   replicated   1/1         opensourcemano/mon:releasefive-daily                     *:8662->8662/tcp
wkcvqfpw92q9   osm_mongo                 replicated   1/1         mongo:latest
6cwyepbugmpq   osm_nbi                   replicated   1/1         opensourcemano/nbi:releasefive-daily                     *:9999->9999/tcp
tfrhnl107wwn   osm_pol                   replicated   1/1         opensourcemano/pol:releasefive-daily
y4pmzh7tjwof   osm_prometheus            replicated   1/1         prom/prometheus:v2.4.3                                   *:9091->9090/tcp
kcobab5k4snb   osm_ro                    replicated   1/1         opensourcemano/ro:releasefive-daily                      *:9090->9090/tcp
c17cil28zvn2   osm_ro-db                 replicated   1/1         mysql:5
zveo50ftgy4f   osm_zookeeper             replicated   1/1         wurstmeister/zookeeper:latest
```

Figure 3-2. OSM service list.

Besides, OSM-light-UI service SHOULD allow web-access to its portal:



Project Name: admin
Created: -
Modified: -

2 NS packages
2 VNF packages
0 NS Instances
0 VNF Instances

Figure 3-3. OSM Rel. FIVE portal.

Finally, an additional important step should be taken, after installing DevStack in order to connect both platforms. This action consists on adding a VIM account to the OSM VM to allow the interaction with DevStack, this can be done through the GUI or by command. To better understand this step, it is going to be explained in section *3.2.1.3.4 Adding a VIM account to OSM*.

3.2.1.3.3 DevStack Configuration

As already mentioned, this scenario pretends to be a first contact with a potential 5G sliced environment with few resources and, thus, implementing a full OpenStack stack was out of scope. Thereupon, DevStack offers the perfect solution to this case, as it allows a quick configuration of a "mock" OpenStack environment and brings the opportunity to learn how this environment works. Its installation, as well as OSM, is pretty straightforward but, before digging in that process it is important to clarify that only the basic OpenStack modules are required in this scenario as we are willing to learn how they work, this are:

- Keystone
- Placement
- Glance
- Nova
- Neutron
- Horizon
- Cinder
- Swift

Once the proper VM is configured, the next steps should be followed in order to prepare DevStack's installation process:

```
$ sudo useradd -s /bin/bash -d /opt/stack -m stack
$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
$ sudo su - stack
$ git clone https://opendev.org/openstack/devstack
$ cd devstack
```

Text Box 3-3. DevStack pre-installation commands.

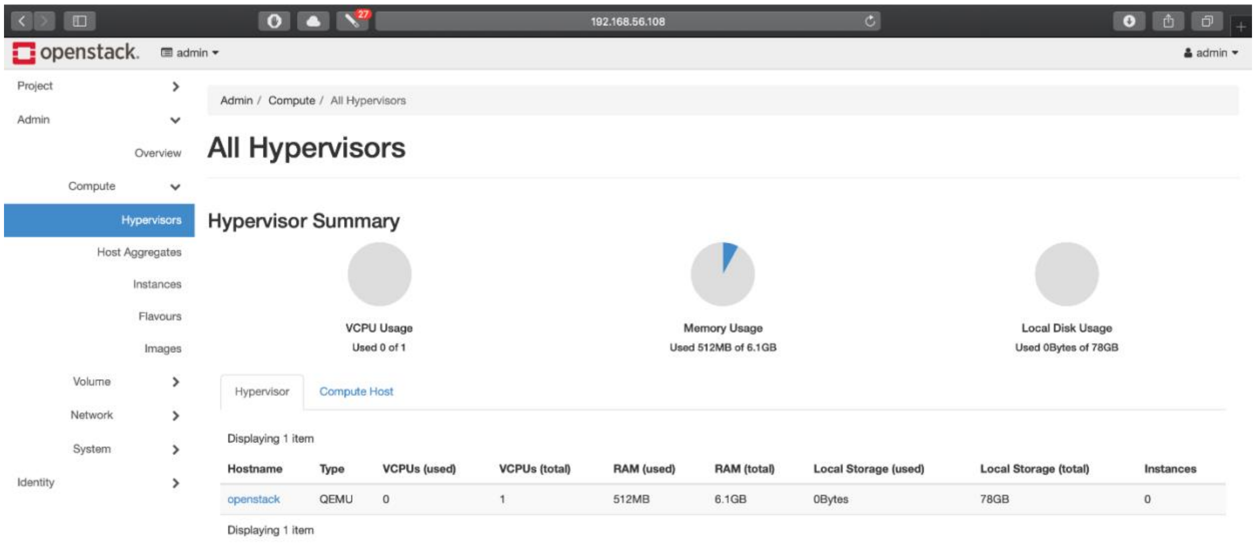
These commands will generate a user named "stack" with the home directory situated in "/opt/stack" directory, give this user sudo privileges, log into the user account and download the necessary DevStack scripts from the GitHub master branch. Afterwards, inside the "devstack" directory multiple files are to be find, but for the time being, the most important one is the one named "local.conf". This file is an INI file that allows direct configuration of the OpenStack environment through local parameters e.g. login paths, user and DDBB passwords, modules to be installed, additional plugins, IP ranges, image downloading, flavor definition, etc.

The *local.conf* file used can be found in the *Annex 10.1 Initial Scenario DevStack "local.conf" file*, its contents configure different administrator and Database passwords (not the most secure ones), establish where it is going to be served Horizon's dashboard (HOST_IP) as well as Floating and Fixed IPs range, activates a log for the installation process, downloads non-default images and configures required Swift parameters. By default, all the basic modules are installed within their latest version, no code ha to be added. Once this is finished, the installation process can start launching the following command at the DevStack directory:

```
$ ./stack.sh
```

Text Box 3-4. DevStack installation Script.

This command will execute and configure several scripts and local variables which will build the entire environment from endpoints and APIs, to networks, gateways and even flavors and images. It will also create admin and demo projects under the "Default" domain, then it is over cli and GUI accesses should be available.



All Hypervisors

Hypervisor Summary

VCPU Usage
Used 0 of 1

Memory Usage
Used 512MB of 6.1GB

Local Disk Usage
Used 0Bytes of 78GB

Displaying 1 item

Hostname	Type	VCPUs (used)	VCPUs (total)	RAM (used)	RAM (total)	Local Storage (used)	Local Storage (total)	Instances
openstack	QEMU	0	1	512MB	6.1GB	0Bytes	78GB	0

Displaying 1 item

Figure 3-4. Devstack dashboard, compute module check.

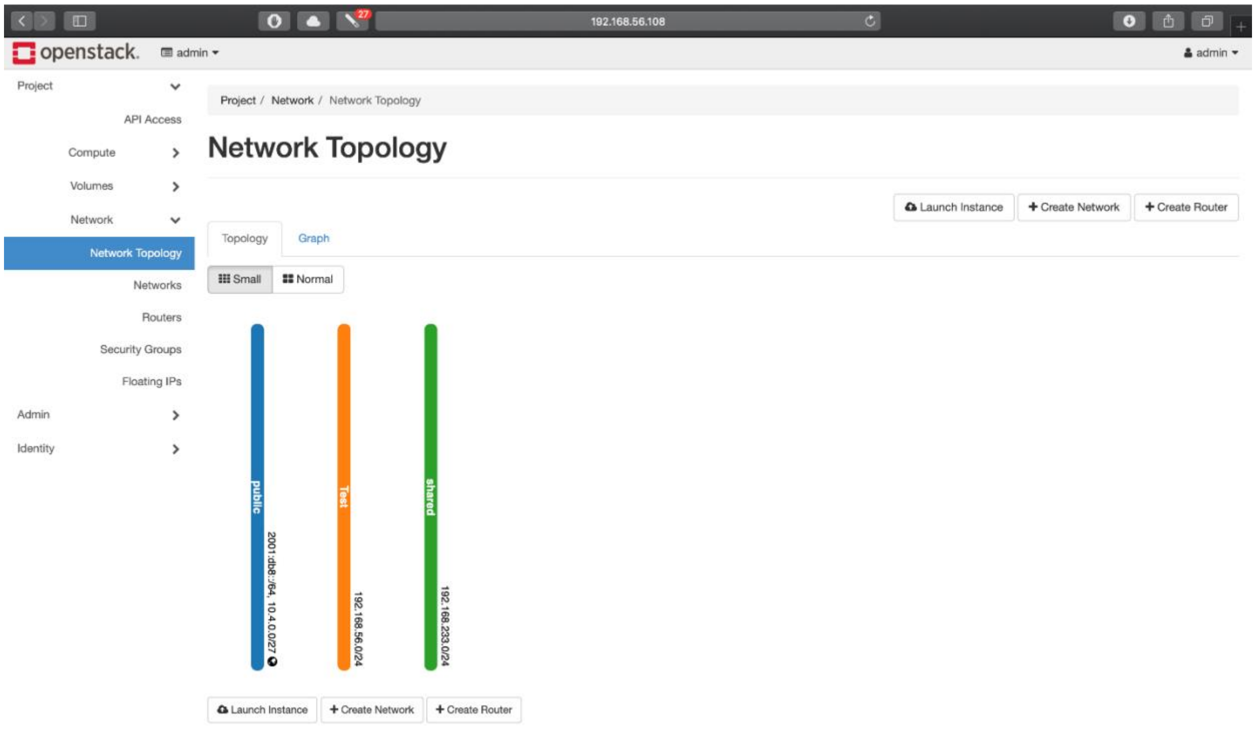
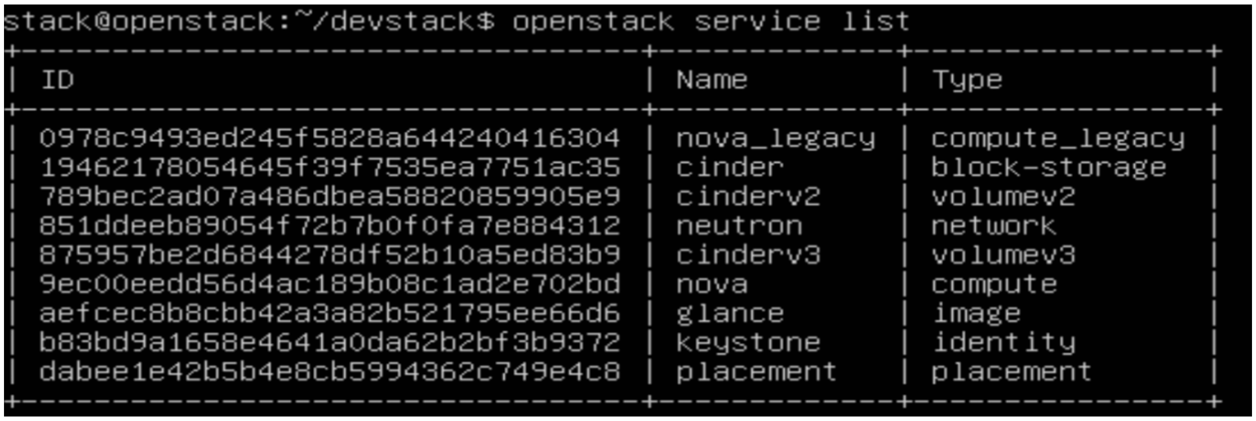


Figure 3-5. DevStack post-installation network topology.



```
stack@openstack:~/devstack$ openstack service list
+----------------------------------+-------------+----------------+
| ID                               | Name        | Type           |
+----------------------------------+-------------+----------------+
| 0978c9493ed245f5828a644240416304 | nova_legacy | compute_legacy |
| 19462178054645f39f7535ea7751ac35 | cinder      | block-storage  |
| 789bec2ad07a486dbea58820859905e9 | cinderv2    | volumev2       |
| 851ddeeb89054f72b7b0f0fa7e884312 | neutron     | network        |
| 875957be2d6844278df52b10a5ed83b9 | cinderv3    | volumev3       |
| 9ec00eedd56d4ac189b08c1ad2e702bd | nova        | compute        |
| aefcec8b8cbb42a3a82b521795ee66d6 | glance      | image          |
| b83bd9a1658e4641a0da62b2bf3b9372 | keystone    | identity       |
| dabee1e42b5b4e8cb5994362c749e4c8 | placement   | placement      |
+----------------------------------+-------------+----------------+
```

Figure 3-6. Devstack service check.

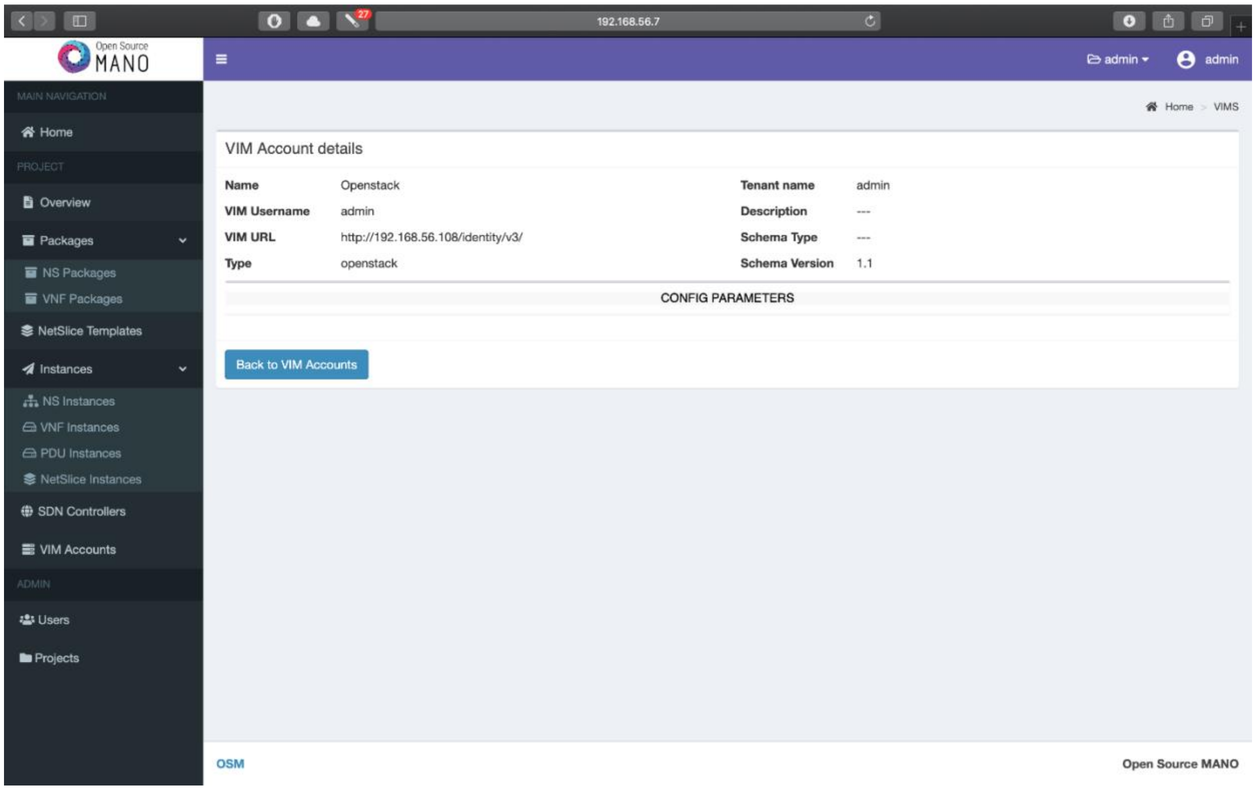
3.2.1.3.4 Adding a VIM account to OSM

At last, as mentioned at the end of section *3.2.1.3.2 OSM Configuration* previous section, once DevStack is completely installed and configured it will have available the authentication endpoint which is the one needed by OSM to orchestrate the LCM of the proper NSs and network services. In this case, the referred endpoint is: *http://192.168.56.108/identity/v3/*. In order to add it to OSM, the following command should be executed in the VM:

```
$ osm vim-create --name openstack-test --user admin --password admin1234 \
    --auth_url http://192.168.56.108/identity/v3/ --tenant admin --
account_type openstack
```

Text Box 3-5. OSM VIM adding.

This command will create a new VIM account at the OSM VM with the given authentication parameters and name it "openstack-test". By default, this command stablishes the project/tenant domain as "Default", if other is used additional parameters should be added (in the production-level scenarios this will be shown).



VIM Account details

Name	Openstack	Tenant name	admin
VIM Username	admin	Description	---
VIM URL	http://192.168.56.108/identity/v3/	Schema Type	---
Type	openstack	Schema Version	1.1

CONFIG PARAMETERS

Figure 3-7. OSM VIM account check.

3.2.2 Testbed Scenario

The second scenario had a clear scope, which is to serve as the main NFVI/NFVO experimental environment for the UC3M, not for the 5G-VINNI project. At first, this scenario was expected to be a fully operative E2E 5G-NSA NS scenario but, due to the HW characteristics of the involved devices, this objective could not be achieved. As a result, in this scenario are going to be carried out all the NFVO Day-0/Day-1 first-trial experiments as it has enough resources to launch them and eases the error correction process because direct access to the infrastructure is available (the scenario is located inside one of the UC3M Telematics' Department laboratories).

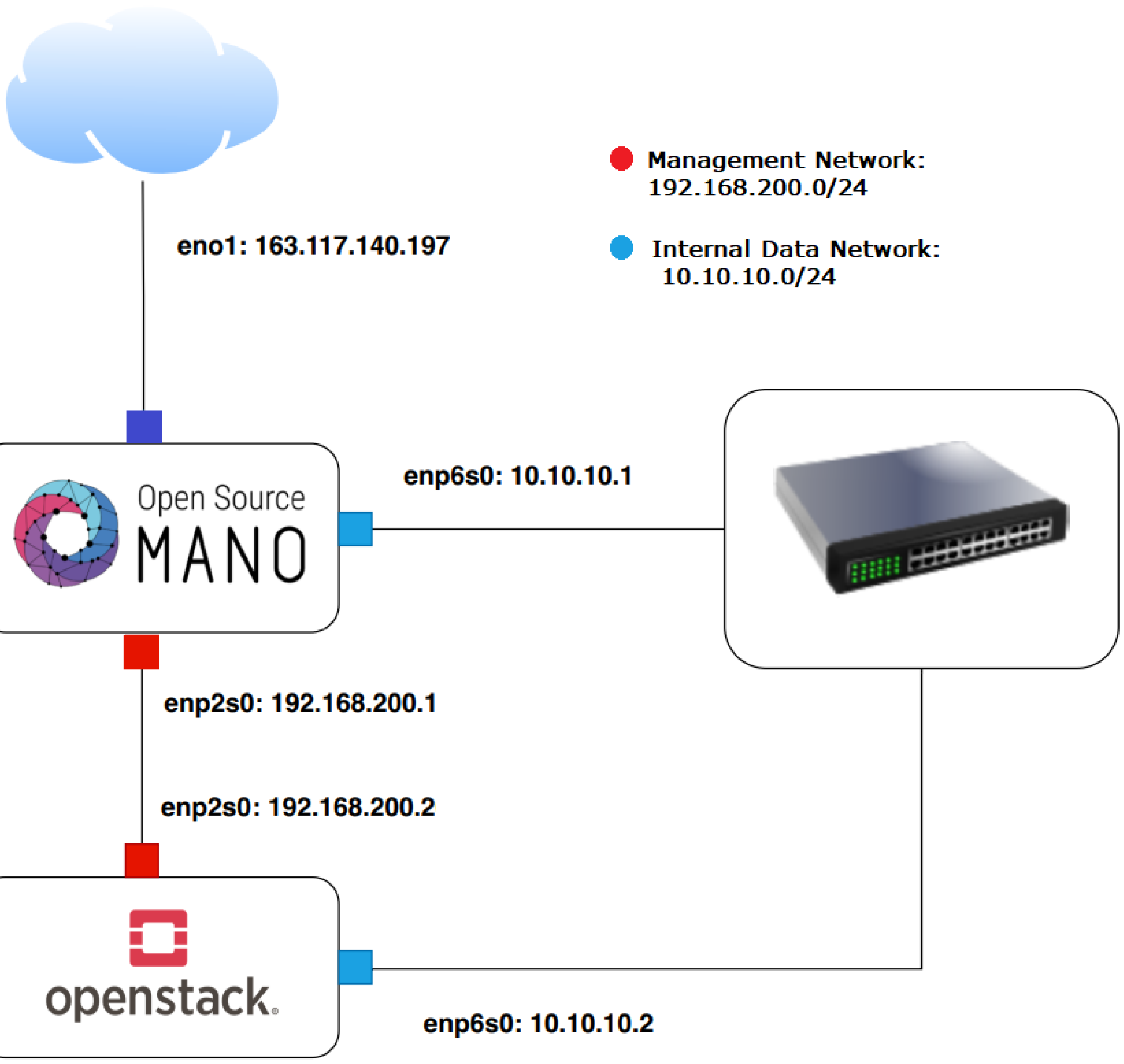


Figure 3-8. Testbed scenario Network Architecture.

Four physical devices are available, therefore, it was decided that this scenario was going to be composed completely by physical devices, more precisely, by four "*Portable Multiport Server*" devices[48]. This scenario requires external access to avoid only a local access from UC3M, thereupon, a public IP was assigned to it; the rest of the devices will work under a NAT router and internal networks. In the above figure, two of these "*Portable Multiport Server*" devices are shown:

- **OSM:** Contains the orchestrator and has the NAT router role.
- **OpenStack:** Implements DevStack.
- **Physical Switch:** Provided for future scalability (currently with two machines is enough to run the experiments, but in the future the two remaining nodes may be added as extra compute nodes).

As a summary, this scenario allows to test innovative NFVO experiments and it is prepared to scale-out if needed.

3.2.2.1 HW requirements

As no virtualized infrastructure is used in this scenario, the HW requirements are directly limited by the physical machines HW specifications. Nonetheless, the HW requirements (especially the Network interfaces numbers) vary slightly compared to other scenarios due to the architecture design:

Machine	CPUs	RAM	Disk	Network Interfaces
Multiport Server	**4**	**8 GB**	**128 GB**	**6**
OSM	2	8 GB	40 GB	3
DevStack	2	8 GB	60 - 80 GB	2
Future Compute Nodes	4	8 GB	128 GB	2

Table 3-5. Scenario 2 HW requirements.

48 See Section 9.2.1 *Hardware* for a detailed explanation of its characteristics.

3.2.2.2 SW Requirements

No hypervisor was required for this scenario, DevStack was used as the OpenStack development stack to ease and accelerate the installation process. Two OSM versions have been used within this scenario.

DEVICE	OS
OSM REL. FIVE	Ubuntu Xenial 16.04
OSM REL. SIX	Ubuntu Bionic 18.04.2
DEVSTACK (LATEST STABLE VERSION)	Ubuntu Bionic 18.04.2

Table 3-6. SW OS reuirements.

3.2.2.3 Configuration

3.2.2.3.1 Network Configuration

This scenario does not use any virtualized infrastructure. Thereupon, as the only machine with a public IP is the OSM machine, this device must act as a NAT router to redirect the traffic to its respective destinies. During the OSM machine OS installation process, its *eno1* interface was configured to use the public assigned IP (*163.117.140.197*); afterwards, two networks were required: an internal network for the data flows between both machines and a network to give Internet access to the virtual public network generated by DevStack.

NETWORK	IP RANGE	GATEWAY
EXTERNAL (PUBLIC IP)	163.117.140.0/24	163.117.140.2
INTERNAL DATA	10.10.10.0/24	10.10.10.1 (enp6s0 OSM machine interface)
PUBLIC INTERNET ACCESS (DEVSTACK)	192.168.200.0/24	-

Table 3-7. Scenario 2 available networks.

In order to achieve this environment, some further configuration had to done in the OSM machine:

- Modify */etc/networks/interfaces* file[49]
- Edit */etc/sysctl.conf* to enable port-forwarding, changing sentence *net.ipv4.ip_forward=0* to value 1
- Apply changes: *"$ sudo sysctl -p /etc/sysctl.conf"*
- Generate required iptables rules to correctly forward the traffic:

[49] See *10.5 Scenario 2 OSM machine /etc/network/interfaces* file.

```
$ sudo iptables -t nat -A POSTROUTING -o enp6s0 -j MASQUERADE
$ sudo iptables -A FORWARD -i eno1 -o enp6s0 -m state --state \
RELATED , ESTABLISHED -j ACCEPT
$ sudo iptables -A FORWARD -i enp6s0 -o eno1 -j ACCEPT
$ sudo iptables -t nat -A PREROUTING -p tcp -m tcp --dport 23 -j \
DNAT --to-destination 10.10.10.2:22
$ sudo iptables -t nat -A PREROUTING -p tcp -m tcp --dport 443 -j \
DNAT --to-destination 10.10.10.2:80
$ sudo iptables -t nat -A PREROUTING -p tcp -m tcp --dport 9696 -j \
DNAT --to-destination 10.10.10.2:9696
$ sudo iptables -t nat -A PREROUTING -p tcp -m tcp --dport 6080 -j \
DNAT --to-destination 10.10.10.2:6080
```

Text Box 3-6. Iptables rules.

With the previous commands NAT is activated for the outgoing traffic on interface *enp6s0*, traffic forwarding is enabled between *eno1-enp6s0* interfaces and vice versa; and, to access DevStack services (Dashboard, Neutron API and console access) from the public IP and *ssh.*

It is also recommended to install the package *iptables-persistent* to maintain the current configuration even if the system reboots or is shut-down. To do so, execute the following commands:

```
$ sudo apt install iptabeles-persistent -y
$ sudo netfilter-persistent save
```

Text Box 3-7. Persistent iptables rules.

Once all this configuration is over, each interface ends with the following configuration:

MACHINE	INTERFACE	IP	GW
OSM	Eno1	163.117.140.197	163.117.140.2
	Enp2s0	192.168.200.1	-
	Enp6s0	10.10.10.1	-
DEVSTACK	Enp2s0	192.168.200.2	-
	Enp6s0	10.10.10.2	10.10.10.1

Table 3-8. Scenario 2 interfaces assignment.

3.2.2.3.2 OSM Configuration

Two OSM versions have been tested in this scenario, at the deployment time OSM Rel. FIVE was installed using the following commands:

```
$ wget https://osm-download.etsi.org/ftp/osm-5.0-five/install_osm.sh
$ chmod +x install_osm.sh
$ ./install_osm.sh -t v.5.0.5 --elk_stack --pm_stack 2>&1  | \
  tee osm_install_log.txt
```

Text Box 3-8. Scenario 2 OSM Rel.FIVE installation.

As it can be seen, the command is pretty like the one used in the first scenario but with the difference of two additional arguments at the installation command that enable the micro-service architecture to add Performance and Fault management features to OSM. At some point, metrics and monitoring results are expected to be used to make automated LCM.

Furthermore, three weeks after the deployment was already completed OSM rel. SIX was released with a lot of promising updates and performance upgrades, as well as an improved IM. Therefore, to avoid installing this new release in the production-level scenario and coming across with unknown issues, it was decided to install this new version in this scenario in order to test its new features and capabilities, check NSDs/VNFDs compatibility, check errors, etc. Although some problems migrating from OSM Rel. FIVE to OSM Rel. SIX have been reported to the OSM community, in this case, none was encountered.

First, all the old OSM services must be shut down:

```
$ docker stack rm osm && sleep 60 # The sleep is for making sure the stack \
removal finishes before upgrading
```

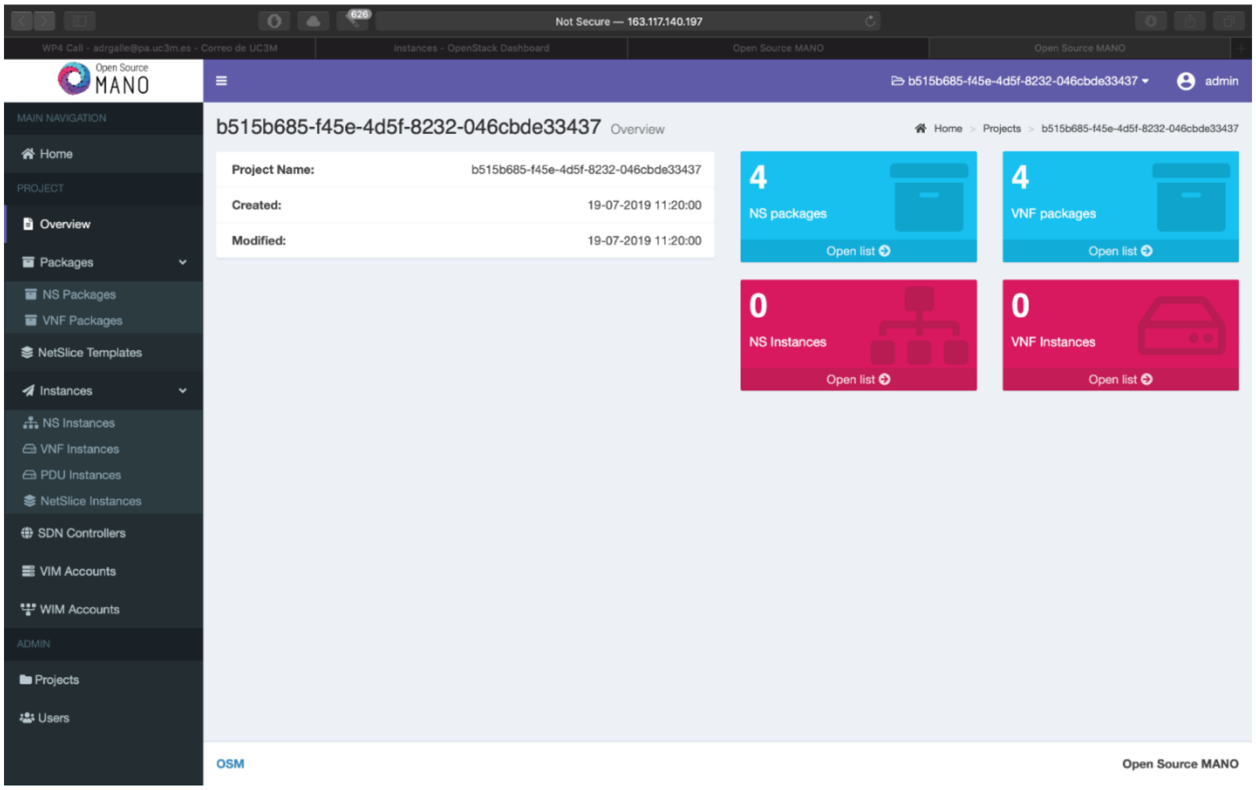
Text Box 3-9. OSM deletion.

Secondly, the new release must be installed:

```
$ wget https://osm-download.etsi.org/ftp/osm-6.0-six/install_osm.sh
$ chmod +x install_osm.sh
$ ./install_osm.sh --elk_stack --pm_stack 2>&1  | \
  tee osm_six_install_log.txt
```

Text Box 3-10. OSM Rel. SIX installation.

Once the process is finished, OSM portal must be accessible:

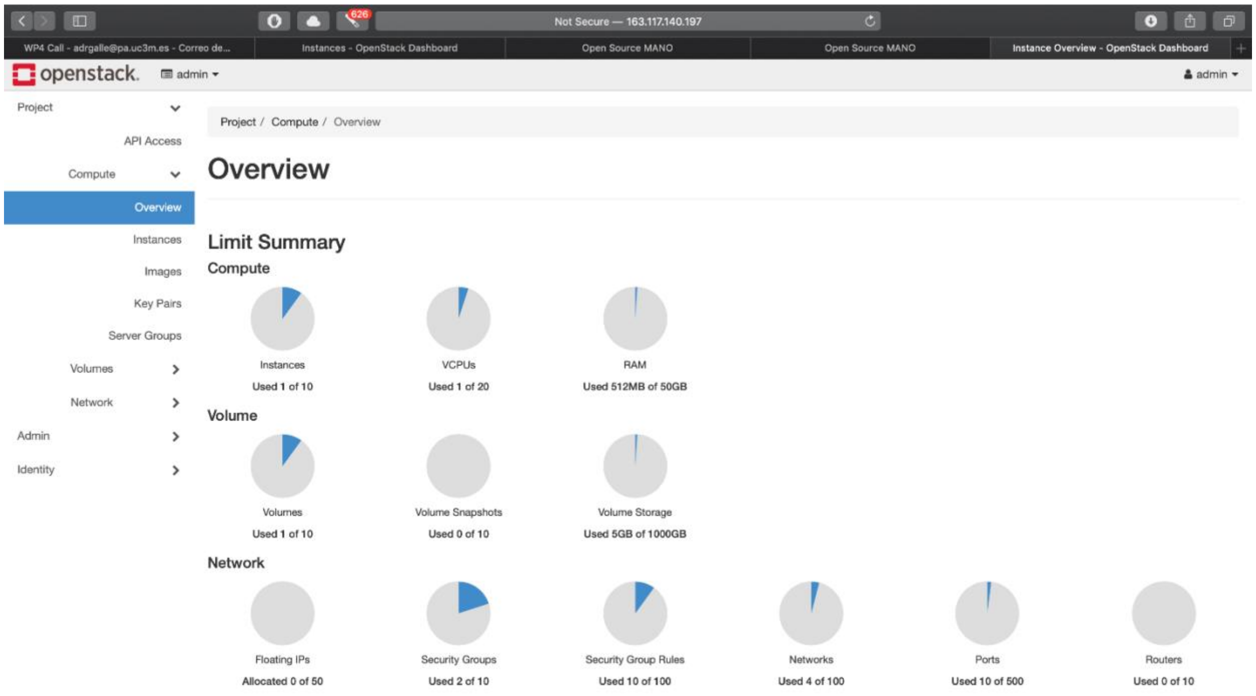


Project Name:	b515b685-f45e-4d5f-8232-046cbde33437
Created:	19-07-2019 11:20:00
Modified:	19-07-2019 11:20:00

Figure 3-9. OSM Rel. SIX portal overview.

3.2.2.3.3 DevStack Configuration

Same steps as in the previous have been followed to install DevStack, with minor changes in the *local.conf* file[50]. No problems were found during the installation and configuration process.


Not Secure — 163.117.140.197
WP4 Call - adrgalle@pa.uc3m.es - Correo de...
Instances - OpenStack Dashboard
Open Source MANO
Open Source MANO
Instance Overview - OpenStack Dashboard
openstack.
admin
admin
Project
API Access
Compute
Overview
Instances
Images
Key Pairs
Server Groups
Volumes
Network
Admin
Identity
Project / Compute / Overview

Overview

Limit Summary

Compute

Instances Used 1 of 10
VCPUs Used 1 of 20
RAM Used 512MB of 50GB

Volume

Volumes Used 1 of 10
Volume Snapshots Used 0 of 10
Volume Storage Used 5GB of 1000GB

Network

Floating IPs Allocated 0 of 50
Security Groups Used 2 of 10
Security Group Rules Used 10 of 100
Networks Used 4 of 100
Ports Used 10 of 500
Routers Used 0 of 10

Figure 3-10. Scenario 2 DevStack Overview.

50 See Annex *Scenario 2 local.conf DevStack File.*

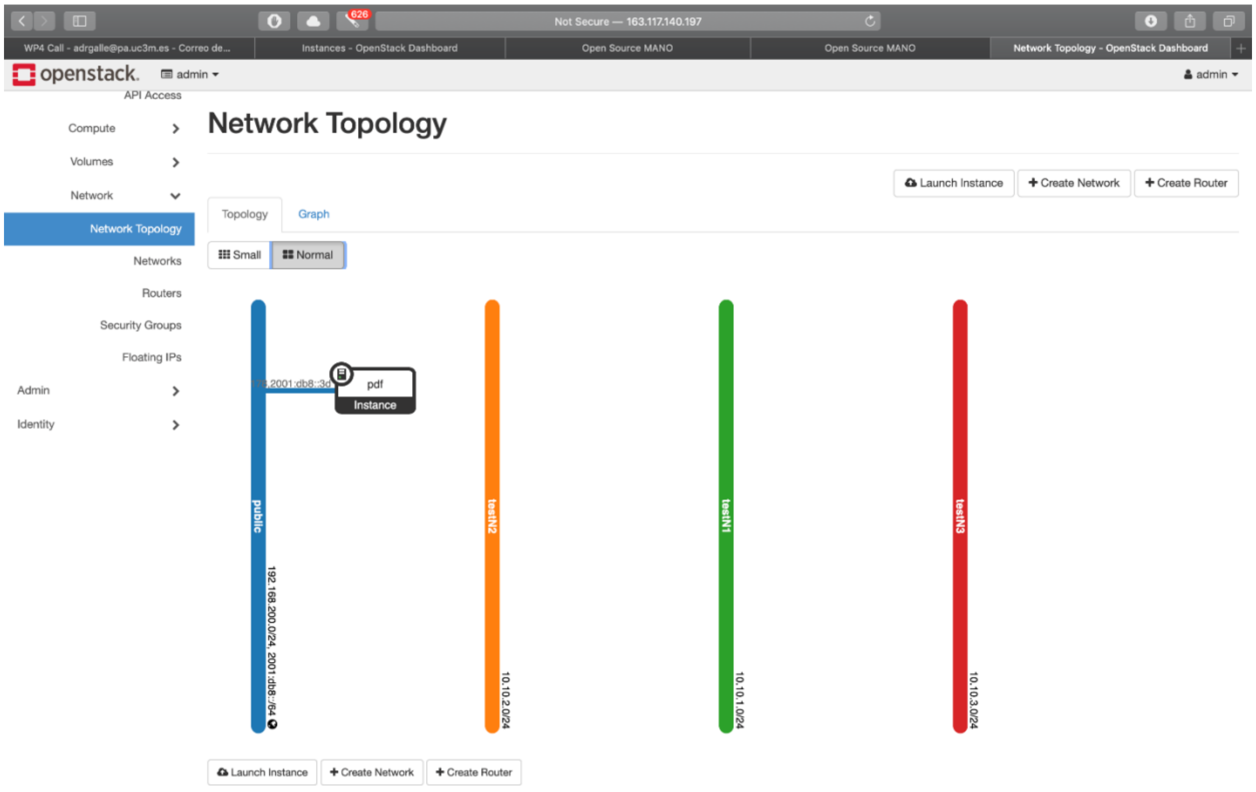
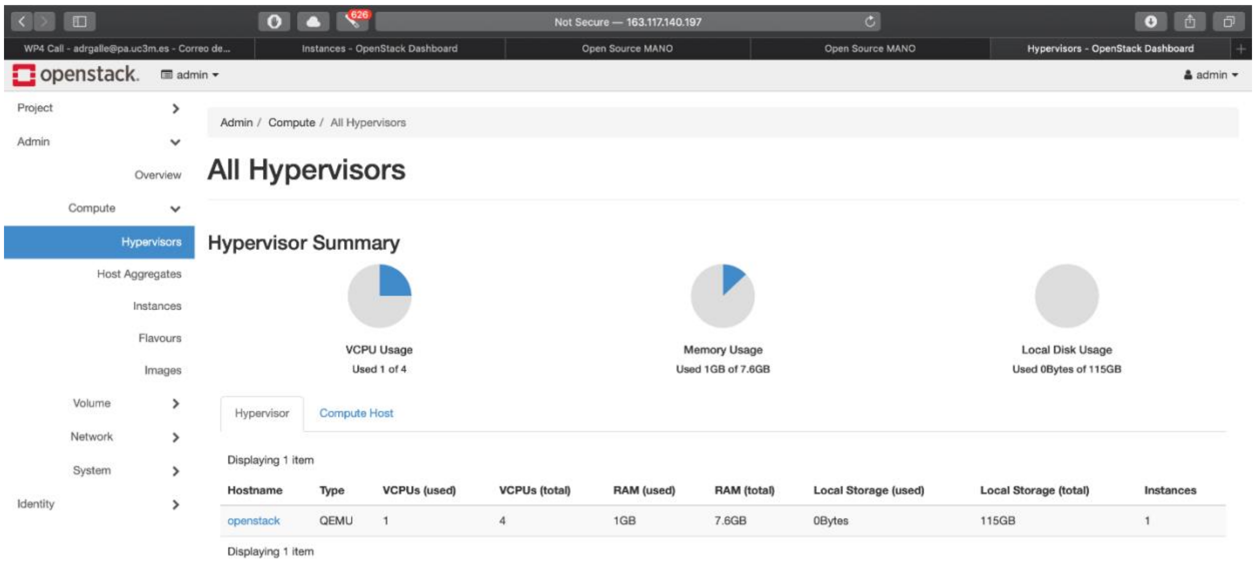


Figure 3-11. Scenario 2 DevStack Network Topology.



All Hypervisors

Hypervisor Summary

VCPU Usage
Used 1 of 4

Memory Usage
Used 1GB of 7.6GB

Local Disk Usage
Used 0Bytes of 115GB

Displaying 1 item

Hostname	Type	VCPUs (used)	VCPUs (total)	RAM (used)	RAM (total)	Local Storage (used)	Local Storage (total)	Instances
openstack	QEMU	1	4	1GB	7.6GB	0Bytes	115GB	1

Displaying 1 item

Figure 3-12. Scenario 2 DevStack Hypervisor Summary.

3.2.3 Production-Level Scenario 1

This scenario was intended to be the final step in the scenario designing phase of this Master Thesis but, due to several issues (explained below) this scenario ended as a half step between a completely functional production scenario and the previous-explained Testbed scenario. It is also important to remark that from this point onwards all the designs and infrastructure are shared with the 5G-VINNI Spanish facility and they will be used by them as their main 5G environment.

Every production-level scenario must have into account more details than user-oriented scenarios as their resource consumption is much greater, they may contain critical data and they must support greater computing efforts. Therefore, the first step to be taken was to select the location where the Main Server was going to be hosted, fortunately, UC3M has an agreement with IMDEA networks Institute[51] at Leganés (Madrid) and, consequently, the author of this project and the 5G-VINNI Spanish facility had access to the 5TONIC research lab[52] and Datacenter. 5TONIC is an open co-creation laboratory founded by Telefónica and IMDEA Networks Institute, that focus its developing efforts on 5G technologies ranging from H2020 projects' demonstrations to pilot 5G services and development of highly innovative technology through European's projects. In addition to the 5G-VINNI European project, it hosts other 5GPPP projects:

- **5G-EVE**
- **5GENESIS**
- **5G-Crosshaul**
- **5G-Ex**
- **5G-NORMA**
- **5G-TRANSFORMER**
- **5G-MONARCH**
- **5G-CORAL**
- **…**

Once all the paperwork was done and the 5G-VINNI had access to the 5TONIC's Datacenter, the next step was to select the proper server HW to be hosted at it. After collating the overall requirements of this scenario and the opinions and experience from other 5GPPP projects hosted at 5TONIC, the "Datacenter Remote Main Server" machine was bough[53]. Afterwards, the server was installed by 5TONIC's technicians and they gave the author VPN access to it and a dedicated network for the 5G-VINNI project inside the Datacenter: *10.5.10.0/16* (currently only the first 256 IPs are used, so it is used as a 255.255.255.0 network).

51 https://www.networks.imdea.org/

52 https://www.5tonic.org/

53 See Section *9.2.1 Hardware* for a detailed explanation of its characteristics.

Unlike the previous scenarios, this one is going to oversee the launching and hosting of complex experiments that will require larger resources, thereupon, a full OpenStack stack is required in order to have complete control over the platform and its behavior. Besides, it was decided that VirtualBox SW did not suite the requirements of this scenario at all and it was substituted by KVM as the VMs manager (this decision brought a lot of ease to the deployment but also some issues that will be addressed later). Below a high-level view of the current scenario is shown:

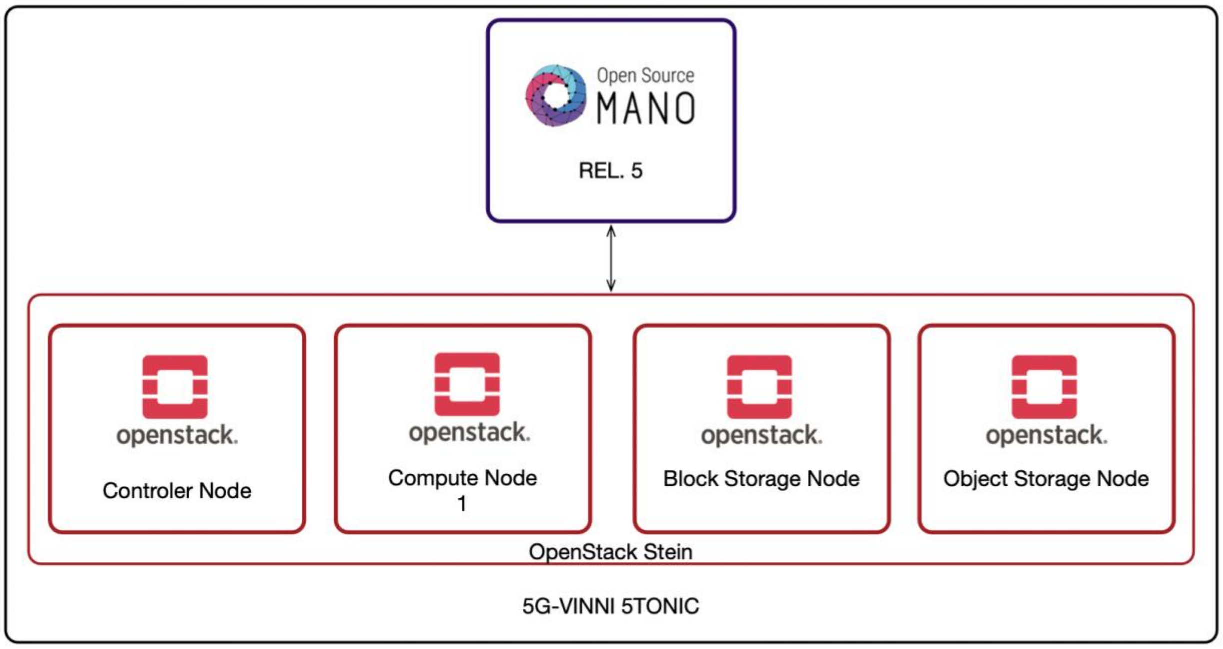


Figure 3-13. High-Level View of the production-level scenario 1.

Only the NFVI/NFVO and MANO parts are shown because, at the deployment time of this scenario, the HW needed to implement the eNB and the UE, including both servers and USRPs, was not available because of severe delays in the buying process.

Back to *Figure 3-13. High-Level View of the production-level scenario* 1., inside the "Datacenter Remote Main Server" machine several VMs are to be generated. In first place, one VM is required to host OSM and two to four VMs are required to host the OpenStack Stein (latest release) full stack. This variability in the latest number of VMs is due to the freedom that OpenStack Architecture gives to the developer at the time of deploying its full stack, block and Object storage nodes are optional and can be integrated directly inside the controller node. This flexibility derived in the generation of two possible designs:

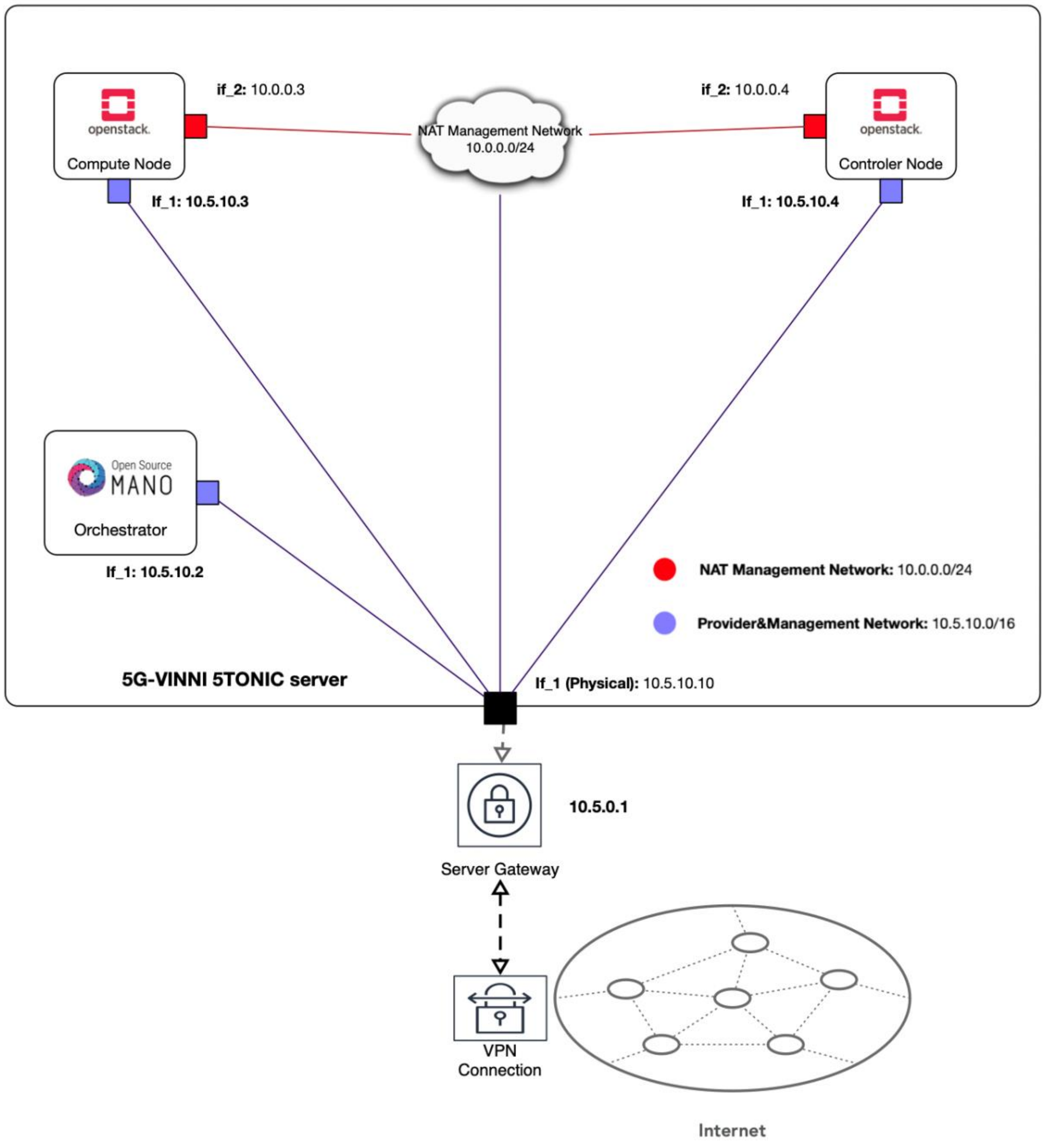


Figure 3-14. Production-Level Scenario 1 Option 1.

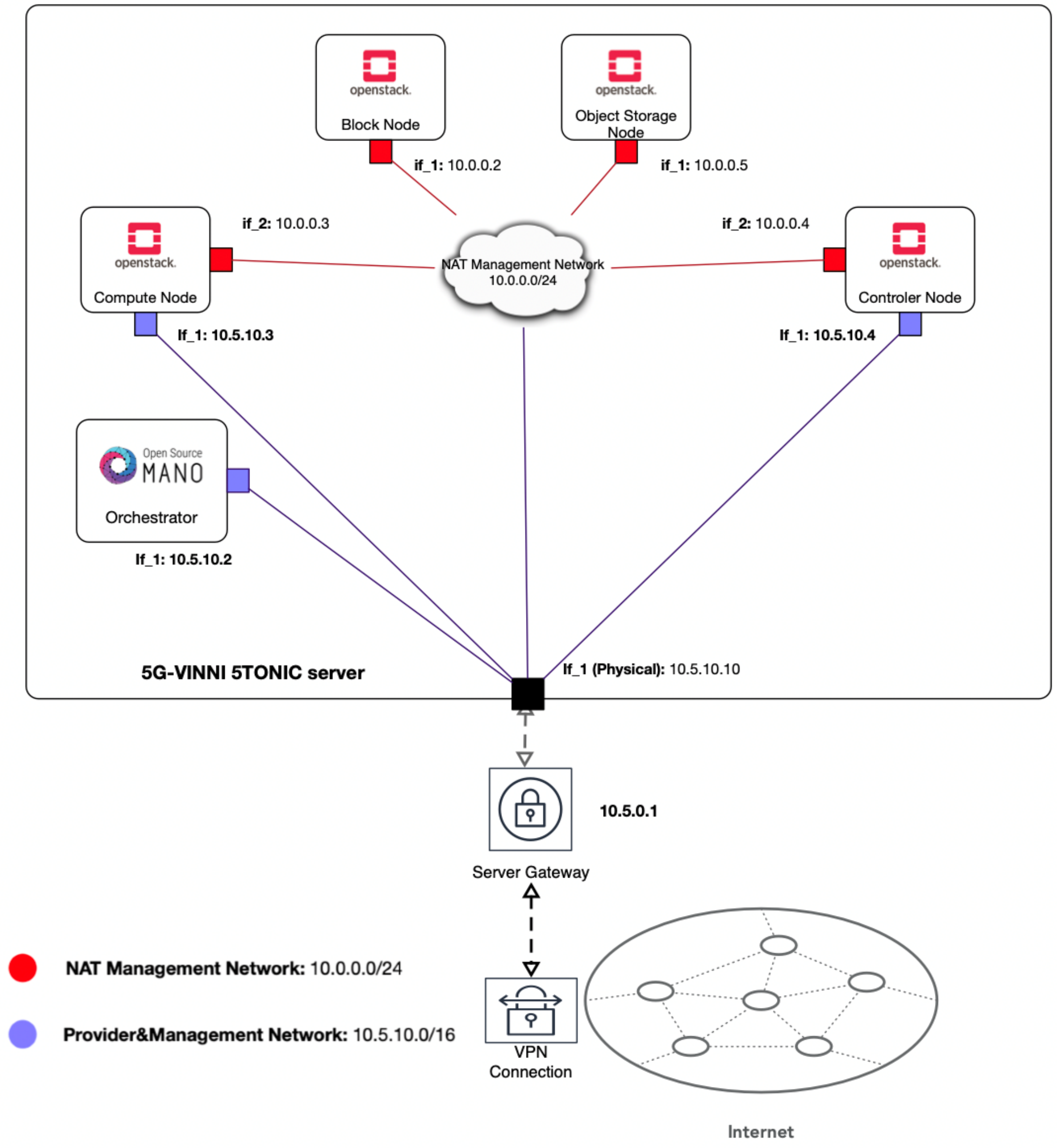


Figure 3-15. Production-Level Scenario 1 Option 2.

The main advantages and disadvantages between these two options are listed below:

	Advantages	Disadvantages
Option 1	• Simpler design. • Easy to maintain and manage as less VMs are involved. • Centralized functionality.	• Less control over the functionality of each node. • Controller node performance will suffer as it hosts the functionality of the storage nodes. • Complex configuration on the Controller node.
Option 2	• VMs functionality is clearly defined. • Finding issues is easier as their origin is clearer. • Distributed Functionality. • More flexible and scalable.	• Complex design. • More VMs are to be managed and maintained. • More resources are needed.

Table 3-9. Scenario Design Options Comparison.

As a fine-grained control is required over the resulting infrastructure and it is expected to be highly scalable, Option 2 was selected as the final design, resulting in a fully virtualized NFVI/NFVO-MANO scenario composed by five VMs:

- **OSM VM:** As in previous scenarios, in charge of the MANO functionalities.
- **Controller VM:** In charge of managing data exchanges between the other OpenStack Components. Runs the following services:
    - Identity
    - Image
    - Placement
    - Dashboard
    - Management portions of compute and networking services
    - Networking Agents
- **Compute Node VM:** Runs the hypervisor part of the environment as well as a networking service agent that connects the generated VMs to the available virtual networks. More than one compute node may be deployed, for this scenario one was enough.
- **Block Storage Node VM:** Optional node, it contains the disks that the Block Storage and the Shared Filesystem services provision to instances.
- **Object Storage Node VM:** Optional node, contains the disks that the Object Storage Service uses for storing accounts, container and objects.

3.2.3.1 HW Requirements

To obtain the best possible performance on each of the virtualized nodes having into account the total amount of resources available in the "Datacenter Remote Main Server" machine and without collapsing the host system, a study was carried out. To have a clear view of the results of this study, a table where each of the VM HW requirements is detailed and compared with the total amount of available resources, is shown:

Machine	CPUs	RAM	Disk	Network Interfaces
Physical Server	**20**	**32 GB**	**12 TB**	**7**
OSM	2	6 GB	100 GB	1
Controller	2	4 GB	100 GB	2
Compute	10	12 GB	2 TB	2
Object Storage	2	4 GB	1 TB (Disk1: 100 GB, Disk2: 900 GB)	1
Block Storage	2	4 GB	1TB (Disk1: 100 GB, Disk2: 450 GB, Disk3: 450 GB)	1
Total	**18**	**30 GB**	**4.2 TB**	**7 (virtualized)**
Remaining physical server resources	2	2 GB	7.8 TB	6[54]

Table 3-10. HW requirements Production-level scenario 1.

It may be thought that there is a lack of RAM in the developed scenario as the compute node, in charge of generating new VMs, has access to 12GB but it must be reminded that more compute nodes can be added to the OpenStack infrastructure increasing the overall amount of resources available for the creation of VMs and with the plus value of not needing to change any configuration.

54 One of the physical interfaces is used as a bridged interface for KVM.

3.2.3.2 SW Requirements

In this case, KVM has been selected as the virtualization manager for the physical server, OpenStack latest release "Stein" as the cloud solution and OSM Rel. FIVE has been maintained as the current orchestrator. Into the bargain, each of these platforms have certain requirements which are directly related with OS hosted by the respective physical or virtual machines.

MACHINE	REQUIRED OS
Physical Server	Ubuntu 18.04.2 LTS Bionic
OSM	Ubuntu 16.04 LTS Xenial
Controller	Ubuntu 18.04.2 LTS Bionic
Compute	Ubuntu 18.04.2 LTS Bionic
Block Storage	Ubuntu 18.04.2 LTS Bionic
Object Storage	Ubuntu 18.04.2 LTS Bionic

Table 3-11. OS requirements production-level scenario 1.

3.2.3.3 Configuration

3.2.3.3.1 Network Configuration

In this scenario, the infrastructure is part of a Datacenter network so the design of its architecture must be done in a careful way in order to avoid problems with other projects and to avoid problems to other projects. A couple of requirements and goals were considered:

1. The range of IPs from the Datacenter network is a *"/16"* but, at this point, only a "/24" could be used leaving a usable range of 10.5.10.0/24 but with a GW located in 10.5.0.1.
2. Isolation between MANO and OpenStack data flows is to be accomplished.
3. One virtualized network is desired.

As a result, aside from the Datacenter Network, a bridge interface was created on the physical server in order to be used by KVM's VMs to create virtual interfaces connected to that network. The physical server OS, Ubuntu 18.04.2, uses Netplan as the network manager, in order to configure a bridge interface, the file *"/etc/netplan/50-cloud-init.yaml"* was edited as shown in *Annex 10.2 Physical server Netplan config scenario 3.* Besides, as this bridge is going to be used by KVM, an extra step must be taken, an XML file descripting the bridge has to be created (see *Annex 10.3 Physical server libvirt bridge br0 definition)* and added to the Libvirt networks using the following commands:

```
# virsh net-define br0.xml
# virsh net-start br0
# virsh net-autostart br0
```

Text Box 3-11. Adding br0 to libvirt.

These commands will add this new network to the KVM options, enabling it and making it to start automatically every time the system reboots.

Additionally, using part of KVM's features, a virtualized network was created to transport OpenStack data flows. To fulfill this task, the ideal KVM network configuration was a NAT network, this type of network allows the VMs connected to it to send traffic between them and, besides, it adds an Internet GW (at first this was not a problem but when configuring the compute and controller node's VMs, it has to be considered if no routing issues are desired, as there are two available GWs). To create this network, with a DHCP agent, another XML was needed (see *Annex 10.4 Physical server libvirt NAT network definition)* and Libvirt was advised of this new network with the following commands:

```
# virsh net-define nat-net.xml
# virsh net-start NAT_mgmt
# virsh net-autostart NAT_mgmt
```

Text Box 3-12. Adding NAT-mgmt to libvirt.

After all this configuration was done, the following networks were usable by KVM:

NETWORK	IP RANGE	GATEWAY
BR0 (PROVIDER)	10.5.10.0/24	10.5.0.1
NAT_MGMT (DATA)	10.0.0.0/24	10.0.0.1

Table 3-12. KVM available networks.

Finally, as it has been mentioned, to avoid major routing issues/complex network configurations at Controller and Compute nodes' VMs, it was decided to leave them with one single GW. On a first approach, they were configured to use the NAT network's GW but, with this configuration, these machines couldn't be accessed from the VPN due to the "nating" process carried by KVM. Thereupon, they were finally configured with the provider network's GW. *Table 3-13. IP distribution scenario* 3. Shows the resulting IP distribution for this scenario:

Table 3-13. IP distribution scenario 3.

MACHINE	INTERFACE	IP	GW
PHYSICAL SERVER	Eno1 (physical)	10.5.10.10	10.5.0.1
OSM	Ens3 (bridged)	10.5.10.2	10.5.0.1
CONTROLLER	Ens3 (bridged)	10.5.10.4	10.5.0.1
	Ens4 (NAT network)	10.0.0.4	-
COMPUTE	Ens3 (bridged)	10.5.10.3	10.5.0.1

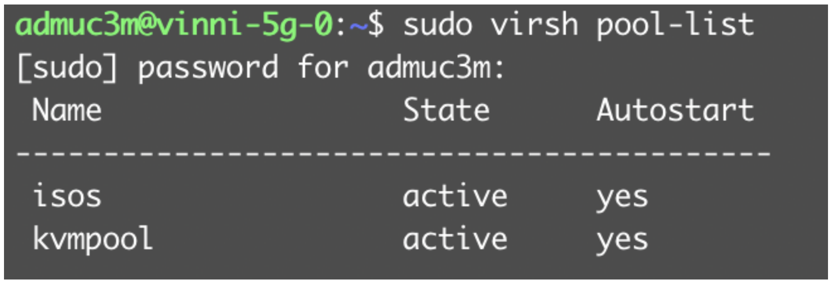
	Ens4 (NAT network)	10.0.0.3	-
BLOCK STORAGE	Ens3 (NAT network)	10.0.0.2	10.0.0.1
OBJECT STORAGE	Ens3 (NAT network)	10.0.0.5	10.0.0.1

3.2.3.3.2 Virtual Machines Configuration

In this scenario KVM has been used as the hypervisor/VIM, this means that a storage pool must be configured to storage VMs' disks. To accomplish this objective the following commands were executed:

```
# virsh pool-define-as kvmpool dir /var/lib/libvirt/images
# virsh pool-autostart kvmpool
# virsh pool-start kvmpool
```

Text Box 3-13. KVM storage pool definition.



```
admuc3m@vinni-5g-0:~$ sudo virsh pool-list
[sudo] password for admuc3m:
 Name                 State      Autostart
-------------------------------------------
 isos                 active     yes
 kvmpool              active     yes
```

Figure 3-16. Storage Pool list (physical server scenario3).

Subsequently, every single VM was generated using KVM tools. In the next text boxes, the command/s used for the creation of each VM are shown:

```
$ sudo virt-install -name OSM -memory 6144 -vcpus 2 -cdrom isos/ubuntu-
16.04.6-server-amd64.iso -disk pool=kvmpool,size=100 -boot
cdrom,fd,hd,network -network bridge=br0 -virt-type kvm -graphics
vnc,port=5901,listen=0.0.0.0 -os-type linux -os-variant ubuntu16.04
```

Text Box 3-14. OSM VM creation.

Using the above command a VM named "OSM" will be generated using an Ubuntu 16.04.6 ISO with 6GB of RAM, 2 vCPUS, 100GB disk and one network interface connected to the bridge physical interface (to the provider/datacenter network) and its image will be transmitted by a VNC server which is served on the physical server IP on port 5900 (10.5.10.10:5900), to allows the initial configuration of the VM.

```
$ sudo virt-install --name openstack_ctrl --memory 4096 --vcpus 2 --cdrom
isos/ubuntu-18.04.2-server-amd64.iso --disk pool=kvmpool,size=100 --boot
cdrom,fd,hd,network --network bridge=br0 --network=NAT_mgmt --virt-type kvm
--graphics vnc,port=5901,listen=0.0.0.0 --os-type linux --os-variant
ubuntu18.04
```

Text Box 3-15. OpenStack Controller VM creation.

```
$ sudo virt-install --name openstack_cmp1 –memory 12288 --vcpus 10 --cdrom
isos/ubuntu-18.04.2-server-amd64.iso --disk pool=kvmpool,size=2000 --boot
cdrom,fd,hd,network --network bridge=br0 --network=NAT_mgmt --virt-type kvm
--graphics vnc,port=5902,listen=0.0.0.0 --os-type linux --os-variant
ubuntu18.04
```

Text Box 3-16. OpenStack Compute Node One VM creation.

```
$ sudo virt-install --name openstack_block --memory 4096 --vcpus 2 --cdrom
isos/ubuntu-18.04.2-server-amd64.iso --disk pool=kvmpool,size=100 --boot
cdrom,fd,hd,network --network=NAT_mgmt --virt-type kvm --graphics
vnc,port=5903,listen=0.0.0.0 --os-type linux --os-variant ubuntu18.04
```

Text Box 3-17. OpenStack Block Storage Node VM creation.

```
$ sudo time qemu-img create -f qcow2
/var/lib/libvirt/images/openstack_block-disk2.qcow2 100G -o
prealocation=full
$ sudo chown libvirt-qemu:kvm /var/lib/libvirt/images/openstack_block-
disk2.qcow2
$ sudo virsh attach-disk --domain openstack_block
/var/lib/libvirt/openstack_block-disk2.qcow2 --target vdb --persistent --
config --live
```

Text Box 3-18. Openstack Block Storage Node disk attachment.

```
$ sudo virt-install --name openstack_object --memory 4096 --vcpus 2 --cdrom
isos/ubuntu-18.04.2-server-amd64.iso --disk pool=kvmpool,size=100 --boot
cdrom,fd,hd,network --network=NAT_mgmt --virt-type kvm --graphics
vnc,port=5904,listen=0.0.0.0 --os-type linux --os-variant ubuntu18.04
```

Text Box 3-19. OpenStack Object Storage Node creation.

```
$ sudo time qemu-img create -f qcow2
/var/lib/libvirt/images/openstack_object-disk[2,3].qcow2 450G -o
prealocation=full

$ sudo chown libvirt-qemu:kvm /var/lib/libvirt/images/openstack_object-
disk[2,3].qcow2

$ sudo virsh attach-disk --domain openstack_object
/var/lib/libvirt/openstack_object-disk2.qcow2 --target vdb --persistent --
config –live

$ sudo virsh attach-disk --domain openstack_object
/var/lib/libvirt/openstack_object-disk3.qcow2 --target vdc --persistent --
config --live
```

Text Box 3-20. OpenStack Object Storage Node disk attachment.

As it can be seen, the creation of the VMs is straightforward, the only difference is that, for the Object and Block Storage VMs some more disks were needed in order to fulfill their respective OpenStack modules (cinder and Swift) requirements. These disks are created with full pre-allocation, although it is not the best manner of optimizing space in the physical server, because if not KVM understands that the disk has a size under 64kB, and it does not allow to make any kind of partitions. Moreover, every disk is formatted in qcow2 format to enable KVM's snapshot support and they are attached with the arguments "--persitent" and "config --live" to be able to Plug&Play these disks and to maintain them attached to the VM even if it reboots.

Further, the initial configuration of each VM was done using the program "*VNC Viewer*" in the Laptop machine. This configuration consisted on following the steps of the installation agents on each on the ISO images, updating all the packages and configuring the network interfaces as explained before. When the installation was over, a snapshot was taken in every VM to avoid re-configuring it from scratch if a major problem arises.

```
$ sudo virsh snapshot-create-as --domain {VM-NAME} --name "{SNAPSHOT-NAME}"
```

Text Box 3-21. Snapshot creation command (KVM).

3.2.3.3.3 OSM configuration

OSM configuration was done exactly as described in the previous scenario (testbed scenario) for the Rel. FIVE. No issues were encountered.

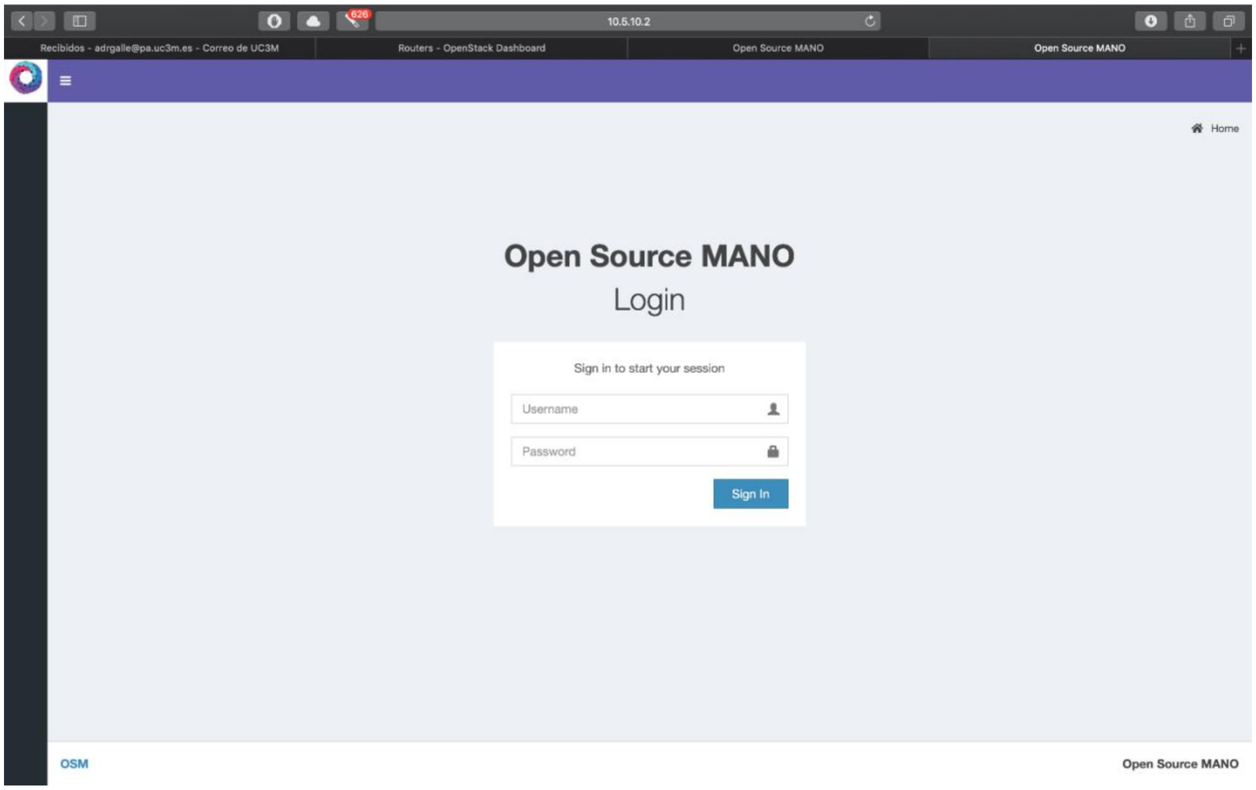


Figure 3-17. OSM Rel. FIVE Dashboard.

3.2.3.3.4 OpenStack Configuration

This was, in parallel with the network design, the most complex task, as it involves configuring several modules, SW and files in different VMs sensitive to API version differences and syntax errors. In order to avoid repeating the full configuration/installation process, since various installation guides were followed step by step, a reference to these configuration guides is going to be shown for each of the installed modules and, if any change has been made it will be explained:

Module	Guide	Changes
Password Generation & Security	https://docs.openstack.org/install-guide/environment-security.html	None
Host Networking	https://docs.openstack.org/install-guide/environment-networking.html	None

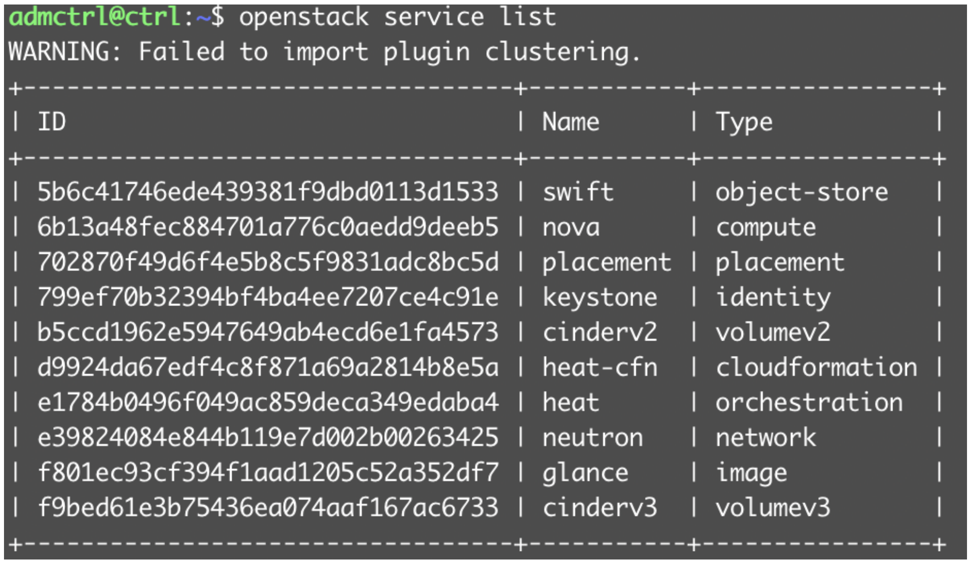
Network Time Protocol	https://docs.openstack.org/install-guide/environment-ntp.html	None
OpenStack Package Repositories	https://docs.openstack.org/install-guide/environment-packages.html	None
SQL Database	https://docs.openstack.org/install-guide/environment-sql-database.html	None
Message Queue	https://docs.openstack.org/install-guide/environment-messaging.html	None
Memcached	https://docs.openstack.org/install-guide/environment-memcached.html	None
Etcd	https://docs.openstack.org/install-guide/environment-etcd.html	None
KeyStone	https://docs.openstack.org/keystone/stein/install/	None
Glance	https://docs.openstack.org/glance/stein/install/	None
Placement	https://docs.openstack.org/placement/stein/install/	None
Nova	https://docs.openstack.org/nova/stein/install/	Only one compute node was configured. Qemu was configured as VIM as the second level of virtualization didn't allow to use KVM.
Neutron	https://docs.openstack.org/neutron/stein/install/	Networking option 2 was selected (provider + self-service networks)
Horizon	https://docs.openstack.org/horizon/stein/install/	Layer-3 networking services in file /etc/openstack-dashboard/local_settings.py at the controller node, were enabled.
Swift	https://docs.openstack.org/swift/stein/install/	Used port 5000 instead of port 35357 for the auth_url at the filter:authtoken section in file /etc/swift/proxy-server.conf at the controller node.
Cinder	https://docs.openstack.org/cinder/stein/install/index.html	Backup service enabled.
Heat	https://docs.openstack.org/heat/stein/install/	None

Table 3-14. Openstack Modules configuration.

The entire installation process lasted four days (around 20h), several first-time-installation errors were encountered. The worst of all was that the nova-compute module did not connect with the nova-api module and, after reading the logs several times and trying a full re-installation of the modules it turned out that it was a problem of the API versions (the compute node API version was older than the one installed in the controller node, the origin of this problem was that in the compute node the package update was not carried out). Once the initial OpenStack Stein installation and configuration was done, and all the modules and features were validated, more user-oriented configuration processes were left:

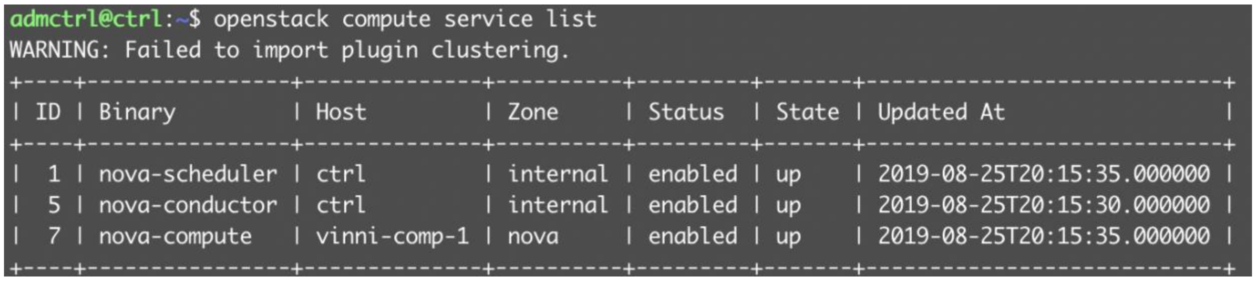
- **Creation of users, projects and domains:**
  - **Domain:** 5G-VINNI-5TONIC
  - **Project:** demo
  - **User:** 5gvinnidev
- **Network Creation:** A provider and a self-service network were created, the first one is a shared network for all the projects (it consists of the IP range of the datacenter network) and a self-service network with a GW router to the public network (Internet access) was generated for the demo project with the IP range: 192.168.20.0/24. See
- **Flavor Creation:** Several flavors were created in order to allow a wide range of flexibility for the VM resources selection.
- **Images:** Several qcow2 images were added to have a complete image catalog.

Bellow, some images are shown in order to give a proof of concept of the correct installation of the OpenStack Stein full stack for this scenario:



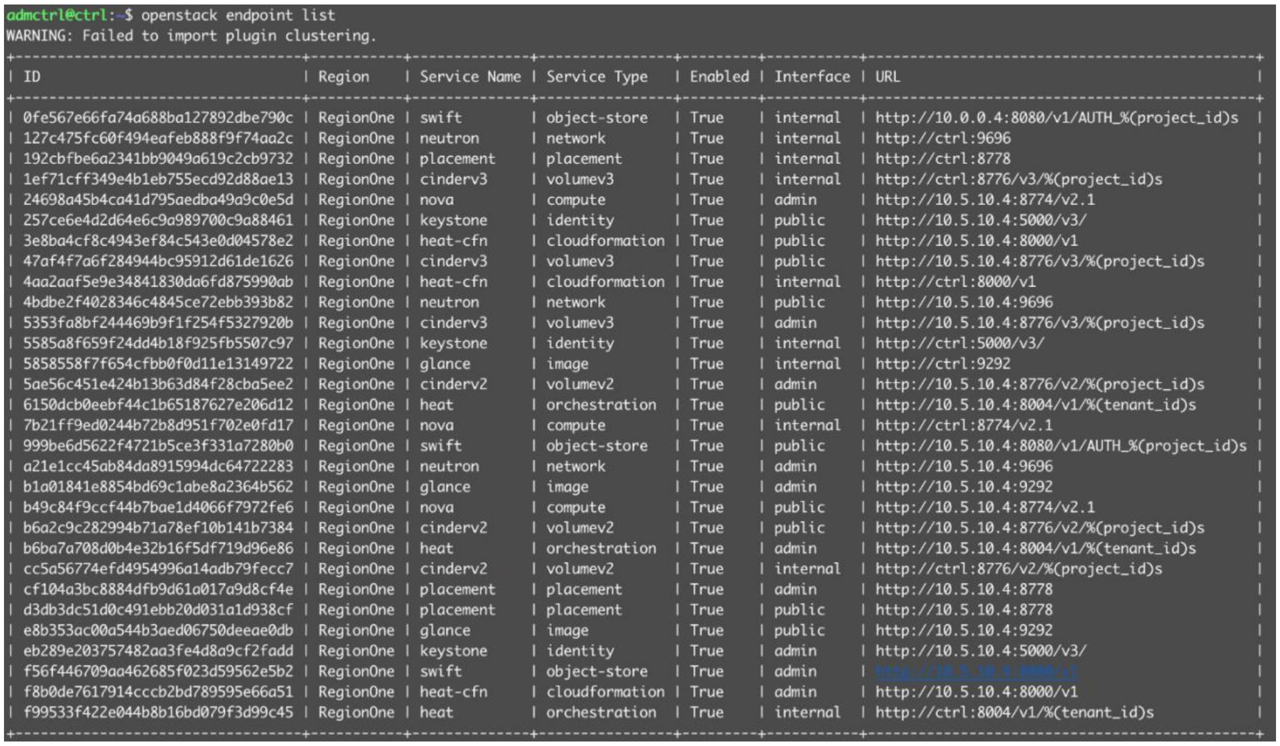
```
admctrl@ctrl:~$ openstack service list
WARNING: Failed to import plugin clustering.
+----------------------------------+-----------+----------------+
| ID                               | Name      | Type           |
+----------------------------------+-----------+----------------+
| 5b6c41746ede439381f9dbd0113d1533 | swift     | object-store   |
| 6b13a48fec884701a776c0aedd9deeb5 | nova      | compute        |
| 702870f49d6f4e5b8c5f9831adc8bc5d | placement | placement      |
| 799ef70b32394bf4ba4ee7207ce4c91e | keystone  | identity       |
| b5ccd1962e5947649ab4ecd6e1fa4573 | cinderv2  | volumev2       |
| d9924da67edf4c8f871a69a2814b8e5a | heat-cfn  | cloudformation |
| e1784b0496f049ac859deca349edaba4 | heat      | orchestration  |
| e39824084e844b119e7d002b00263425 | neutron   | network        |
| f801ec93cf394f1aad1205c52a352df7 | glance    | image          |
| f9bed61e3b75436ea074aaf167ac6733 | cinderv3  | volumev3       |
+----------------------------------+-----------+----------------+
```

Figure 3-18. OpenStack Stein scenario3 service list.



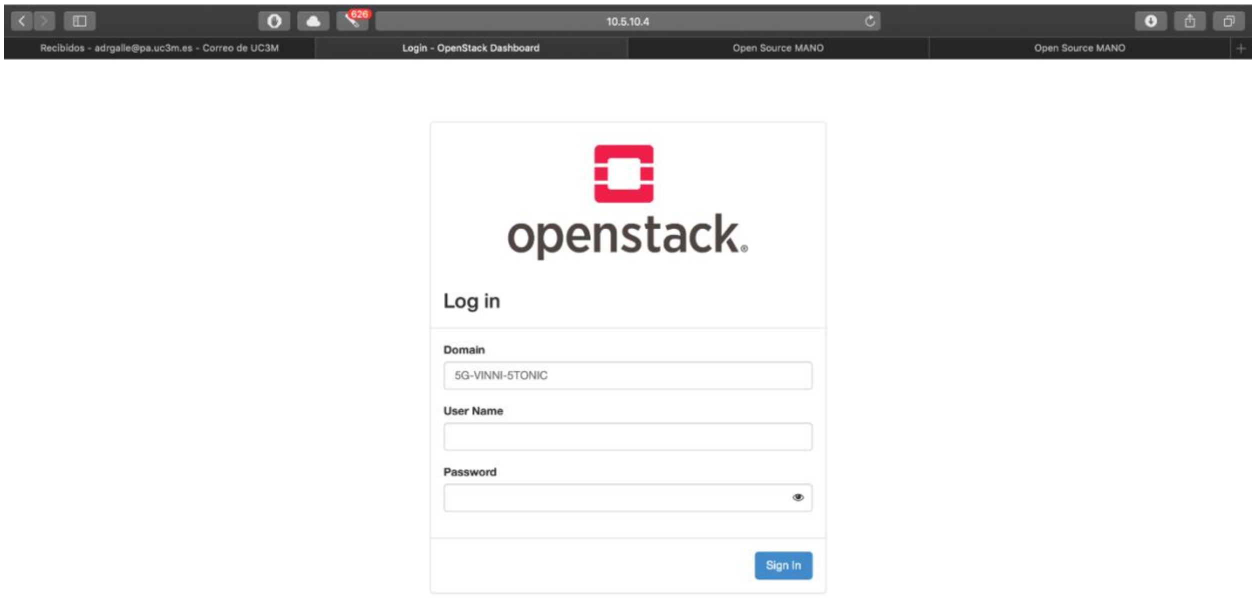
```
admctrl@ctrl:~$ openstack compute service list
WARNING: Failed to import plugin clustering.
+----+----------------+--------------+----------+---------+-------+----------------------------+
| ID | Binary         | Host         | Zone     | Status  | State | Updated At                 |
+----+----------------+--------------+----------+---------+-------+----------------------------+
|  1 | nova-scheduler | ctrl         | internal | enabled | up    | 2019-08-25T20:15:35.000000 |
|  5 | nova-conductor | ctrl         | internal | enabled | up    | 2019-08-25T20:15:30.000000 |
|  7 | nova-compute   | vinni-comp-1 | nova     | enabled | up    | 2019-08-25T20:15:35.000000 |
+----+----------------+--------------+----------+---------+-------+----------------------------+
```

Figure 3-19. OpenStack Stein scenario3 compute service (nova) list.



```
admctrl@ctrl:~$ openstack endpoint list
WARNING: Failed to import plugin clustering.
+----------------------------------+-----------+--------------+----------------+---------+-----------+----------------------------------------------+
| ID                               | Region    | Service Name | Service Type   | Enabled | Interface | URL                                          |
+----------------------------------+-----------+--------------+----------------+---------+-----------+----------------------------------------------+
| 0fe567e66fa74a688ba127892dbe790c | RegionOne | swift        | object-store   | True    | internal  | http://10.0.0.4:8080/v1/AUTH_%(project_id)s  |
| 127c475fc60f494eafeb888f9f74aa2c | RegionOne | neutron      | network        | True    | internal  | http://ctrl:9696                             |
| 192cbfbe6a2341bb9049a619c2cb9732 | RegionOne | placement    | placement      | True    | internal  | http://ctrl:8778                             |
| 1ef71cff349e4b1eb755ecd92d88ae13 | RegionOne | cinderv3     | volumev3       | True    | internal  | http://ctrl:8776/v3/%(project_id)s           |
| 24698a45b4ca41d795aedba49a9c0e5d | RegionOne | nova         | compute        | True    | admin     | http://10.5.10.4:8774/v2.1                   |
| 257ce6e4d2d64e6c9a989700c9a88461 | RegionOne | keystone     | identity       | True    | public    | http://10.5.10.4:5000/v3/                    |
| 3e8ba4cf8c4943ef84c543e0d04578e2 | RegionOne | heat-cfn     | cloudformation | True    | public    | http://10.5.10.4:8000/v1                     |
| 47af4f7a6f284944bc95912d61de1626 | RegionOne | cinderv3     | volumev3       | True    | public    | http://10.5.10.4:8776/v3/%(project_id)s      |
| 4aa2aaf5e9e34841830da6fd875990ab | RegionOne | heat-cfn     | cloudformation | True    | internal  | http://ctrl:8000/v1                          |
| 4bdbe2f4028346c4845ce72ebb393b82 | RegionOne | neutron      | network        | True    | public    | http://10.5.10.4:9696                        |
| 5353fa8bf244469b9f1f254f5327920b | RegionOne | cinderv3     | volumev3       | True    | admin     | http://10.5.10.4:8776/v3/%(project_id)s      |
| 5585a8f659f24dd4b18f925fb5507c97 | RegionOne | keystone     | identity       | True    | internal  | http://ctrl:5000/v3/                         |
| 5858558f7f654cfbb0f0d11e13149722 | RegionOne | glance       | image          | True    | internal  | http://ctrl:9292                             |
| 5ae56c451e424b13b63d84f28cba5ee2 | RegionOne | cinderv2     | volumev2       | True    | admin     | http://10.5.10.4:8776/v2/%(project_id)s      |
| 6150dcb0eebf44c1b65187627e206d12 | RegionOne | heat         | orchestration  | True    | public    | http://10.5.10.4:8004/v1/%(tenant_id)s       |
| 7b21ff9ed0244b72b8d951f702e0fd17 | RegionOne | nova         | compute        | True    | internal  | http://ctrl:8774/v2.1                        |
| 999be6d5622f4721b5ce3f331a7280b0 | RegionOne | swift        | object-store   | True    | public    | http://10.5.10.4:8080/v1/AUTH_%(project_id)s |
| a21e1cc45ab84da8915994dc64722283 | RegionOne | neutron      | network        | True    | admin     | http://10.5.10.4:9696                        |
| b1a01841e8854bd69c1abe8a2364b562 | RegionOne | glance       | image          | True    | admin     | http://10.5.10.4:9292                        |
| b49c84f9ccf44b7bae1d4066f7972fe6 | RegionOne | nova         | compute        | True    | public    | http://10.5.10.4:8774/v2.1                   |
| b6a2c9c282994b71a78ef10b141b7384 | RegionOne | cinderv2     | volumev2       | True    | public    | http://10.5.10.4:8776/v2/%(project_id)s      |
| b6ba7a708d0b4e32b16f5df719d96e86 | RegionOne | heat         | orchestration  | True    | admin     | http://10.5.10.4:8004/v1/%(tenant_id)s       |
| cc5a56774efd4954996a14adb79fecc7 | RegionOne | cinderv2     | volumev2       | True    | internal  | http://ctrl:8776/v2/%(project_id)s           |
| cf104a3bc8884dfb9d61a017a9d8cf4e | RegionOne | placement    | placement      | True    | admin     | http://10.5.10.4:8778                        |
| d3db3dc51d0c491ebb20d031a1d938cf | RegionOne | placement    | placement      | True    | public    | http://10.5.10.4:8778                        |
| e8b353ac00a544b3aed06750deeae0db | RegionOne | glance       | image          | True    | public    | http://10.5.10.4:9292                        |
| eb289e203757482aa3fe4d8a9cf2fadd | RegionOne | keystone     | identity       | True    | admin     | http://10.5.10.4:5000/v3/                    |
| f56f446709aa462685f023d59562e5b2 | RegionOne | swift        | object-store   | True    | admin     | http://10.5.10.4:8080/v1                     |
| f8b0de7617914cccb2bd789595e66a51 | RegionOne | heat-cfn     | cloudformation | True    | admin     | http://10.5.10.4:8000/v1                     |
| f99533f422e044b8b16bd079f3d99c45 | RegionOne | heat         | orchestration  | True    | internal  | http://ctrl:8004/v1/%(tenant_id)s            |
+----------------------------------+-----------+--------------+----------------+---------+-----------+----------------------------------------------+
```

Figure 3-20. OpenStack Stein scenario3 full endpoint list.


10.5.10.4
Recibidos - adrgalle@pa.uc3m.es - Correo de UC3M
Login - OpenStack Dashboard
Open Source MANO
Open Source MANO
openstack.
Sign In

Log in

Domain
5G-VINNI-5TONIC

User Name

Password

Figure 3-21. Openstack Stein scenario3 Dashboard.

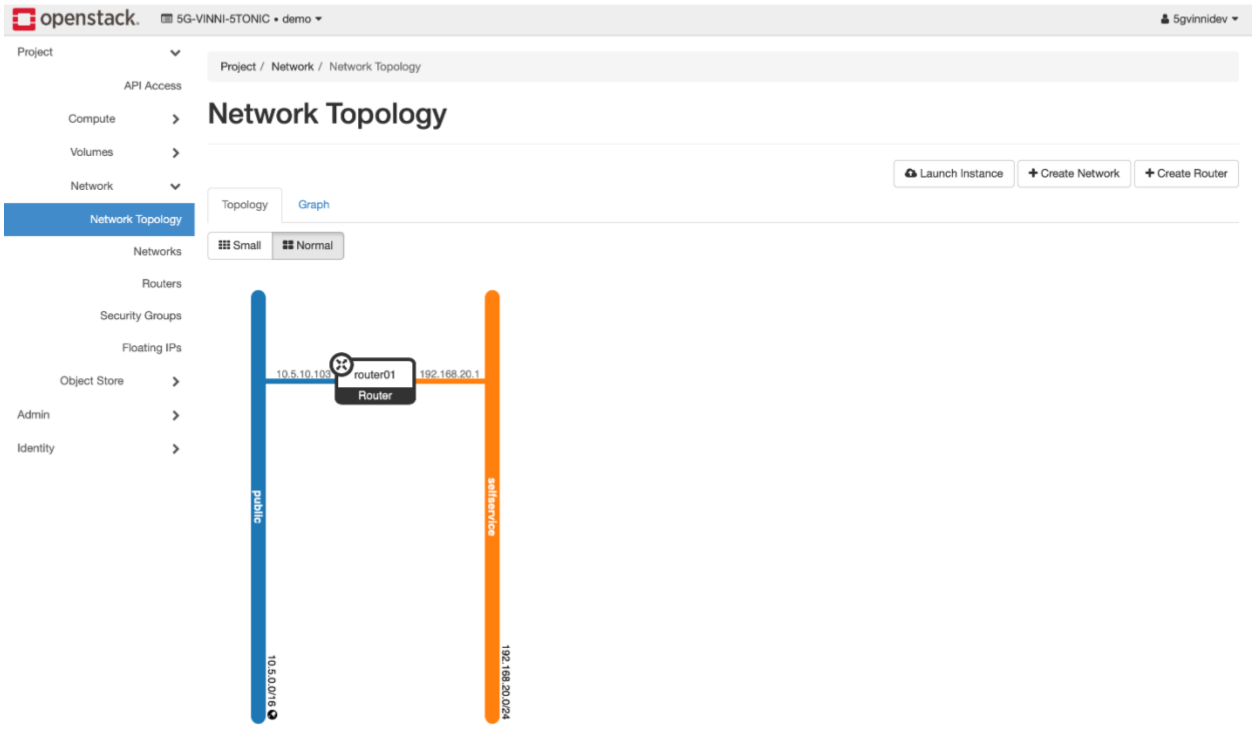


Figure 3-22. OpenStack Stein scenario3 initial network topology.

3.2.3.3.5 Radio Configuration

In this scenario no Radio HW was acquired for the NFVI deployment due to delay problems with the HW provider, therefore no installation/configuration could be done.

3.2.3.3.6 Issues

After the configuration of the whole NFVI/NFVO environment (OSM + OpenStack) some tests were deployed and the author encountered several major issues, most of them related with the fact of using nested virtualization layers. Here, a summary of the most relevant ones is shown:

1. ***"Resolf.conf"* issue:** When the *self-service network* is created on the controller node, after a couple of minutes or if a VM is instantiated before with an interface attached to this network, it was found that the controller node lost its capacity to ping to any domain name but it could ping to single IPs e.g. Controller node could not ping to www.google.com but it could ping to the domain's IP. If the VM and the network were erased, the controller node regained the ability to do pings to domain names. After several research, it was found that the neutron-server-api installed on the controller node, when creating this network modified the "resolf.conf" file, deleting the entrance that allowed the controller node to have DNS capabilities. Changing the configuration manually worked but, to avoid potential issues, a cronjob was created with a script that checked the content of this file and, if the "nameserver 8.8.8.8" was deleted, the script added it. The error was raised to the OpenStack community as no trace in any log was left when this action took place.
2. **Compute Node Virtualization:** KVM, on the physical server, was configured to accept nested virtualization in order to be able to launch the OpenStack VMs (first virtualization level) and to allow the compute node to create its own VMs (second virtualization layer) with HW acceleration. After some VMs were launched in the OpenStack environment (directly from OpenStack or from the Orchestrator) it was discovered that the instantiation process of the VMs took more time than expected and the overall performance of the VMs was poor. To check the origin of the problem the author checked HW acceleration support in both, the physical server and in the compute node[55], in the physical server the 44 cores supported HW acceleration but, in the compute node, this option was not supported (even though the physical server was configured to do so) and instead of using KVM as VIM, QEMU was used. No solution was found to this problem, as the physical server configuration was re-checked and was fine.
3. **Second virtualization level:** At first, the previous issue only affected the performance of the VMs, but they were fully usable. However, after some tests with different OS images the author came across with the fact that the Linux kernel implemented in some of these

[55] The following bash command was used: "$ egrep -c '(vmx|svm)' /proc/cpuinfo".

images generated a kernel-boot error that did not allow to instantiate the VMs. This error was found with the following images:

a. Ubuntu 18.04.2 (linux kernel 4.15.2)
b. Cisco Router Image
c. Arista Switch Image

A patch for this issue is implemented with the ubuntu-linux-kernel 5.2.10, so upgrading the kernel in the case of the ubuntu image solved the conflict. In the other two cases, no solution was found as no direct access was permitted to the kernel.

The last two issues were severely critical as the deployed infrastructure is going to be used by 3rd parties at some point in the 5G-VINNI project and, logically, we (the author and the other 5G-VINNI developers) could not say to these partners that they cannot use certain images in their experiments. Thereupon, a new scenario was required without these problems.

3.2.4 Production-Level Scenario 2

This final scenario was not expected but it had to be designed because of the nested-virtualization issues from its previous version scenario. The priority was to allow any potential experimenter, even the ones from inside the 5G-VINNI Spanish facility, to launch experiments without limiting their options choosing an OS image or without experimenting bad performance inside their instantiated VMs. Therefore, nested virtualization on the compute node had to be eliminated as that was the origin of the problem.

The solution came up with the form of a new physical server[56] which was especially adapted to perform a compute node functionality (multithread CPUs, DDR4 expandable RAM and SSD disks). Curiously, when this server arrived (from now on it will be referred as *"Datacenter Remote Compute Node")* 5TONIC's technicians were on holidays and, as the installation of the new HW had very high priority, the author did install the new *"Datacenter Remote Compute Node"* physically at the Datacenter.

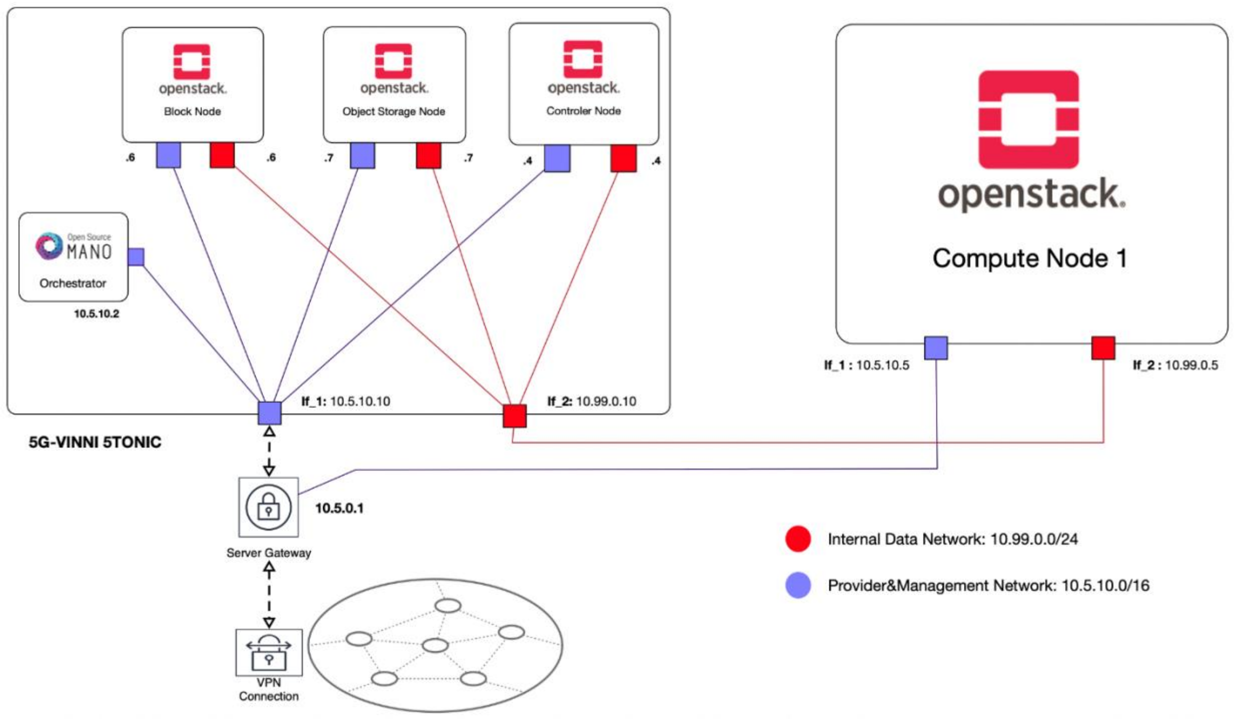


Figure 3-23. Final Scenario Network Architecture.

In the left part of the above figure, the *"Datacenter Remote Main Server"* can be seen hosting four VMs (one less than in the past scenario) and, in the right, the new physical machine is shown. As one of the OpenStack nodes was out of KVM's NAT network range a new physical network was

56 See Section 9.2.1 *Hardware* for a detailed explanation of its characteristics.

addressed. Furthermore, no radio schemas are shown because, once again, when the deployment of this device was done only the machines that will host the UE/eNB had arrived, the USRPs cards did not. These devices are located physically out of the 5TONIC's Datacenter and, therefore, a new optic fiber line had to be deployed from the server's Top of Rack (ToR) switch to a switch in the lab where they are located.

Apart from the fact that the compute node is no longer a VM, the rest of the VMs have the same functionalities as the ones explained in the previous scenario.

3.2.4.1 HW Requirements

With the incorporation of the new physical server as the compute node, more resources were freed from the *"Datacenter Remote Main Server"* and, thus, the VMs assigned resources could be adapted for an optimized performance.

Machine	CPUs	RAM	Disk	Network Interfaces
Physical Server	**20**	**32 GB**	**12 TB**	**7**
OSM	2	8 GB	100 GB	1
Controller	2	8 GB	100 GB	2
Object Storage	2	6 GB	1 TB (Disk1: 100 GB, Disk2: 900 GB)	2
Block Storage	2	4 GB	1TB (Disk1: 100 GB, Disk2: 450 GB, Disk3: 450 GB)	2
Total	**18**	**26 GB**	**2.2 TB**	**7 (virtualized)**
Remaining physical server resources	2	6 GB	9.8 TB	5[57]

Table 3-15. HW requirements final scenario.

The new compute node has the following specifications:

Machine	CPUs	RAM	Disk	Network Interfaces
Compute Physical Node	**88**	**64 GB**	**8 TB**	**7**

Table 3-16. Physical Compute Node specifications.

Some resources are saved in the main server for future VMs generation.

57 Two of the physical interfaces are used as bridged interfaces by KVM.

3.2.4.2 SW Requirements

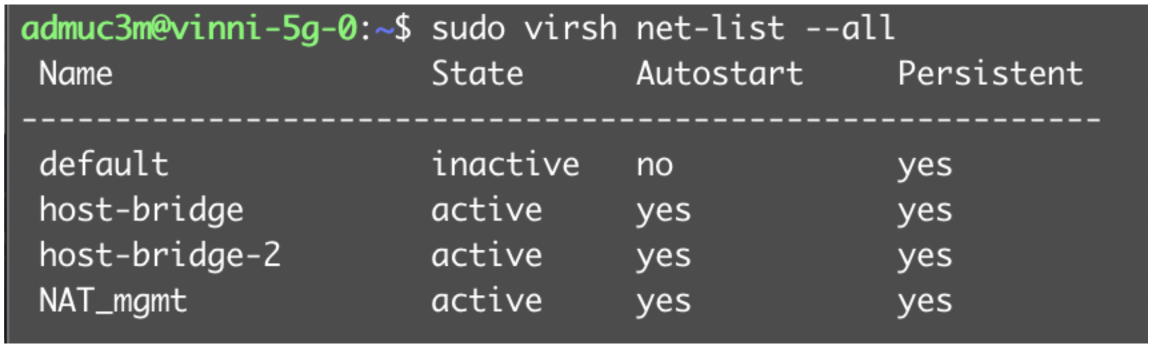
Equal to previous scenario, the new physical device OS is the same as the compute node's VM.

3.2.4.3 Configuration

3.2.4.3.1 Network Configuration

In this scenario, using KVM's NAT network from the previous scenario makes no sense as the new physical machine is not KVM dependent. To solve this "problem" a new physical network was required to 5TONIC's Datacenter to be used as an Internal Data Network, the 10.99.0.0/24 range was granted.

An Ethernet Cat-6 cable was connected on the secondary network interface of each physical machine to stablish the connection between them (there were no available switches at that moment and, as the 5TONIC's technician were on holidays, this was the only possible solution). Besides, the secondary network interface of the *"Datacenter Remote Main Server"* had to be turned into a bridge, as it was done with the main network interface in the previous scenario, to allow KVM to access this new network and enable its VMs to access it:



```
admuc3m@vinni-5g-0:~$ sudo virsh net-list --all
 Name                 State      Autostart     Persistent
----------------------------------------------------------
 default              inactive   no            yes
 host-bridge          active     yes           yes
 host-bridge-2        active     yes           yes
 NAT_mgmt             active     yes           yes
```

Figure 3-24. KVM's networks.

It is important to remark that the new Internal Data network has no gateway to the Internet and, thereupon, the multiple GW problem experienced in the previous scenario using the NAT network was solved but, due to this very reason, another network interface had to be attached to the Block and Object Storage VMs.

Once all the configuration was done, the following networks were available for this scenario:

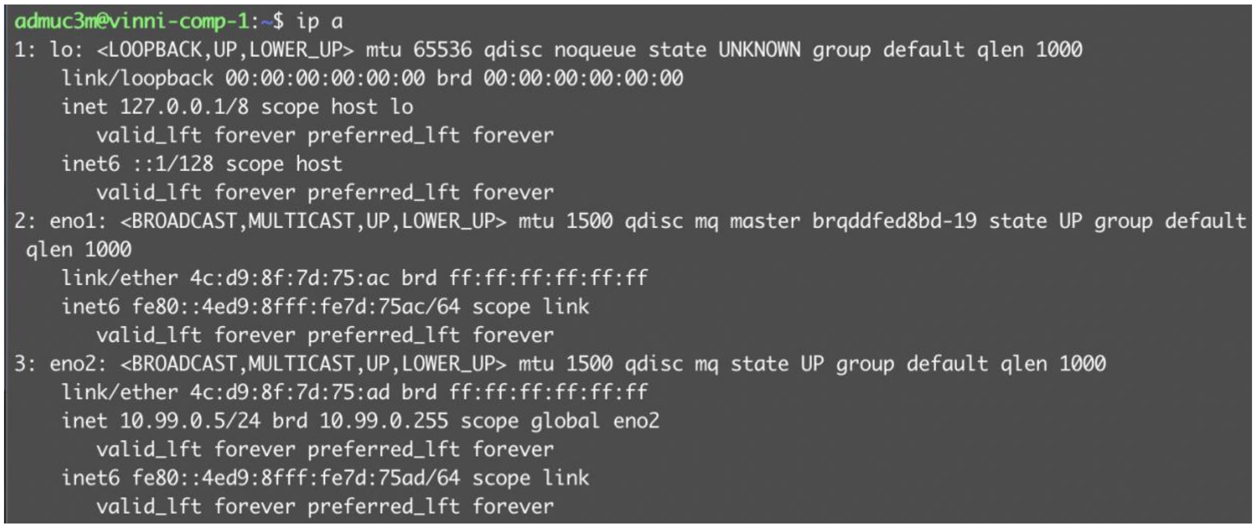
NETWORK	IP RANGE	GATEWAY
BR0 (PROVIDER)	10.5.10.0/24	10.5.0.1
BR1 (INTERNAL DATA)	10.99.0.0/24	-

Figure 3-25. Final scenario networks.

MACHINE	INTERFACE	IP	GW
	Eno1 (physical)	10.5.10.10	10.5.0.1

DATACENTER REMOTE MAIN SERVER	Eno2 (physical)	10.99.0.10	-
COMPUTE	Ens3 (physical)	10.5.10.5	10.5.0.1
	Ens4 (physical)	10.99.0.5	-
OSM	Ens3 (bridged)	10.5.10.2	10.5.0.1
CONTROLLER	Ens3 (bridged - br0)	10.5.10.4	10.5.0.1
	Ens4 (bridged - br1)	10.99.0.4	-
BLOCK STORAGE	Ens3 (bridged - br0)	10.5.10.6	10.5.0.1
	Ens4 (bridged - br1)	10.99.0.6	-
OBJECT STORAGE	Ens3 (bridged - br0)	10.5.10.7	10.5.0.1
	Ens4 (bridged - br1)	10.99.0.7	-

Table 3-17. IP assignation.

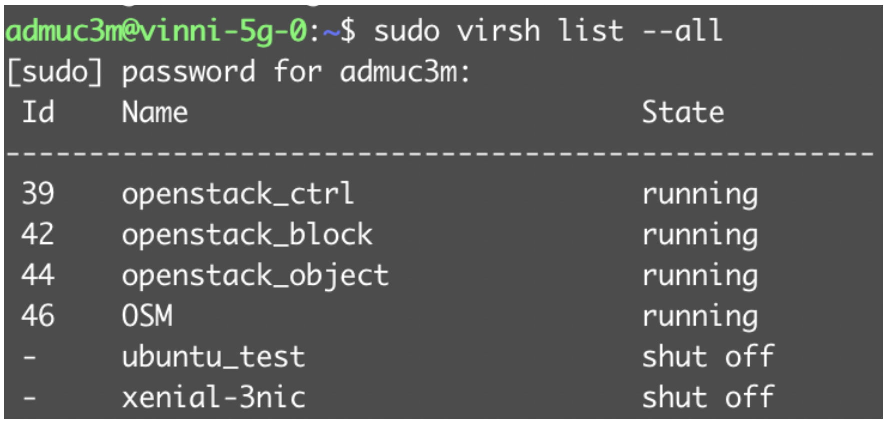


```
admuc3m@vinni-comp-1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master brqddfed8bd-19 state UP group default
 qlen 1000
    link/ether 4c:d9:8f:7d:75:ac brd ff:ff:ff:ff:ff:ff
    inet6 fe80::4ed9:8fff:fe7d:75ac/64 scope link
       valid_lft forever preferred_lft forever
3: eno2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 4c:d9:8f:7d:75:ad brd ff:ff:ff:ff:ff:ff
    inet 10.99.0.5/24 brd 10.99.0.255 scope global eno2
       valid_lft forever preferred_lft forever
    inet6 fe80::4ed9:8fff:fe7d:75ad/64 scope link
       valid_lft forever preferred_lft forever
```

Figure 3-26. Physical Compute Node IPs.

3.2.4.3.2 VMs Configuration

To avoid any configuration issues, all the previous scenario VMs were deleted and the ones needed for this scenario were launched with the same commands but adapted to the new HW requirements of each machine (even the same VNC ports were used). Bellow, the current running VMs are shown:

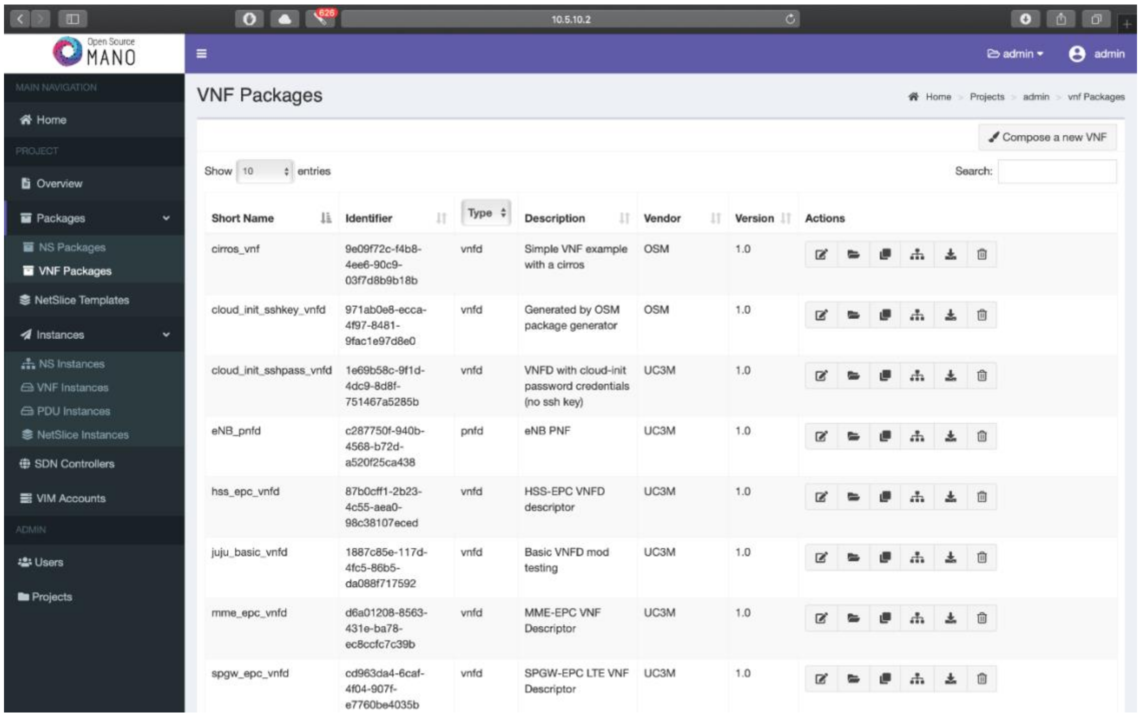


```
admuc3m@vinni-5g-0:~$ sudo virsh list --all
[sudo] password for admuc3m:
 Id    Name                           State
----------------------------------------------------
 39    openstack_ctrl                 running
 42    openstack_block                running
 44    openstack_object               running
 46    OSM                            running
 -     ubuntu_test                    shut off
 -     xenial-3nic                    shut off
```

Figure 3-27. KVM's VMs.

3.2.4.3.3 OSM Configuration

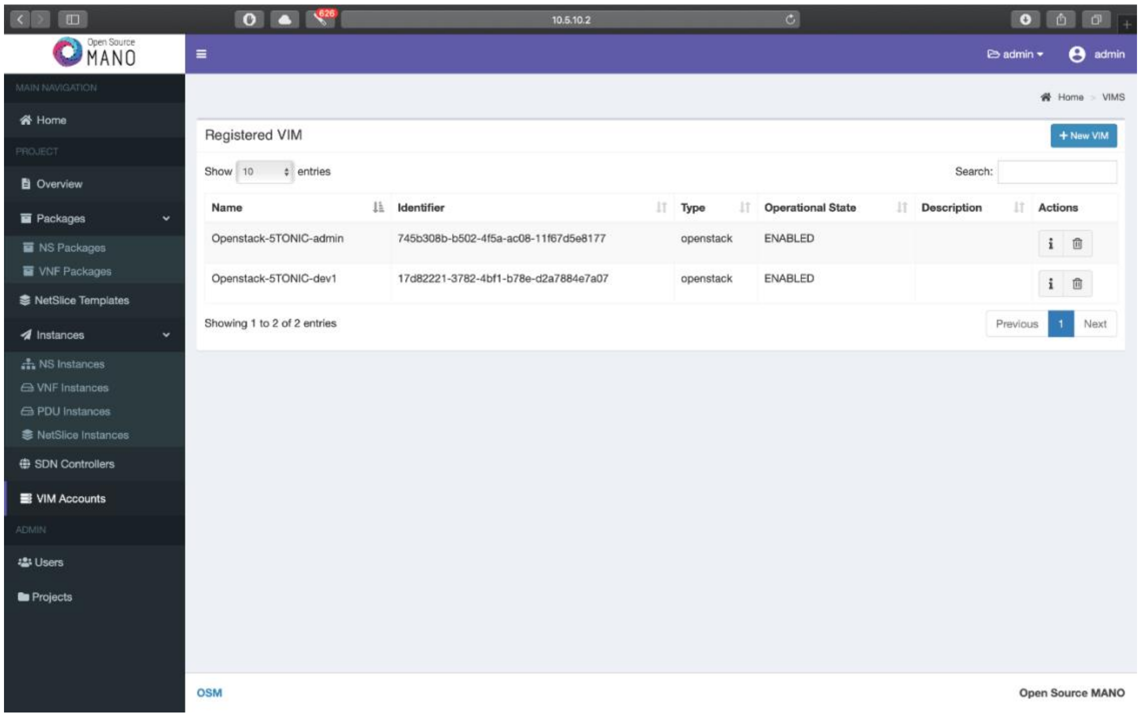
Equal to the previous scenario.



VNF Packages

Short Name	Identifier	Type	Description	Vendor	Version	Actions
cirros_vnf	9e09f72c-f4b8-4ee6-90c9-03f7d8b9b18b	vnfd	Simple VNF example with a cirros	OSM	1.0	
cloud_init_sshkey_vnfd	971ab0e8-ecca-4f97-8481-9fac1e97d8e0	vnfd	Generated by OSM package generator	OSM	1.0	
cloud_init_sshpass_vnfd	1e69b58c-9f1d-4dc9-8d8f-751467a5285b	vnfd	VNFD with cloud-init password credentials (no ssh key)	UC3M	1.0	
eNB_pnfd	c287750f-940b-4568-b72d-a520f25ca438	pnfd	eNB PNF	UC3M	1.0	
hss_epc_vnfd	87b0cff1-2b23-4c55-aea0-98c38107eced	vnfd	HSS-EPC VNFD descriptor	UC3M	1.0	
juju_basic_vnfd	1887c85e-117d-4fc5-86b5-da088f717592	vnfd	Basic VNFD mod testing	UC3M	1.0	
mme_epc_vnfd	d6a01208-8563-431e-ba78-ec8ccfc7c39b	vnfd	MME-EPC VNF Descriptor	UC3M	1.0	
spgw_epc_vnfd	cd963da4-6caf-4f04-907f-e7760be4035b	vnfd	SPGW-EPC LTE VNF Descriptor	UC3M	1.0	

Figure 3-28. OSM VNF packages view.



Registered VIM

Show 10 entries Search:

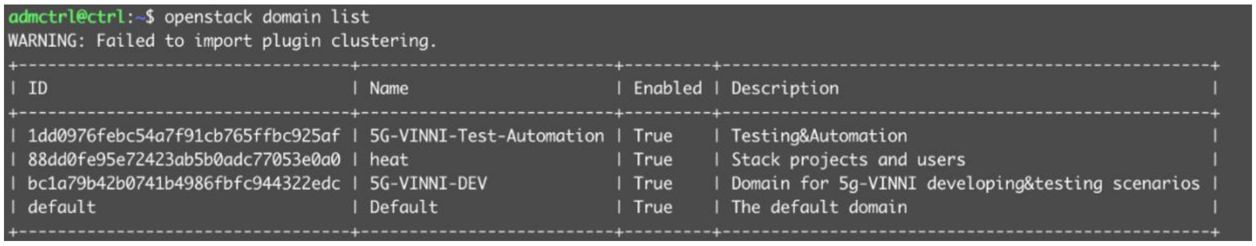
Name	Identifier	Type	Operational State	Description	Actions
Openstack-5TONIC-admin	745b308b-b502-4f5a-ac08-11f67d5e8177	openstack	ENABLED		
Openstack-5TONIC-dev1	17d82221-3782-4bf1-b78e-d2a7884e7a07	openstack	ENABLED		

Showing 1 to 2 of 2 entries Previous 1 Next

Figure 3-29. OSM VIM view.

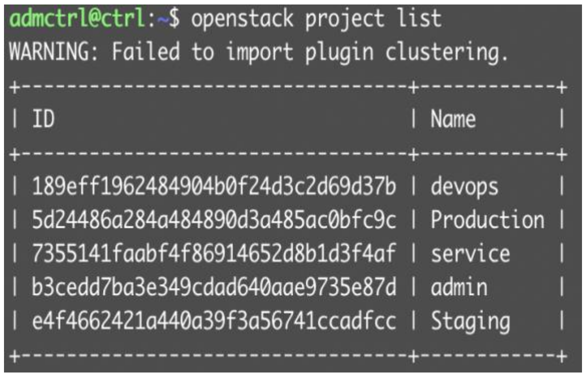
3.2.4.3.4 OpenStack Configuration

Same as in the previous scenario but this time *Libvirt* uses KVM instead of QEMU on the compute node and the *self-provider network* is 192.168.30.0/24 instead of 192.168.20.0/24. The installation process took only one day (around 9h), the experience obtained from the previous scenario utterly accelerated the process. More users, projects and domains had to be created to fulfill 5G-VINNI's testers requirements:

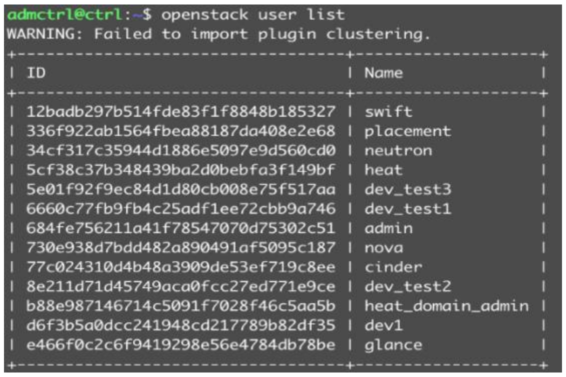


```
admctrl@ctrl:~$ openstack domain list
WARNING: Failed to import plugin clustering.
+----------------------------------+--------------------------+---------+---------------------------------------------------+
| ID                               | Name                     | Enabled | Description                                       |
+----------------------------------+--------------------------+---------+---------------------------------------------------+
| 1dd0976febc54a7f91cb765ffbc925af | 5G-VINNI-Test-Automation | True    | Testing&Automation                                |
| 88dd0fe95e72423ab5b0adc77053e0a0 | heat                     | True    | Stack projects and users                          |
| bc1a79b42b0741b4986fbfc944322edc | 5G-VINNI-DEV             | True    | Domain for 5g-VINNI developing&testing scenarios |
| default                          | Default                  | True    | The default domain                                |
+----------------------------------+--------------------------+---------+---------------------------------------------------+
```

Figure 3-30. OpenStack Domain list.

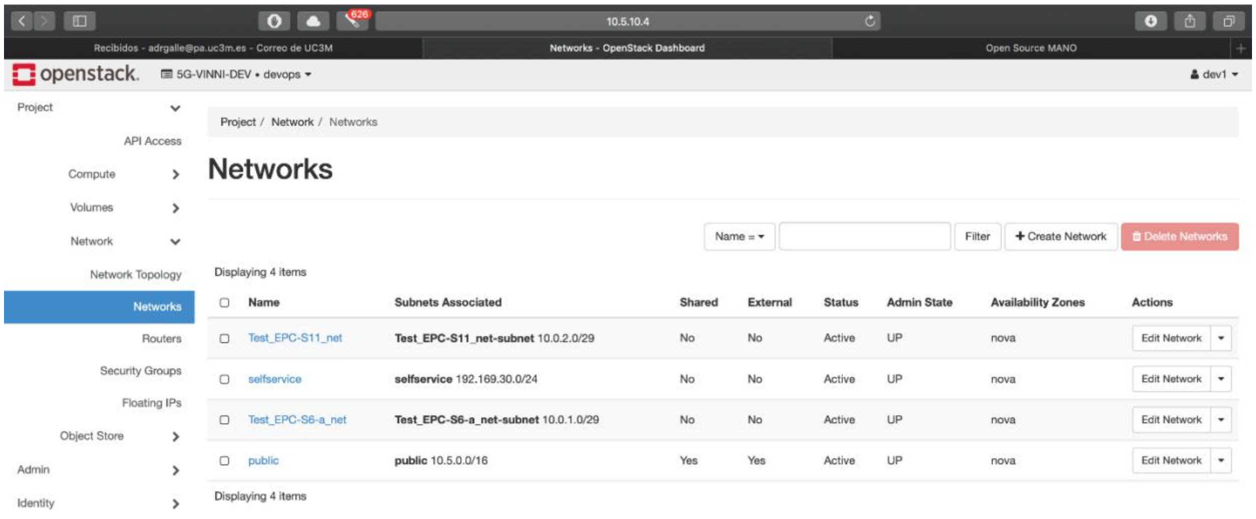


```
admctrl@ctrl:~$ openstack project list
WARNING: Failed to import plugin clustering.
+----------------------------------+------------+
| ID                               | Name       |
+----------------------------------+------------+
| 189eff1962484904b0f24d3c2d69d37b | devops     |
| 5d24486a284a484890d3a485ac0bfc9c | Production |
| 7355141faabf4f86914652d8b1d3f4af | service    |
| b3cedd7ba3e349cdad640aae9735e87d | admin      |
| e4f4662421a440a39f3a56741ccadfcc | Staging    |
+----------------------------------+------------+
```



```
admctrl@ctrl:~$ openstack user list
WARNING: Failed to import plugin clustering.
+----------------------------------+-------------------+
| ID                               | Name              |
+----------------------------------+-------------------+
| 12badb297b514fde83f1f8848b185327 | swift             |
| 336f922ab1564fbea88187da408e2e68 | placement         |
| 34cf317c35944d1886e5097e9d560cd0 | neutron           |
| 5cf38c37b348439ba2d0bebfa3f149bf | heat              |
| 5e01f92f9ec84d1d80cb008e75f517aa | dev_test3         |
| 6660c77fb9fb4c25adf1ee72cbb9a746 | dev_test1         |
| 684fe756211a41f78547070d75302c51 | admin             |
| 730e938d7bdd482a890491af5095c187 | nova              |
| 77c024310d4b48a3909de53ef719c8ee | cinder            |
| 8e211d71d45749aca0fcc27ed771e9ce | dev_test2         |
| b88e987146714c5091f7028f46c5aa5b | heat_domain_admin |
| d6f3b5a0dcc241948cd217789b82df35 | dev1              |
| e466f0c2c6f9419298e56e4784db78be | glance            |
+----------------------------------+-------------------+
```

Figure 3-31. OpenStack Users and projects.



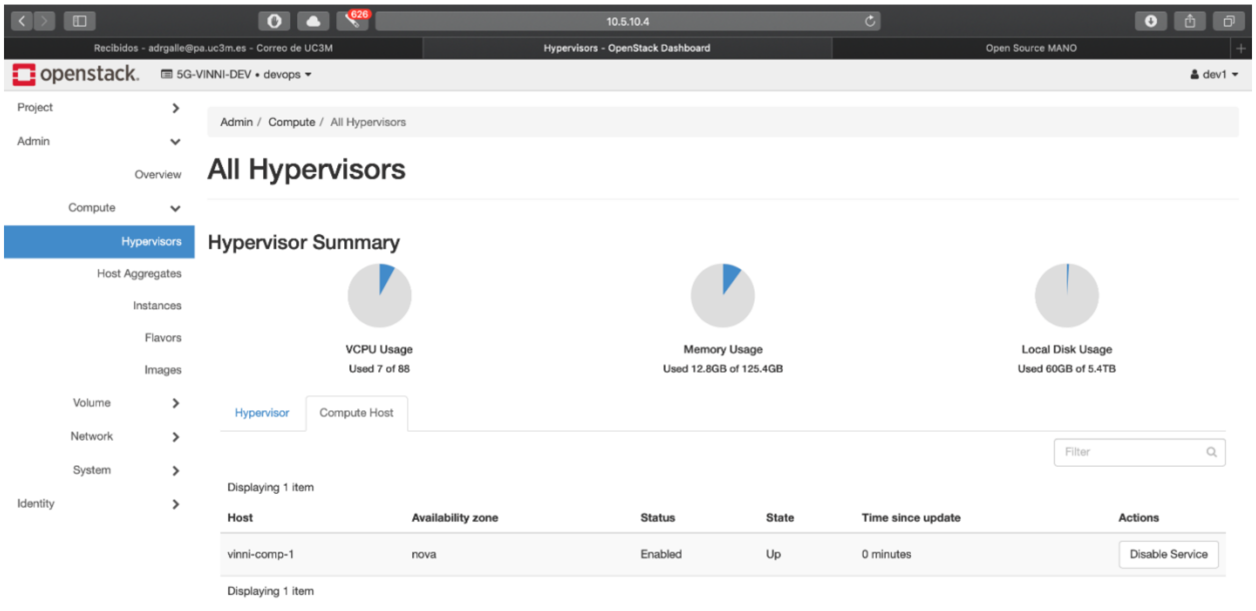
Networks

Displaying 4 items

Name	Subnets Associated	Shared	External	Status	Admin State	Availability Zones	Actions
Test_EPC-S11_net	Test_EPC-S11_net-subnet 10.0.2.0/29	No	No	Active	UP	nova	Edit Network
selfservice	selfservice 192.169.30.0/24	No	No	Active	UP	nova	Edit Network
Test_EPC-S6-a_net	Test_EPC-S6-a_net-subnet 10.0.1.0/29	No	No	Active	UP	nova	Edit Network
public	public 10.5.0.0/16	Yes	Yes	Active	UP	nova	Edit Network

Displaying 4 items

Figure 3-32. OpenStack networks.



All Hypervisors

Hypervisor Summary

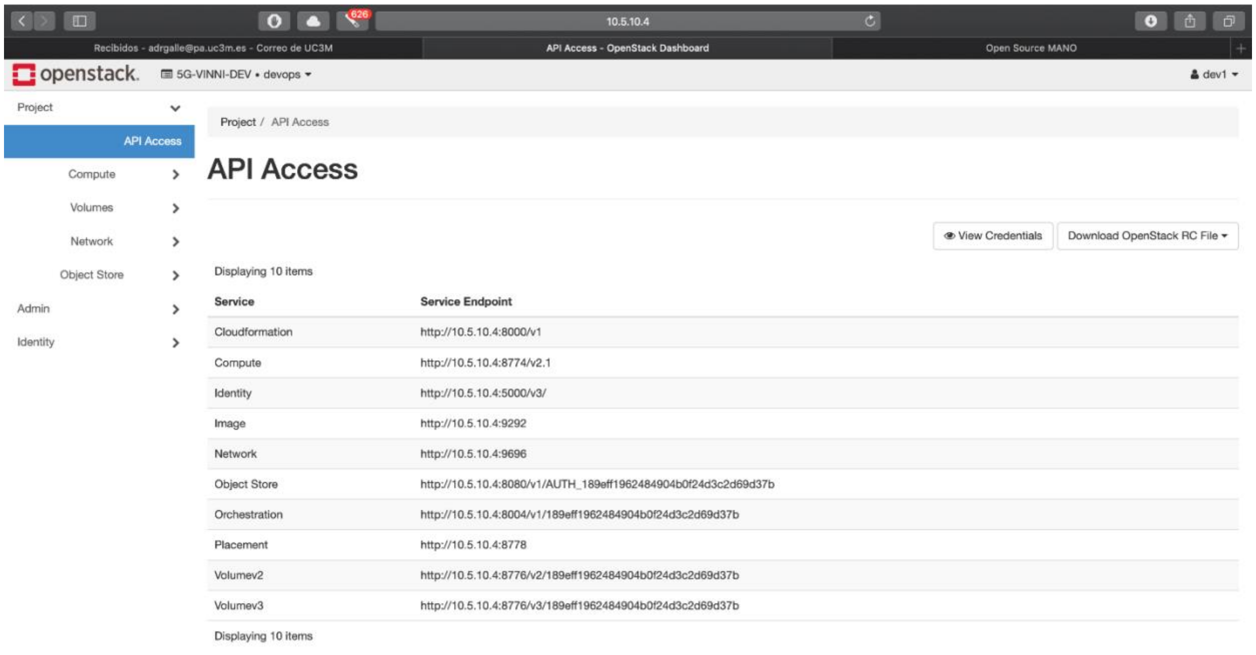
VCPU Usage	Memory Usage	Local Disk Usage
Used 7 of 88	Used 12.8GB of 125.4GB	Used 60GB of 5.4TB

Displaying 1 item

Host	Availability zone	Status	State	Time since update	Actions
vinni-comp-1	nova	Enabled	Up	0 minutes	Disable Service

Displaying 1 item

Figure 3-33. OpenStack's Compute Node Resources.



API Access

Displaying 10 items

Service	Service Endpoint
Cloudformation	http://10.5.10.4:8000/v1
Compute	http://10.5.10.4:8774/v2.1
Identity	http://10.5.10.4:5000/v3/
Image	http://10.5.10.4:9292
Network	http://10.5.10.4:9696
Object Store	http://10.5.10.4:8080/v1/AUTH_189eff1962484904b0f24d3c2d69d37b
Orchestration	http://10.5.10.4:8004/v1/189eff1962484904b0f24d3c2d69d37b
Placement	http://10.5.10.4:8778
Volumev2	http://10.5.10.4:8776/v2/189eff1962484904b0f24d3c2d69d37b
Volumev3	http://10.5.10.4:8776/v3/189eff1962484904b0f24d3c2d69d37b

Displaying 10 items

Figure 3-34. OpenStack Endpoints.

A huge increase on performance was noticed in the current scenario and no more nested virtualization problems were encountered.

4 Test, Evaluation and Results

This chapter is, probably, one of the most important in all the project as it is focused in testing the acquired knowledge since the beginning of it, evaluate through different trials the generated scenarios and extract some technical conclusions from the obtained results in each of the tests. It is also, aside from the past chapter, the most empiric and experimental chapter of this Master Thesis as, thank to these tests, the final objective of the project, building a NSA E2E 5G scenario through Open Platforms, is going to be build and fulfilled.

Several tests have been generated due to the huge number of concepts involved in their development and the wide range of deployed scenarios available to be used. The tests have been structured on a "*bottom-up*" approach, this means that before achieving the final objective, other tests have been carried out in order to understand and check the involved technologies, platforms and required features. Five sections compose this chapter, one for each of the developed tests, starting from the "easiest" (bottom) in the first scenario, up to the final test (top) in the latest scenario:

1. **Ping-Pong Trial:** Simple and basic test that will help to understand the OSM VNFD/NSD IM definitions.
2. **Day-0 Config:** Focused on learning the insights of OSM's Day-0 Config concept through Cloud-Config files.
3. **Day-1 Config:** Includes various tests that show and explain how to achieve OSM's Day-1 Config concept through different SWs. A minimal example of OSM's Day-2 Config Concept is also given, but it is not further developed as it has no use in this Master Thesis.
4. **vEPC:** Integrates all the previous studied concepts and applies them in order to build a fully automated LTE vEPC.
5. **E2E NS NSA 5G scenario:** Builds a hybrid infrastructure that incorporates the vEPC and a PNF acting as eNB, combining them into an eMBB NS.

4.1 Test 1: Ping-Pong

This first test has the objective of acquiring the required knowledge to build any kind of VNF and Network Service over OSM using its IM descriptors. OSM has one of the most complete IM currently available and, besides, both VNFD and NSD IMs are ETSI-NFV compliant (NST IM is still being implemented to be ETSI-NFV compliant in OSM Rel. FIVE). These IMs can be accessed in different formats:

- Plain Text: https://osm.etsi.org/wikipub/index.php/OSM_Information_Model
- YANG:https://osm.etsi.org/gitweb/?p=osm/IM.git;a=tree;f=models/yang;h=93cb280243a958eb32f52b1b90114d74ca9e1f26;hb=HEAD
- Navigable: http://osm-download.etsi.org/repository/osm/debian/ReleaseSIX/docs/osm-im/osm_im_trees/vnfd.html

In first place, it is important to know the main objects in each of the IMs as they are going to be used repeatedly in every VNFD/NSD. In the following tables a slight description is given for the VNFDs and NSDs IMs:

VNFD	
Element	**Description**
vnfd	VNFDs' main element, the rest of the elements are children of this one. VNFD's id, name, version and description are some of its direct children.
mgmt-interface	Every VNFD needs one management interface, even though it is not connected to a network with Internet access.
vdu	Virtual Device Unit. Is the element that defines with its children the VNF virtual characteristics: vcpus, RAM, disk, image, network interfaces…
connection-point	Establishes a reference point to be used in a potential Network Service to connect the current VNF with others.

Table 4-1. OSM VNFD IM main elements.

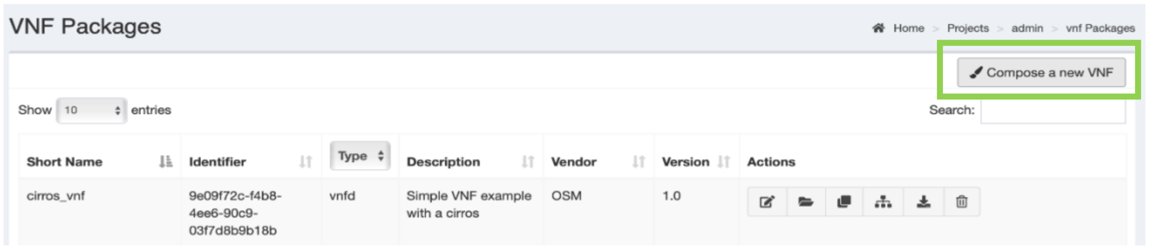
NSD	
Element	**Description**
nsd	NSDs' main object. Same characteristics as *vnfd* object.
constituent-vnfd	Allows to specify which defined VNFs, through their VNFDs name, compose the current NSD.
vld	Virtual Link Descriptor. Allows to create different networks to connect the different VNFDs connection-points.

Table 4-2. OSM NSD IM main elements.

Much more elements compose both IMs, but for this Test those ones are the most important (in the following tests more will be shown).

Once the IMs are known, the next step is to create the files containing the proper VNFD and NSD; to do so, OSM Rel. FIVE allows various ways of uploading the files to the OSM machine:

1. **OSM Portal:** Inside OSM's dashboard, in the *"VNF/NS Packages"* tag, in the upper-right corner a button name *"Compose a new VNF/NS*[58]*"* is available. If selected, the user can build a YAML file directly into the portal. The major disadvantage of this method is that, although it allows a rapid creation of the VNFD/NSD, it does not allow to add extra configuration through other files (Day-0, Day-1 and Day-2 config).



Short Name	Identifier	Type	Description	Vendor	Version	Actions
cirros_vnf	9e09f72c-f4b8-4ee6-90c9-03f7d8b9b18b	vnfd	Simple VNF example with a cirros	OSM	1.0	

Figure 4-1. OSM VNF/NSD creation method-1.

2. **Manually:** If the previous method is not used, OSM requires "**.tar.gz*" files for both, CLI and GUI, in order to upload VNFDs or NSDs. These files MUST have in the root the YAML files containing the VNFDs/NSDs, apart from that, several folders MAY be added such as: icons (OSM Rel.FIVE does not support showing icons in its portal, this was discovered after launching this test), images (allows using OS images that are not addressed in OpenStack), charms (Day-1 Config), cloud-init (Day-0 config)… Once the internal file structure is defined, the "**.tar.gz*" MUST be generated with the following commands:

```
$ cd <path to vnfd/nsd>

$ find * -type f -print | while read line; do md5sum $line >> checksums.txt;
done  #Generates a checksum file (required)

$ tar zcvf <vnf/nsd_name>_[vnf|ns].tar.gz <vnf/nsd_name>_[vnf|ns]
```

Text Box 4-1. OSM VNF/NSD creation method 2.

58 In this case NS stands for Network Service.

The resulting files have to be uploaded to the OSM machine, this process can be done using the GUI by drag&droping the file into the *"VNF/ NS Packages"* tag or using the following command in the CLI:

```
# VNFDs

$ osm vnfd-create <vnfd_name>.tar.gz

# NSDs

$ osm nsd-create <nsd_name>.tar.gz
```

Text Box 4-2. OSM VNFD/NSD CLI uploading.

Both methods will pop-up error messages if something went wrong.

3. **DevOps Repo Tools:** This method was discovered after this first test and it is the preferred by the author due to its tremendous flexibility and 100% no-error chance in the files structure (sometimes OSM pop-ups errors to one file that has the same structure than other if they have been made manually). In order to use it, the following commands SHOULD be executed:

```
# Download git devops repo

$ git clone https://osm.etsi.org/gerrit/osm/devops

# Create a folder with all the required files (VNFs)

$ ./devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t vnfd --
image <IMAGE_PATH> -c <VNF_NAME>

# Create a folder with all the required files (NSs)

$ ./devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t nsd -c
<NS_NAME>

# Create *.tar.gz file (VNFs). (-N argument does not delete the folder)

$ ./devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t vnfd -N
<VNF_NAME>_vnfd

# Create *.tar.gz file (VNFs). (-N argument does not delete the folder)

$ ./devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t nsd -N
<NS_NAME>_nsd
```

Text Box 4-3. OSM VNFD/NSD creation method-3.

4.1.1 Test Characteristics

This section provides a summary of the current Test Characteristics.

Test 1	
Scenario	Initial
VNFs	2
VNFDs	1
NSs	1
NSDs	1
PNFs	0
PNFDs	0
NSTs	0
Networks	1
Total RAM	512 MB
Total vCPUS	2
Total Disk	4 GB
Day-0 Config	None
Day-1 Config	None
Day-2 Config	None

Table 4-3. Test 1 summary.

As it can be seen, it is a really light test that does not require high computational usage.

4.1.2 Test Description

This test was the first contact with a NFVI/NFVO 5G environment so, the main focus was to understand how to create more complex models with this as a base. It creates a single Network Service composed by two identical VNFs that run a cloud-testing OS image known as Cirros[59] and are connected by an internal network, without Internet connection, that also acts as the management network. The main objective of this test is to build a correct Network Service and to end up being capable of sending pings between both machines.

To create VNFD and NSD files the manual method was used (at the time of developing this tests the author did not know the existence of the third method), resulting in the following files structure:

[59] https://launchpad.net/cirros

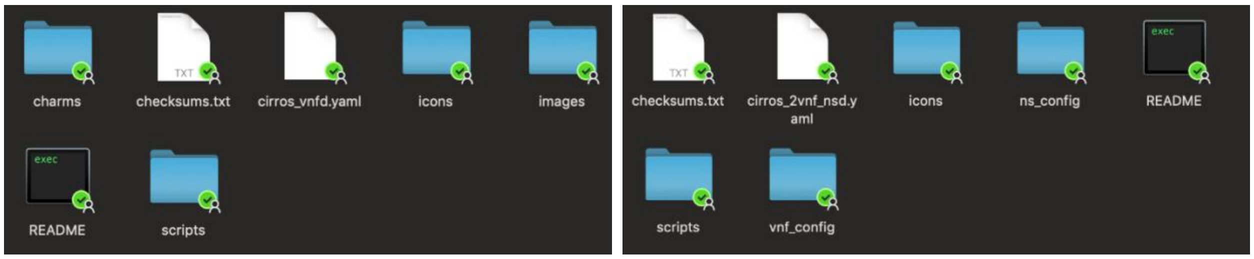


Figure 4-2. Test 1 files structure.

Neither *charms*, *scripts*, *images, ns_config* or *vnf_config* folders were used but they were needed to upload the files correctly to the OSM machine. The files were uploaded drag&dropping them to the OSM dashboard.

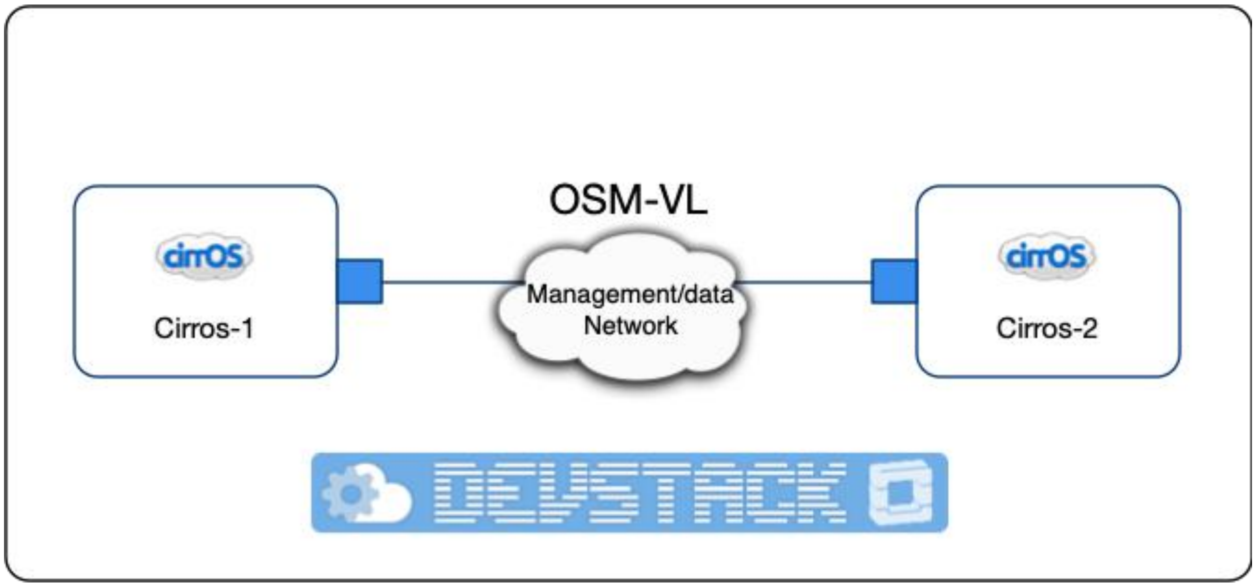


Figure 4-3. Test 1 High level view.

In the following paragraphs, as they are the first descriptors, both descriptors will be broken-down to analyze each of their elements[60] (in the upcoming tests only the new features will be commented/explained).

60 If the complete version is desired, see *Annex 11.1 Cirros* VNFD *and Annex 12.1 Test-1 NSD.*

Let's start with the VNFD:

```
vnfd:

    -   id: cirros_vnfd

        name: cirros_vnf

        short-name: cirros_vnf

        description: Cirros Basic VNF example

        vendor: AGS

        version: '1.1'

        logo: cirros-64.png
```

Text Box 4-4. Cirros VNFD part 1.

Each metadata field means:

- **id:** VNF identifier (must be unique).
- **name:** Name of the VNF.
- **short-name:** Short name of the VNF.
- **logo:** Name of the icon image file (image file must be inside of the icons folder).
- **vendor:** Person/Organization that created the VNF.
- **version:** Version of the VNF.
- **description:** Brief description of the VNF.

After the initial parameters are defines, connection-points must be addressed:

```
connection-point:

            -   name: eth0

                type: VPORT
```

Text Box 4-5. Cirros VNFD part 2.

Each connection point is composed by the following fields:

- **name:** Name that represents the connection point (must be unique).
- **type:** Type of the connection point port (normally VPORT).

Finally, the VM resources must be defined through the Virtual Device Unit (VDU) metadata

```
vdu:
        -   id: cirros_vnfd-VM
            name: cirros_vnfd-VM
            description: cirros_vnfd-VM
            count: 1
            vm-flavor:
                vcpu-count: 1
                memory-mb: 256
                storage-gb: 2
            image: cirros034
            interface:
            -   name: eth0
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
                external-connection-point-ref: eth0
```

Text Box 4-6. Cirros VNFD part 3.

field:

At least, one VDU MUST be defined in every VNFD. Each parameter is described below:

- **id:** VDU identifier (must be unique)
- **name:** Name of the VDU
- **description:** Brief description
- **count:** Number of VDUs to be defined per VNFD.
- **vm-flavor:** Data structure that holds what computational resources the VDU needs
  - **memory-mb:** Amount of RAM in Mbytes
  - **storage-gb:** Amount of disk in GBytes
  - **vcpu-count:** Amount of vCPUs
- **image:** Name of the image to be used by the VDU
- **interface:** Mapping between the VNF connection points and interfaces provided to the VDU
  - **name:** Interface's name.
  - **type:** It can be defined as an external or internal interface.

- **virtual-interface:** Interface's driver type (ex.: VIRTIO, PARAVIRT,OM-MGMT, etc.). VIRTIO type is recommended.
- **external-connection-point-ref:** Mapping between the defined interface and VNF connection point.

Let's follow up with the NSD:

```
nsd:

    -   id: cirros_2vnf_nsd

        name: cirros_2vnf_ns

        short-name: cirros_2vnf_ns

        description: Cirros Example Network Service

        vendor: AGS

        version: '1.1'

        logo: osm_2x.png
```

Text Box 4-7. Test 1 NSD part 1.

This first part is composed by the same/similar fields than the VNFD.

```
constituent-vnfd:

        -   member-vnf-index: 1

            vnfd-id-ref: cirros_vnfd

        -   member-vnf-index: 2

            vnfd-id-ref: cirros_vnfd
```

Text Box 4-8. Test 1 NSD part 2.

Each element in this list is composed by the following fields:

- **member-vnf-index:** Index to be referenced by.
- **vnfd-id-ref:** VNFD identifier by which the VNF is known in OSM (must be unique).

Finally, the VLDs must be defined in order to interconnect the different VNFs:

```
vld:
            -   id: vld1
                name: cirros_2vnf_nsd_vld1
                short-name: vld1
                type: ELAN
                mgmt-network: 'true'
                vnfd-connection-point-ref:
                -   member-vnf-index-ref: 1
                    vnfd-id-ref: cirros_vnfd
                    vnfd-connection-point-ref: eth0
                -   member-vnf-index-ref: 2
                    vnfd-id-ref: cirros_vnfd
                    vnfd-connection-point-ref: eth0
```

Text Box 4-9. Test 1 NSD part 3.

Each metadata field means:

- **id:** VLD identifier (must be unique).
- **name:** VLD name.
- **short-name:** VLD short name.
- **type:** VLD type (ELAN is the only value supported).
- **vnfd-connection-point-ref:** Reference list to VNF connection points.
    - **member-vnf-index-ref:** Reference index defined in the constituent-vnfd.
    - **vnfd-id-ref:** VNFD identifier.
    - **vnfd-connection-point-ref:** VNFD defined connection point.

Both, the VNFD and the NSD use connection-points (CP): In a VNFD a CP is a virtualized network interface created with the given characteristics in the VNF VM; in a NSD it is the same but it is used to stablish the endpoints of the Virtual Link between the composing VNFs of the Network Service:

Now that every field has been explained and that both descriptors are known and understandable, OSM graph representation of the VNFD and the NSD is presented:

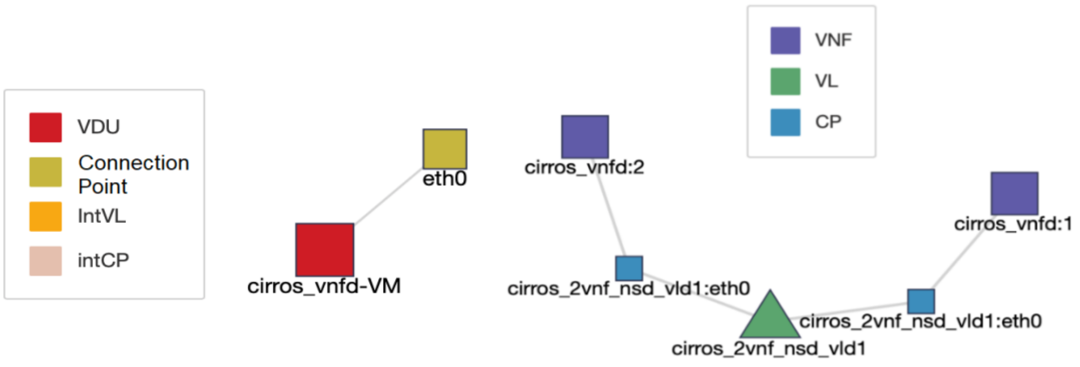


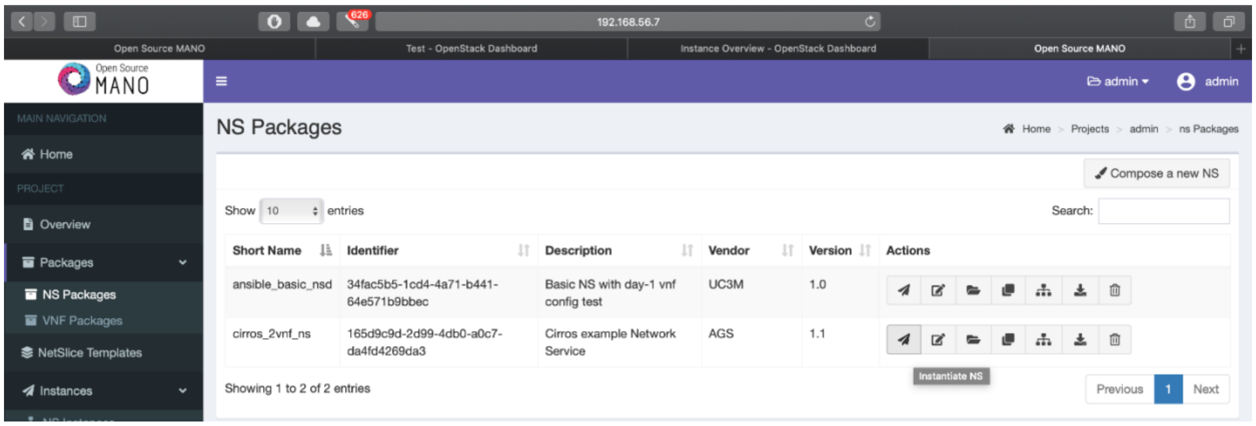
Figure 4-4. Test 1 VNFD/NSD graphs.

As expected, the VNF is composed by a single VDU with one CP called *eth0* and the Network Service is composed by two VNFs with each of their CPs connected through a VLD. As it can be seen, the CP exposed in the VNF graph and the CPs exposed in the NSD are a representation of the same virtualized network interfaces.

4.1.3 Test 1 Results

In this section, the previous NSD is launched from OSM and its correct instantiation and behavior are checked.

To execute a Network Service in OSM Rel. FIVE two methods are available: GUI and CLI; in this Master Thesis only the first will be used as it gives more information about the errors, if occurred. To do so, press the spaceship-like button inside the proper Network Service tag and complete the form with the desired parameters:



NS Packages

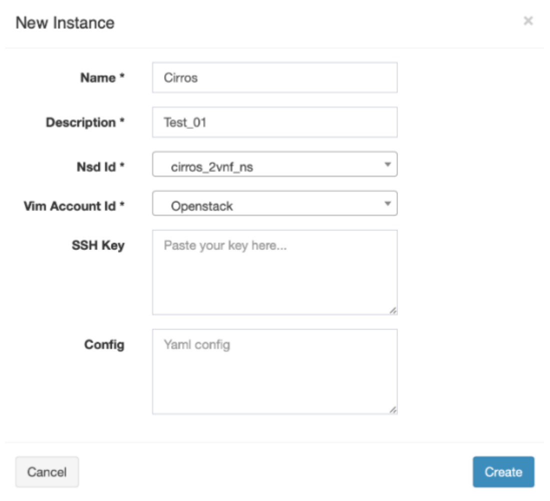
Show 10 entries

Search:

Short Name	Identifier	Description	Vendor	Version	Actions
ansible_basic_nsd	34fac5b5-1cd4-4a71-b441-64e571b9bbec	Basic NS with day-1 vnf config test	UC3M	1.0	
cirros_2vnf_ns	165d9c9d-2d99-4db0-a0c7-da4fd4269da3	Cirros example Network Service	AGS	1.1	

Showing 1 to 2 of 2 entries

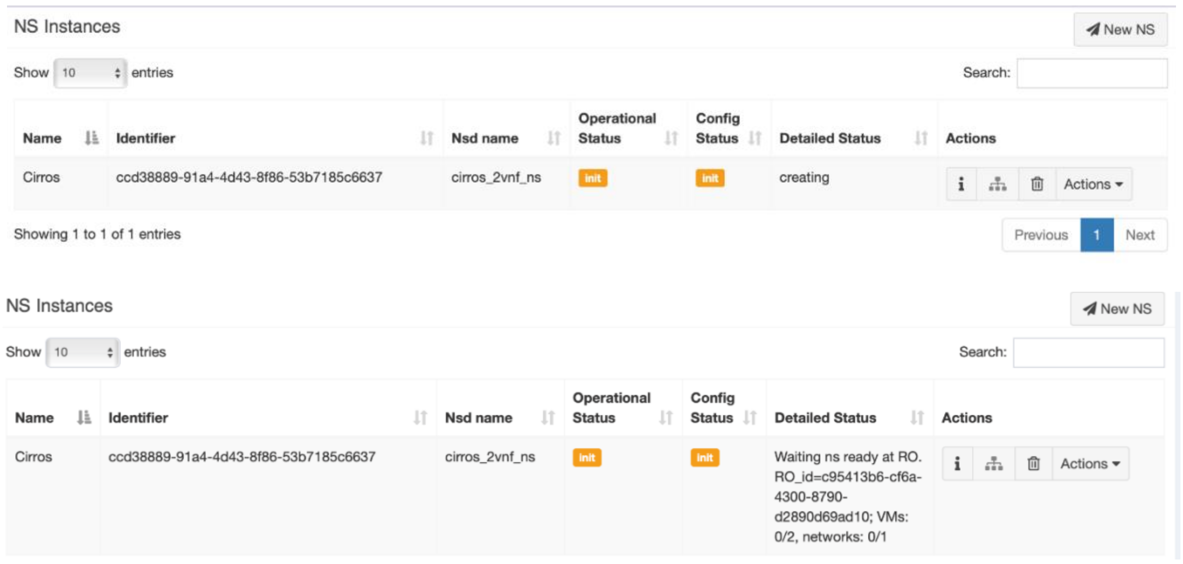
Figure 4-5. Instantiation button.


New Instance

Name *	Cirros
Description *	Test_01
Nsd Id *	cirros_2vnf_ns
Vim Account Id *	Openstack
SSH Key	Paste your key here...
Config	Yaml config

Figure 4-6. Network Service Instance form.

Once the desired parameters have been introduced, if a correct VIM account is configured the OSM RO (Resource Orchestrator) will start to send request to OpenStack in order to build the desired Network Service. If something goes wrong at this point, OSM will prompt a message reporting an issue; the problem is that, generally, these reports are technically poor and they are worthless, so it is recommended to use OpenStack logs instead.


NS Instances

Show 10 entries

Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
Cirros	ccd38889-91a4-4d43-8f86-53b7185c6637	cirros_2vnf_ns	init	init	creating	Actions

Showing 1 to 1 of 1 entries

NS Instances

Show 10 entries

Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
Cirros	ccd38889-91a4-4d43-8f86-53b7185c6637	cirros_2vnf_ns	init	init	Waiting ns ready at RO. RO_id=c95413b6-cf6a-4300-8790-d2890d69ad10; VMs: 0/2, networks: 0/1	Actions

Figure 4-7. OSM Network Service Creation process.

If everything went with no issues, the operational status and config status should turn green and have *"running"* and *"configured"* messages respectively. Moreover, in OpenStack the Network Service composing VMs should be available and the current network/s created:

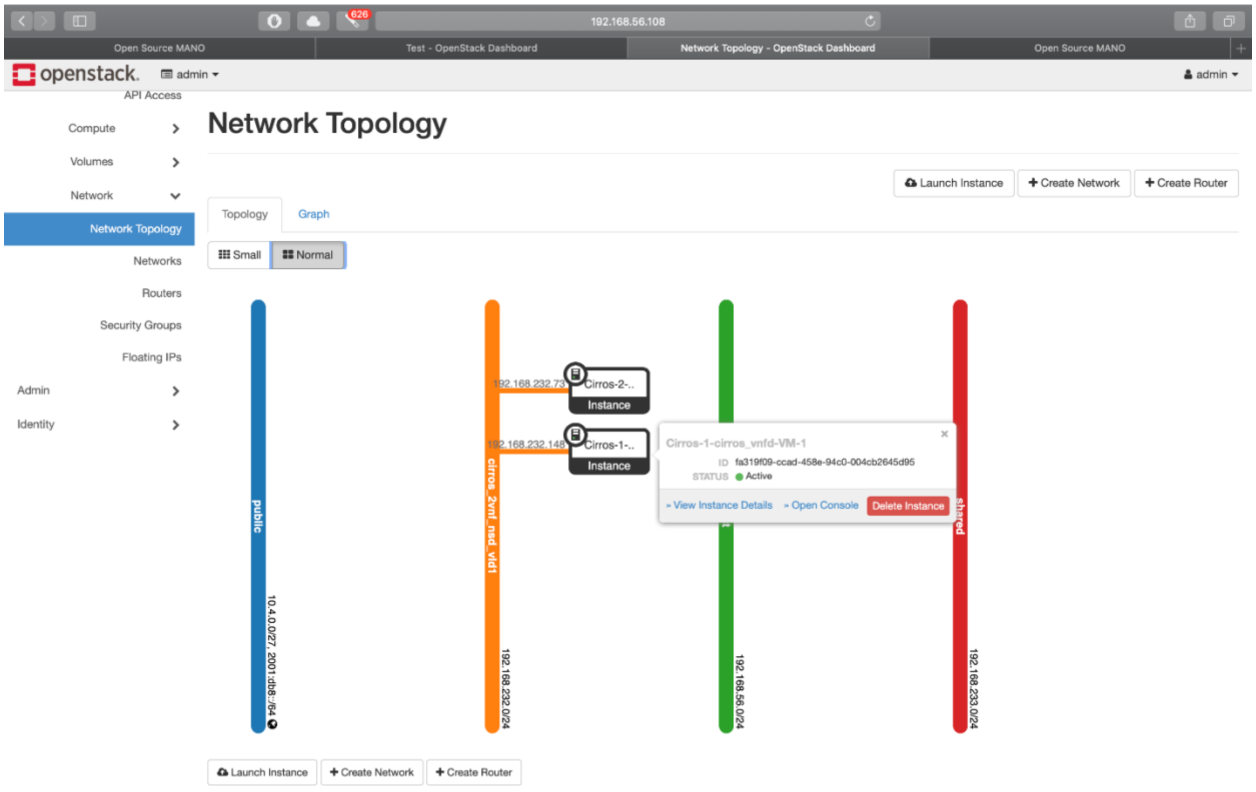
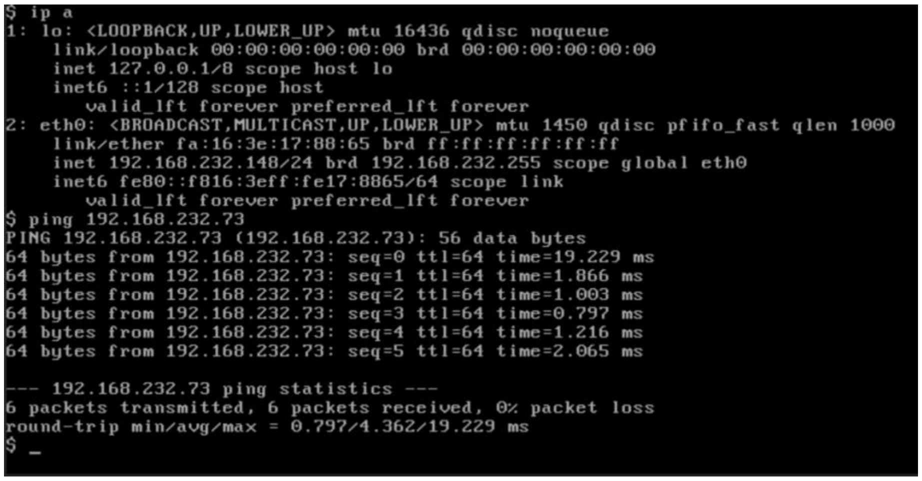


Figure 4-8. Test 1 Network Service network topology.

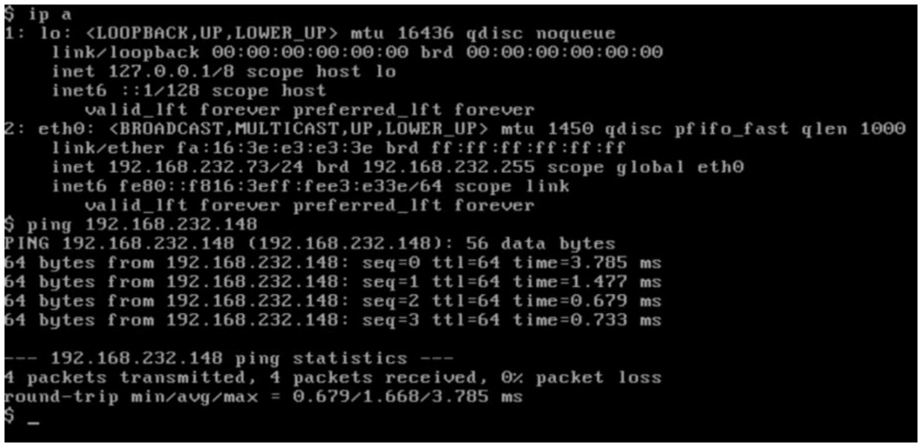
As it can be seen in the figure above both VNFs are up and running inside a network that, as no IP conditional parameters have been defined in the NSD, has been randomly (inside a set of internal network ranges) generated by OpenStack.

Finally, it only remains to check if the behavior of both VNFs is as expected. To check this point, a ping is sent from one VNF to the other and vice versa (in both cases OpenStack VNC console is utilized):



```
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc pfifo_fast qlen 1000
    link/ether fa:16:3e:17:88:65 brd ff:ff:ff:ff:ff:ff
    inet 192.168.232.148/24 brd 192.168.232.255 scope global eth0
    inet6 fe80::f816:3eff:fe17:8865/64 scope link
       valid_lft forever preferred_lft forever
$ ping 192.168.232.73
PING 192.168.232.73 (192.168.232.73): 56 data bytes
64 bytes from 192.168.232.73: seq=0 ttl=64 time=19.229 ms
64 bytes from 192.168.232.73: seq=1 ttl=64 time=1.866 ms
64 bytes from 192.168.232.73: seq=2 ttl=64 time=1.003 ms
64 bytes from 192.168.232.73: seq=3 ttl=64 time=0.797 ms
64 bytes from 192.168.232.73: seq=4 ttl=64 time=1.216 ms
64 bytes from 192.168.232.73: seq=5 ttl=64 time=2.065 ms

--- 192.168.232.73 ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 0.797/4.362/19.229 ms
$ _
```

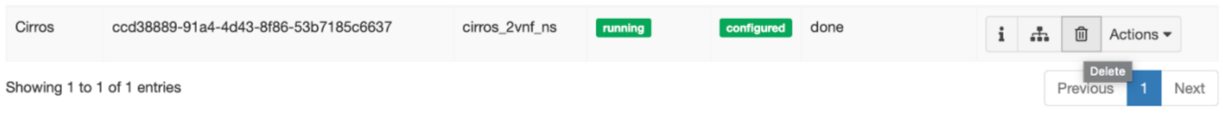
Figure 4-9. Test 1 ping from Cirros_VNF1 to Cirros_VNF2.



```
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc pfifo_fast qlen 1000
    link/ether fa:16:3e:e3:e3:3e brd ff:ff:ff:ff:ff:ff
    inet 192.168.232.73/24 brd 192.168.232.255 scope global eth0
    inet6 fe80::f816:3eff:fee3:e33e/64 scope link
       valid_lft forever preferred_lft forever
$ ping 192.168.232.148
PING 192.168.232.148 (192.168.232.148): 56 data bytes
64 bytes from 192.168.232.148: seq=0 ttl=64 time=3.785 ms
64 bytes from 192.168.232.148: seq=1 ttl=64 time=1.477 ms
64 bytes from 192.168.232.148: seq=2 ttl=64 time=0.679 ms
64 bytes from 192.168.232.148: seq=3 ttl=64 time=0.733 ms

--- 192.168.232.148 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.679/1.668/3.785 ms
$ _
```

Figure 4-10. Test 1 ping from Cirros_VNF2 to Cirros_VNF1.

Everything went as expected and both VNFs were able to communicate with each other through the generated Virtual Link. To an end, if the Network Service is to be erased, the paper button in OSM portal has to be pressed (it is strongly discouraged to delete anything from OpenStack as it will deliver several issues to OSM when the last wants to delete the Network Service).



Cirros	ccd38889-91a4-4d43-8f86-53b7185c6637	cirros_2vnf_ns	running	configured	done	Actions ▾

Showing 1 to 1 of 1 entries

Figure 4-11. Test 1 deletion.

4.2 Test 2: Day-0 Config

This second test is oriented to explore a more complex network configuration at the Network service level, also to use non-trial OS cloud images (Ubuntu will be used instead of Cirros) and, as a main objective, to acknowledge the Day-0 config concept in OSM.

Day-0 config is based on a Linux package, known as Cloud-Init, that automates the initial configuration of a VM. In order to use it, the cloud image/s requested for this test must incorporate the cloud-init package (most of the current GNU/Linux cloud images incorporate it) and a *cloud-init* configuration file (contains the cloud-init configuration data). This piece of SW allows the designers to:

- Setting an instance hostname
- Generating instance private SSH keys and adding public SSH keys to the user's authorized keys path
- Configuring network devices
- Adding and configuring users and groups
- Setting default locales
- …

It is important to remark that although cloud-init is available for almost all Linux VMs, its support for other OSs is not checked; besides, not all the VIMs support cloud-init configuration through their metadata agent. Fortunately, OpenStack Stein and DevStack support this configuration method. Furthermore, despite its usefulness, it is not OSM's "gold bullet" as much of the applications configuration will be carried out by the Day-1 configuration concept (explained in the next Test).

4.2.1 Test Characteristics

This section provides a summary of the current Test Characteristics.

Test 2	
Scenario	Testbed
VNFs	2
VNFDs	2
NSs	1
NSDs	1
PNFs	0
PNFDs	0
NSTs	0
Networks	2
Total RAM	4 GB
Total vCPUS	2

Total Disk	40 GB
Day-0 Config	Yes
Day-1 Config	None
Day-2 Config	None

Table 4-4. Test 2 summary.

As it can be seen, this scenario leverages the past scenario characteristics from a resource point of view (network, RAM, disk…) and augments the configuration complexity adding Day-0 configuration to both VNFs.

4.2.2 Test Description

In this test two identical VNFs are generated except for the Day-0 configuration which, in one of them allows the default user to log in using ssh-password and in the other one, instead, uses an ssh-keypair to allow a new user to log-in (later on, more details will be given). Furthermore, Internet access was desired in this Test in order to manage package updates/upgrades but also, it was desired to isolate this management traffic from the traffic interchanged by both machines. To achieve this objective, two networks were created: the first one, the management network is attached to the public network of the DevStack environment giving access to the VNFs and, the second one, is a private internal network.

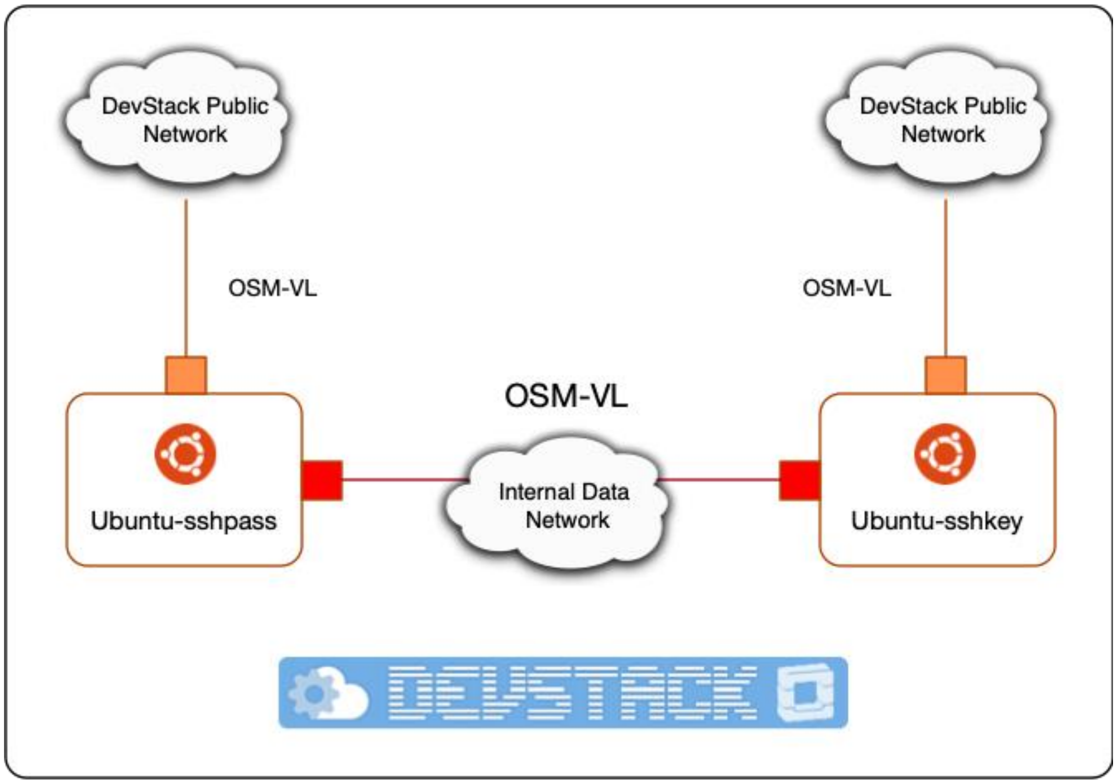
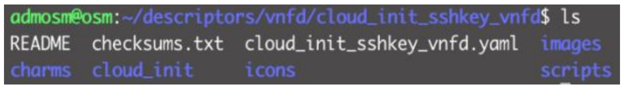
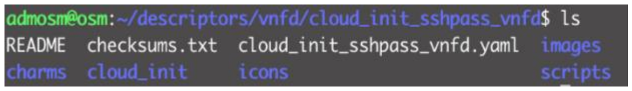


Figure 4-12. Test 2 High Level view.

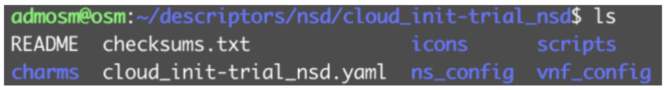
To create the file structure of both VNFs and the Network Service the third method, DevOps tools, has been used. Once the three commands were executed, the following directories were obtained:



```
admosm@osm:~/descriptors/vnfd/cloud_init_sshkey_vnfd$ ls
README  checksums.txt  cloud_init_sshkey_vnfd.yaml  images
charms  cloud_init     icons                        scripts
```



```
admosm@osm:~/descriptors/vnfd/cloud_init_sshpass_vnfd$ ls
README  checksums.txt  cloud_init_sshpass_vnfd.yaml  images
charms  cloud_init     icons                         scripts
```



```
admosm@osm:~/descriptors/nsd/cloud_init-trial_nsd$ ls
README  checksums.txt             icons      scripts
charms  cloud_init-trial_nsd.yaml ns_config  vnf_config
```

Figure 4-13. Test 2 VNFDs/NSDs file structure.

As it can be seen, using this method generates a slightly different file structure than the one generated by hand, it includes, in the case of the VNFs a new directory called "*cloud_init*". This directory is where the cloud-init config file must be located. From now on, as the file structure is a generic template, it is not going to be shown anymore.

Once the correct directories have been generated it remains to edit each of the VNF/NS descriptors. In this test, only the new functionalities are going to be explained with respect to the previous test's VNFD/NSD. To read the full descriptors see *Annexes 11.2 11.3 12.2*.

In the VNFDs a new metadata field has been added in order to upload cloud-init configuration to each VNF:

```
vdu:

        -   cloud-init-file: cloud_init_sshpass.cfg
```

Text Box 4-10. Cloud-init-file metadata field.

Other than that, no differences are remarkable except for the existence of another interface which allows to fulfill the objective of having one management dedicated interface and one internal-data interface.

On the other hand, in the NSD, the most remarkable change other than creating two networks is the fact that one of them is going to use OpenStack's public network as the management network, this is accomplished adding the following metadata fields:

```
vld:

        -   id: mgmt_net

            …

            mgmt-network: 'true'

            vim-network-name: public
```

Text Box 4-11. VIM network metadata field.

After all the descriptor editing process was done, the next step was to define two cloud-init configuration files in order to fulfill the following requirements:

- **"sshpass" VM:** Its hostname must be changed to "sshpass", the default's username should be changed to "sysadm" and login by ssh with a password should be permitted for this user. Besides, the "*/et/resolv.conf*" file should be edited to add, at least, two nameservers and a final message reporting the total instantiation time, should be prompted.
- **"sshkey" VM:** Its hostname must be changed to "sshkey", a new user with administrator privileges should be generated called "sysadm" and with the capability to log by ssh using a key-pair. Moreover, a mock file will be written to the "*/root*" directory in order to show how a pre-conf file may be generated and, additionally, the same "*etc/resolv.conf*" and final message requirements than the other VM are requested here.

Below, the cloud-init files that fulfill these conditions are shown:

```
#cloud-config
hostname: sshpass
system_info:
  default_user:
    name: sysadm
password: 5gvinni2019
chpasswd: { expire: False }
ssh_pwauth: True


manage_resolv_conf: true
resolv_conf:
  nameservers: ['4.4.4.4', '8.8.8.8']


final_message: "The system is finally up, after $UPTIME seconds"
```

Text Box 4-12. Test 2 sshpass cloud-init file.

```
#cloud-config

hostname: sshkey

users:

  - name: sysadm

    ssh-authorized-keys:

      -                                                                      ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDq9x0cn+psvfS1kIWq9JLPvlC8xowIzoPi/ndKgnZlwCw1QLeSla1
BVwBMB7I8cY98h/K2+1IeKtSx0G8cFM0KDArA/A6T2UmnYSYNTWO3fFqAGQP0fPo4IKXGl+ewbWL6AlbH2b
45Fbh7KrbcLMizhkpkSzK6XFIyWo8xI35LjVwtptDh40IMXsDlJXMkTTn4unIlIHTr9ZeaW8/sqYYEJe4xQ
SjIY0eUTIub5w8JgVw+XXuHnGtr2P+2FcxAkDgbFLO3Dgsg7fs80+6CAhKo+mB79cpkrXXGmW/CQPKWXUbZ
XGwslSpbmT06pFOU2Mm243rnre1OuJLVG0zmAfEF98lFbfymcGzI1W86s6wKqz4uPP+tbtgvGALD7Flmi3H
zcsuvLCsRCFOqJE9OUX8ooUSr8ye1Znc/HqLrImRdqDPebm53N0j0g9Wz/aU09LH/gDDjyCUY6nm/xDeiPW
eQm0BM5Y8xJOziDr9fVE85XAJEe9smTOsfsxhJ9eqhldIVYH4+RhI/M7u44HptSprG2GFkoP70MrlIOS2eu
Pc3KNCkFgT9K6IM0v9rCy1dIbG4jUhV+gLGicKBXhJuZ0OU3uPLRxR33xHCHXOrYvcg1prYkkAkoMY4CISg
1cwwkNuq4b+euf9P6Djj+nCA9/R3MMaVyjaXj/79uCQF9kTJ2Q==      galledoh@Adrians-MacBook-
Pro.local

    sudo: ['ALL=(ALL) NOPASSWD:ALL']

    shell: /bin/bash

    groups: sudo


write_files:

  - content: |

        # Hello Hello, I don't know why you say goddbye I say Hello

    fileowner: root:root

    permissions: '0644'

    path: /root/helloworld.txt


manage_resolv_conf: true

resolv_conf:

  nameservers: ['4.4.4.4', '8.8.8.8']


final_message: "The system is finally up, after $UPTIME seconds"
```

Text Box 4-13. Test 2 sshkey cloud-init file.

Now that every field has been explained and that both cloud-init files are known and understandable, OSM graph representation is presented:

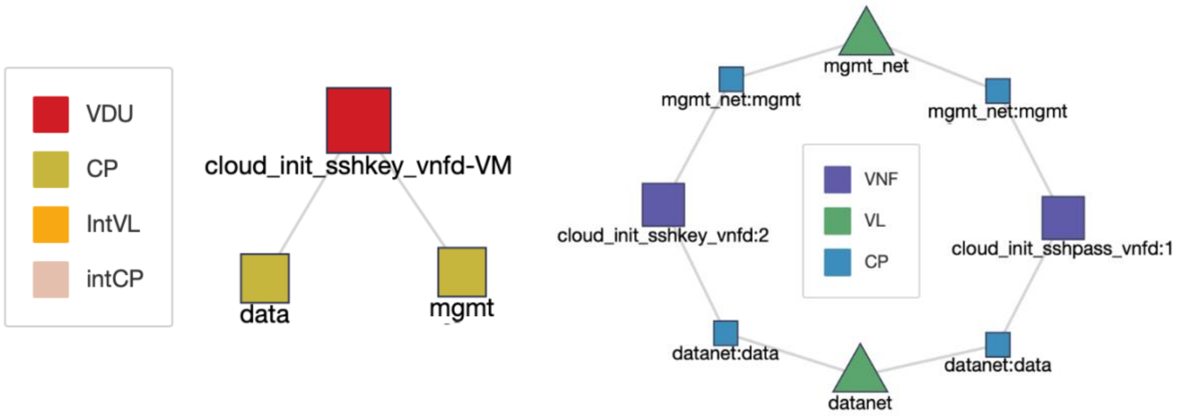
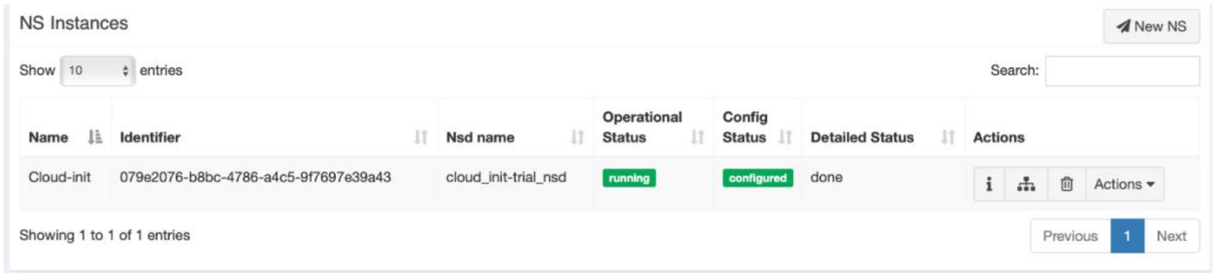


Figure 4-14. Test 2 OSM graphs.

4.2.3 Test Results

In this section, the instantiation process and correct behavior of the described Network Service are checked.

The first step consists on checking the correct instantiation from OSM to OpenStack, this mean that no RO errors are encountered:



NS Instances

New NS

Show 10 entries

Search:

Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
Cloud-init	079e2076-b8bc-4786-a4c5-9f7697e39a43	cloud_init-trial_nsd	running	configured	done	Actions

Showing 1 to 1 of 1 entries

Previous 1 Next

Figure 4-15. Test 2 RO Instantiation check.

The process went on as expected, with no errors.

The next step is to check that the correct network topology has been deployed and that in each of the VMs' OpenStack instantiation logs, the respective cloud-init config is reflected:

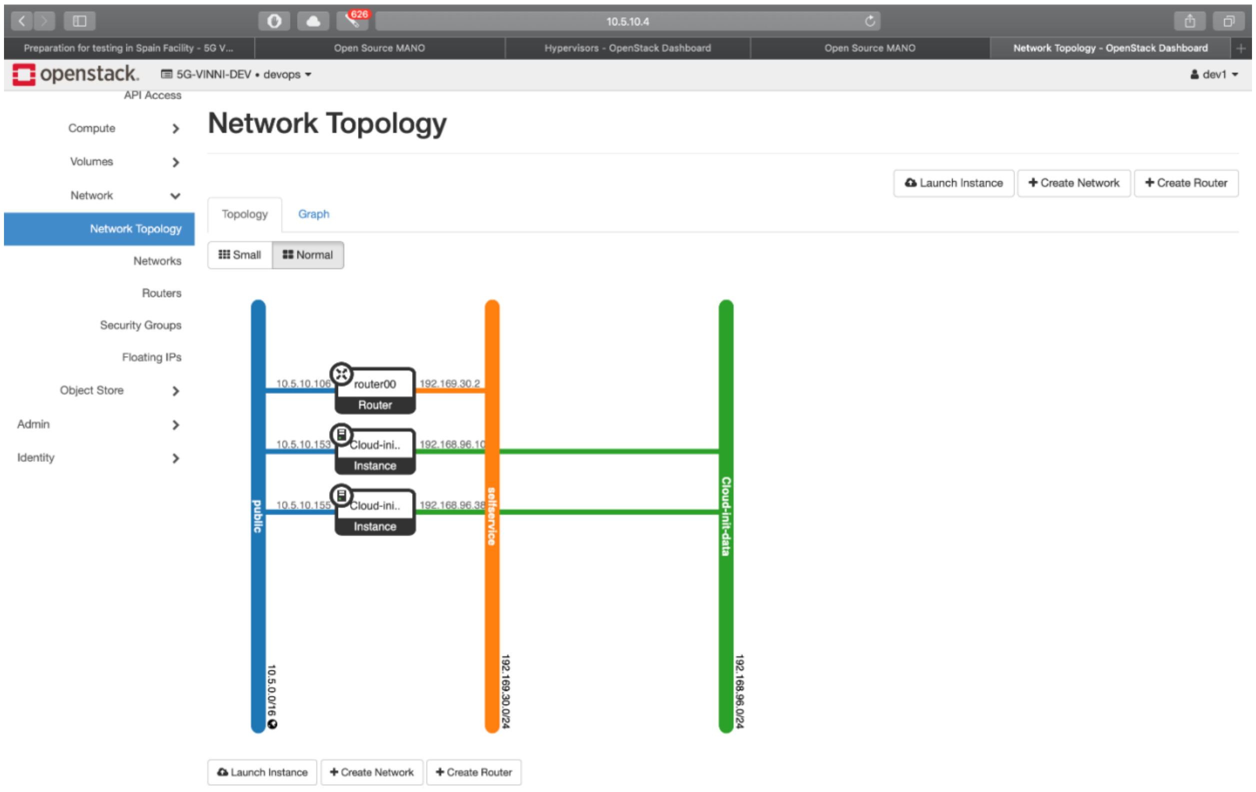
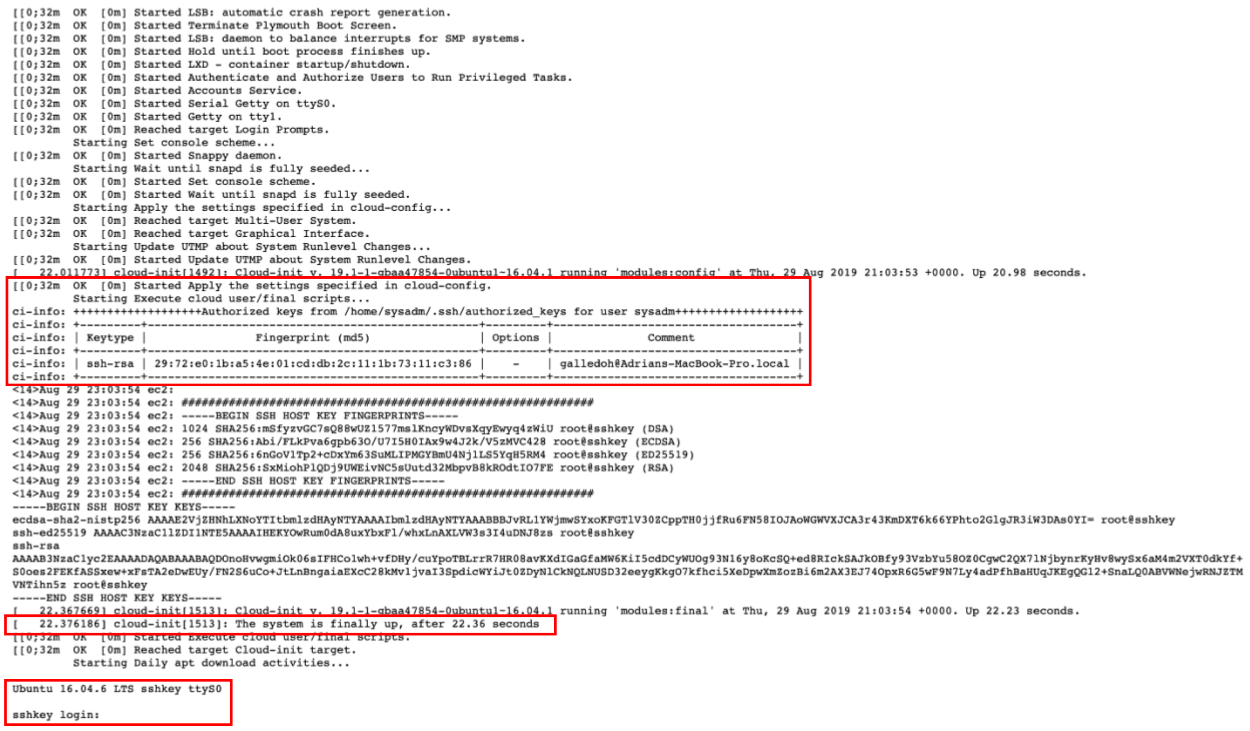


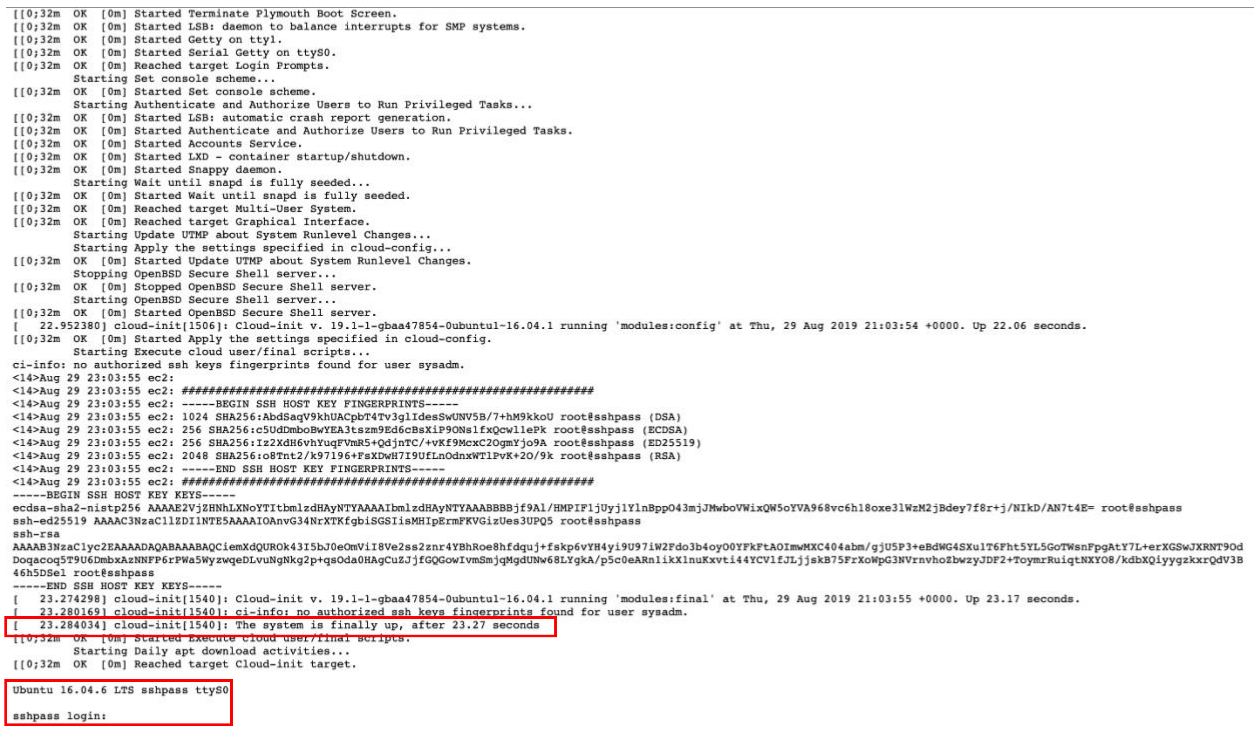
Figure 4-16. Test 2 OpenStack Network Topology.



```
[[0;32m  OK  [0m] Started LSB: automatic crash report generation.
[[0;32m  OK  [0m] Started Terminate Plymouth Boot Screen.
[[0;32m  OK  [0m] Started LSB: daemon to balance interrupts for SMP systems.
[[0;32m  OK  [0m] Started Hold until boot process finishes up.
[[0;32m  OK  [0m] Started LXD - container startup/shutdown.
[[0;32m  OK  [0m] Started Authenticate and Authorize Users to Run Privileged Tasks.
[[0;32m  OK  [0m] Started Accounts Service.
[[0;32m  OK  [0m] Started Serial Getty on ttyS0.
[[0;32m  OK  [0m] Started Getty on tty1.
[[0;32m  OK  [0m] Reached target Login Prompts.
         Starting Set console scheme...
[[0;32m  OK  [0m] Started Snappy daemon.
         Starting Wait until snapd is fully seeded...
[[0;32m  OK  [0m] Started Set console scheme.
[[0;32m  OK  [0m] Started Wait until snapd is fully seeded.
         Starting Apply the settings specified in cloud-config...
[[0;32m  OK  [0m] Reached target Multi-User System.
[[0;32m  OK  [0m] Reached target Graphical Interface.
         Starting Update UTMP about System Runlevel Changes...
[[0;32m  OK  [0m] Started Update UTMP about System Runlevel Changes.
[   22.011773] cloud-init[1492]: Cloud-init v. 19.1-1-gbaa47854-0ubuntu1-16.04.1 running 'modules:config' at Thu, 29 Aug 2019 21:03:53 +0000. Up 20.98 seconds.
[[0;32m  OK  [0m] Started Apply the settings specified in cloud-config.
         Starting Execute cloud user/final scripts...
ci-info: ++++++++++++++++++Authorized keys from /home/sysadm/.ssh/authorized_keys for user sysadm++++++++++++++++++
ci-info: +---------+-------------------------------------------------+---------+-----------------------------------+
ci-info: | Keytype |                Fingerprint (md5)                | Options |              Comment              |
ci-info: +---------+-------------------------------------------------+---------+-----------------------------------+
ci-info: | ssh-rsa | 29:72:e0:1b:a5:4e:01:cd:db:2c:11:1b:73:11:c3:86 |    -    | galledoh@Adrians-MacBook-Pro.local |
ci-info: +---------+-------------------------------------------------+---------+-----------------------------------+
<14>Aug 29 23:03:54 ec2: 
<14>Aug 29 23:03:54 ec2: #############################################################
<14>Aug 29 23:03:54 ec2: -----BEGIN SSH HOST KEY FINGERPRINTS-----
<14>Aug 29 23:03:54 ec2: 1024 SHA256:mSfyzvGC7sQ88wUZ1577mslKncyWDvsXqyEwyq4zWiU root@sshkey (DSA)
<14>Aug 29 23:03:54 ec2: 256 SHA256:Abi/FLkPva6gpb63O/U7I5H0IAx9w4J2k/V5zMVC428 root@sshkey (ECDSA)
<14>Aug 29 23:03:54 ec2: 256 SHA256:6nGoV1Tp2+cDxYm63SuMLIPMGYBmU4Nj1LS5YqH5RM4 root@sshkey (ED25519)
<14>Aug 29 23:03:54 ec2: 2048 SHA256:SxMiohPlQDj9UWEivNC5sUutd32MbpvB8kROdtIO7FE root@sshkey (RSA)
<14>Aug 29 23:03:54 ec2: -----END SSH HOST KEY FINGERPRINTS-----
<14>Aug 29 23:03:54 ec2: #############################################################
-----BEGIN SSH HOST KEY KEYS-----
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBJvRL1YWjmwSYxoKFGTlV30ZCppTH0jjfRu6FN58IOJAoWGWVXJCA3r43KmDXT6k66YPhto2GlgJR3iW3DAs0YI= root@sshkey
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIHEKYOwRum0dA8uxYbxFl/whxLnAXLVW3s3I4uDNJ8zs root@sshkey
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDOnoHvwgmiOk06sIFHColwh+vfDHy/cuYpoTBLrrR7HR08avKXdIGaGfaMW6KiI5cdDCyWUOg93Nl6y8oKcSQ+ed8RIckSAJkOBfy93VzbYu58OZ0CgwC2QX7lNjbynrKyHv8wySx6aM4m2VXT0dkYf+
S0oes2FEKfASSxew+xFsTA2eDwEUy/FN2S6uCo+JtLnBngaiaEXcC28kMvljvaI3SpdicWYiJt0ZDyNlCkNQLNUSD32eeygKkgO7kfhci5XeDpwXmZozBi6m2AX3EJ74OpxR6G5wF9N7Ly4adPfhBaHUqJKEgQGl2+SnaLQ0ABVWNejwRNJZTM
VNTihn5z root@sshkey
-----END SSH HOST KEY KEYS-----
[   22.367669] cloud-init[1513]: Cloud-init v. 19.1-1-gbaa47854-0ubuntu1-16.04.1 running 'modules:final' at Thu, 29 Aug 2019 21:03:54 +0000. Up 22.23 seconds.
[   22.376186] cloud-init[1513]: The system is finally up, after 22.36 seconds
[[0;32m  OK  [0m] Started Execute cloud user/final scripts.
[[0;32m  OK  [0m] Reached target Cloud-init target.
         Starting Daily apt download activities...

Ubuntu 16.04.6 LTS sshkey ttyS0

sshkey login:
```

Figure 4-17. Test 2 sshkey VNF log.

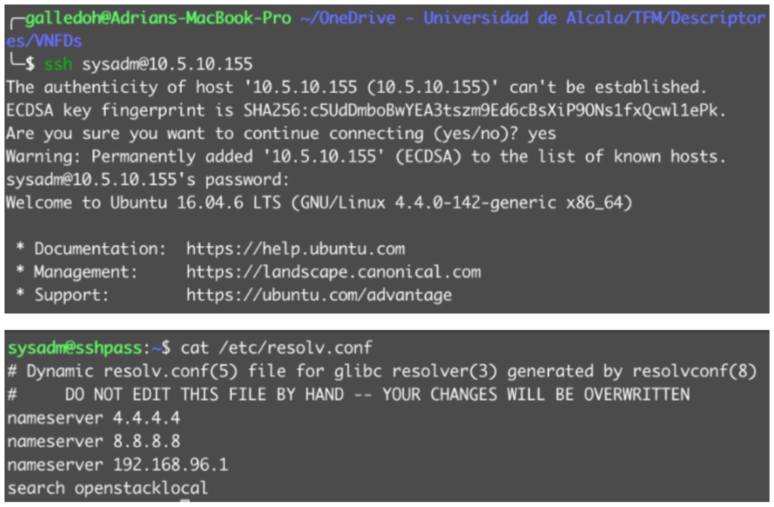
Three log sentences have been remarked: the first one shows that the public ssh-key has been added correctly, the second one indicates the final message with the time that took the machine to be instantiated and, finally, the third one demonstrates that the hostname of this machine is "sshkey".



```
[[0;32m  OK  [0m] Started Terminate Plymouth Boot Screen.
[[0;32m  OK  [0m] Started LSB: daemon to balance interrupts for SMP systems.
[[0;32m  OK  [0m] Started Getty on tty1.
[[0;32m  OK  [0m] Started Serial Getty on ttyS0.
[[0;32m  OK  [0m] Reached target Login Prompts.
         Starting Set console scheme...
[[0;32m  OK  [0m] Started Set console scheme.
         Starting Authenticate and Authorize Users to Run Privileged Tasks...
[[0;32m  OK  [0m] Started LSB: automatic crash report generation.
[[0;32m  OK  [0m] Started Authenticate and Authorize Users to Run Privileged Tasks.
[[0;32m  OK  [0m] Started Accounts Service.
[[0;32m  OK  [0m] Started LXD - container startup/shutdown.
[[0;32m  OK  [0m] Started Snappy daemon.
         Starting Wait until snapd is fully seeded...
[[0;32m  OK  [0m] Started Wait until snapd is fully seeded.
[[0;32m  OK  [0m] Reached target Multi-User System.
[[0;32m  OK  [0m] Reached target Graphical Interface.
         Starting Update UTMP about System Runlevel Changes...
         Starting Apply the settings specified in cloud-config...
[[0;32m  OK  [0m] Started Update UTMP about System Runlevel Changes.
         Stopping OpenBSD Secure Shell server...
[[0;32m  OK  [0m] Stopped OpenBSD Secure Shell server.
         Starting OpenBSD Secure Shell server...
[[0;32m  OK  [0m] Started OpenBSD Secure Shell server.
[   22.952380] cloud-init[1506]: Cloud-init v. 19.1-1-gbaa47854-0ubuntu1-16.04.1 running 'modules:config' at Thu, 29 Aug 2019 21:03:54 +0000. Up 22.06 seconds.
[[0;32m  OK  [0m] Started Apply the settings specified in cloud-config.
         Starting Execute cloud user/final scripts...
ci-info: no authorized ssh keys fingerprints found for user sysadm.
<14>Aug 29 23:03:55 ec2:
<14>Aug 29 23:03:55 ec2: #############################################################
<14>Aug 29 23:03:55 ec2: -----BEGIN SSH HOST KEY FINGERPRINTS-----
<14>Aug 29 23:03:55 ec2: 1024 SHA256:AbdSaqV9khUACpbT4Tv3glIdesSwUNV5B/7+hM9kkoU root@sshpass (DSA)
<14>Aug 29 23:03:55 ec2: 256 SHA256:c5UdDmboBwYEA3tszm9Ed6cBsXiP9ONs1fxQcwl1ePk root@sshpass (ECDSA)
<14>Aug 29 23:03:55 ec2: 256 SHA256:Iz2XdH6vhYuqFVmR5+QdjnTC/+vKf9McxC2OgmYjo9A root@sshpass (ED25519)
<14>Aug 29 23:03:55 ec2: 2048 SHA256:o8Tnt2/k97196+FsXDwH7I9UfLnOdnxWTlPvK+2O/9k root@sshpass (RSA)
<14>Aug 29 23:03:55 ec2: -----END SSH HOST KEY FINGERPRINTS-----
<14>Aug 29 23:03:55 ec2: #############################################################
-----BEGIN SSH HOST KEY KEYS-----
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBBjf9Al/HMPIF1jUyj1YlnBppO43mjJMwboVWixQW5oYVA968vc6hl8oxe3lWzM2jBdey7f8r+j/NIkD/AN7t4E= root@sshpass
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOAnvG34NrXTKfgbiSGSIisMHIpErmFKVGizUes3UPQ5 root@sshpass
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCiemXdQUROk43I5bJ0eOmViI8Ve2ss2znr4YBhRoe8hfdquj+fskp6vYH4yi9U97iW2Fdo3b4oyO0YFkFtAOImwMXC404abm/gjU5P3+eBdWG4SXu1T6Fht5YL5GoTWsnFpgAtY7L+erXGSwJXRNT9Od
Doqacoq5T9U6DmbxAzNNFP6rPWa5WyzwqeDLvuNgNkg2p+qsOda0HAgCuZJjfGQGowIvmSmjqMgdUNw68LYgkA/p5c0eARnlikX1nuKxvti44YCVlfJLjjskB75FrXoWpG3NVrnvhoZbwzyJDF2+ToymrRuiqtNXYO8/kdbXQiyygzkxrQdV3B
46h5DSel root@sshpass
-----END SSH HOST KEY KEYS-----
[   23.274298] cloud-init[1540]: Cloud-init v. 19.1-1-gbaa47854-0ubuntu1-16.04.1 running 'modules:final' at Thu, 29 Aug 2019 21:03:55 +0000. Up 23.17 seconds.
[   23.280169] cloud-init[1540]: ci-info: no authorized ssh keys fingerprints found for user sysadm.
[   23.284034] cloud-init[1540]: The system is finally up, after 23.27 seconds
[[0;32m  OK  [0m] Started Execute cloud user/final scripts.
         Starting Daily apt download activities...
[[0;32m  OK  [0m] Reached target Cloud-init target.

Ubuntu 16.04.6 LTS sshpass ttyS0

sshpass login:
```

Figure 4-18. Test 2 sshpass VNF log.

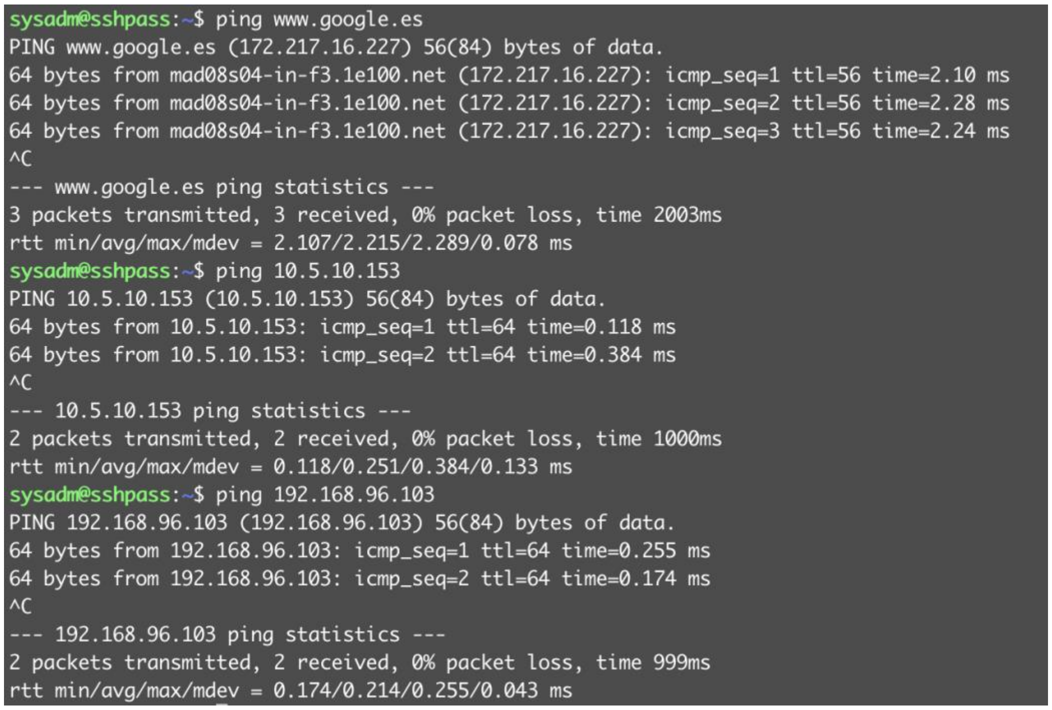
After this first configuration has been verified, in order to check the rest of the desired features, it is mandatory to login inside each VNF. Firstly, "sshpass" VNF is checked:



```
┌─galledoh@Adrians-MacBook-Pro ~/OneDrive - Universidad de Alcala/TFM/Descriptor
es/VNFDs
└─$ ssh sysadm@10.5.10.155
The authenticity of host '10.5.10.155 (10.5.10.155)' can't be established.
ECDSA key fingerprint is SHA256:c5UdDmboBwYEA3tszm9Ed6cBsXiP9ONs1fxQcwl1ePk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.5.10.155' (ECDSA) to the list of known hosts.
sysadm@10.5.10.155's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-142-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
```

```
sysadm@sshpass:~$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 4.4.4.4
nameserver 8.8.8.8
nameserver 192.168.96.1
search openstacklocal
```

Figure 4-19. Test 2 sshpass VNF access and DNS conf check.

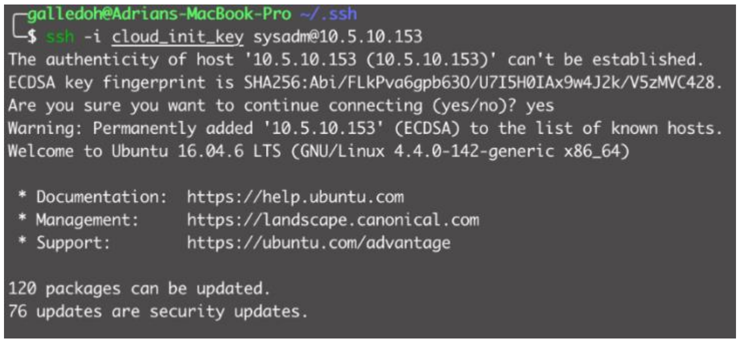


```
sysadm@sshpass:~$ ping www.google.es
PING www.google.es (172.217.16.227) 56(84) bytes of data.
64 bytes from mad08s04-in-f3.1e100.net (172.217.16.227): icmp_seq=1 ttl=56 time=2.10 ms
64 bytes from mad08s04-in-f3.1e100.net (172.217.16.227): icmp_seq=2 ttl=56 time=2.28 ms
64 bytes from mad08s04-in-f3.1e100.net (172.217.16.227): icmp_seq=3 ttl=56 time=2.24 ms
^C
--- www.google.es ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 2.107/2.215/2.289/0.078 ms
sysadm@sshpass:~$ ping 10.5.10.153
PING 10.5.10.153 (10.5.10.153) 56(84) bytes of data.
64 bytes from 10.5.10.153: icmp_seq=1 ttl=64 time=0.118 ms
64 bytes from 10.5.10.153: icmp_seq=2 ttl=64 time=0.384 ms
^C
--- 10.5.10.153 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.118/0.251/0.384/0.133 ms
sysadm@sshpass:~$ ping 192.168.96.103
PING 192.168.96.103 (192.168.96.103) 56(84) bytes of data.
64 bytes from 192.168.96.103: icmp_seq=1 ttl=64 time=0.255 ms
64 bytes from 192.168.96.103: icmp_seq=2 ttl=64 time=0.174 ms
^C
--- 192.168.96.103 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.174/0.214/0.255/0.043 ms
```

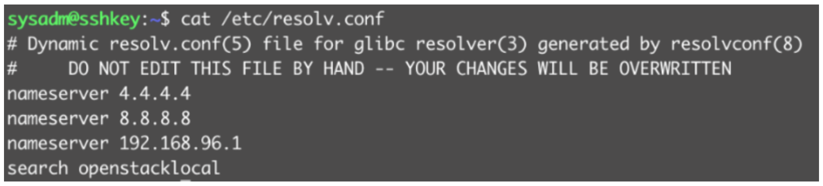
Figure 4-20. Test 2 sshpass VNF network connections check.

Everything went as expected.

Same checks are carried out for the second VNF and an extra test to check the creation of the */root/helloworld.txt"* file:

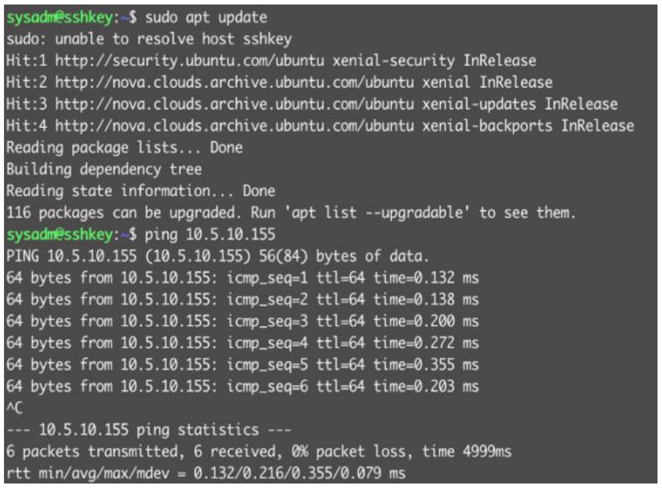


```
┌galledoh@Adrians-MacBook-Pro ~/.ssh
└$ ssh -i cloud_init_key sysadm@10.5.10.153
The authenticity of host '10.5.10.153 (10.5.10.153)' can't be established.
ECDSA key fingerprint is SHA256:Abi/FLkPva6gpb630/U7I5H0IAx9w4J2k/V5zMVC428.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.5.10.153' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-142-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

120 packages can be updated.
76 updates are security updates.
```

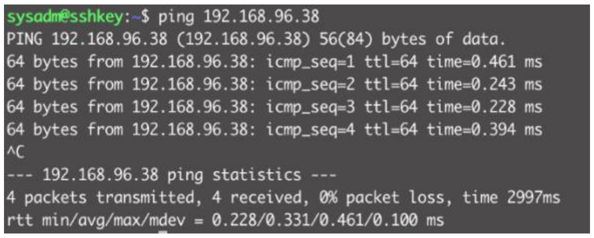


```
sysadm@sshkey:~$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 4.4.4.4
nameserver 8.8.8.8
nameserver 192.168.96.1
search openstacklocal
```

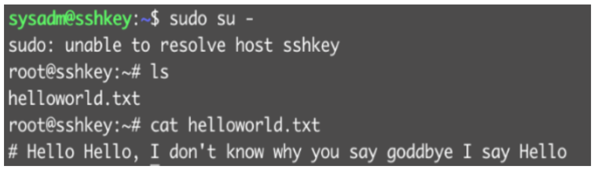
Figure 4-21. Test 2 sshkey VNF access and DNS conf check.



```
sysadm@sshkey:~$ sudo apt update
sudo: unable to resolve host sshkey
Hit:1 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:2 http://nova.clouds.archive.ubuntu.com/ubuntu xenial InRelease
Hit:3 http://nova.clouds.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:4 http://nova.clouds.archive.ubuntu.com/ubuntu xenial-backports InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
116 packages can be upgraded. Run 'apt list --upgradable' to see them.
sysadm@sshkey:~$ ping 10.5.10.155
PING 10.5.10.155 (10.5.10.155) 56(84) bytes of data.
64 bytes from 10.5.10.155: icmp_seq=1 ttl=64 time=0.132 ms
64 bytes from 10.5.10.155: icmp_seq=2 ttl=64 time=0.138 ms
64 bytes from 10.5.10.155: icmp_seq=3 ttl=64 time=0.200 ms
64 bytes from 10.5.10.155: icmp_seq=4 ttl=64 time=0.272 ms
64 bytes from 10.5.10.155: icmp_seq=5 ttl=64 time=0.355 ms
64 bytes from 10.5.10.155: icmp_seq=6 ttl=64 time=0.203 ms
^C
--- 10.5.10.155 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4999ms
rtt min/avg/max/mdev = 0.132/0.216/0.355/0.079 ms
```



```
sysadm@sshkey:~$ ping 192.168.96.38
PING 192.168.96.38 (192.168.96.38) 56(84) bytes of data.
64 bytes from 192.168.96.38: icmp_seq=1 ttl=64 time=0.461 ms
64 bytes from 192.168.96.38: icmp_seq=2 ttl=64 time=0.243 ms
64 bytes from 192.168.96.38: icmp_seq=3 ttl=64 time=0.228 ms
64 bytes from 192.168.96.38: icmp_seq=4 ttl=64 time=0.394 ms
^C
--- 192.168.96.38 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.228/0.331/0.461/0.100 ms
```



```
sysadm@sshkey:~$ sudo su -
sudo: unable to resolve host sshkey
root@sshkey:~# ls
helloworld.txt
root@sshkey:~# cat helloworld.txt
# Hello Hello, I don't know why you say goddbye I say Hello
```

Figure 4-22. Test 2 sshkey VNF network connection and file check.

From all the screenshots in *Figure 4-22. Test 2 sshkey VNF network connection and file check.*, it can be concluded that the test was satisfactory, in fact, finalizing it without any errors supposes a great step towards the development of VNFs with Day-1 configuration as they require the usage of a user with ssh password authorization in every VNF (to run the proper commands). It is important to remark that, although a couple of things can be accomplished using Day-0 configuration, most of them are easily done with Day-1 configuration and, consequently, from now on only the ssh-password/key authorization and hostname editing features will be used for Day-0 configuration purposes.

4.3 Test 3: Day-1 Config

This test is, probably, the most important one towards the final objective as it is the one oriented towards various OSM methods to automate applications installation and configuration in the VNFs. This automation process is known as Day-1 configuration in OSM and it is done, mainly, using *Juju-Charms* and the *Juju* framework.

Juju is an Open source next generation service orchestration framework, composed by a controller, models and charms, that can handle service configuration, services relationships, scaling and LCM. Charms are no more than a set of reactive scripts that allow to deploy and run different kinds of SW, it is important to remark that every charm is organized in layers so that code can be reused (later on a layer will be created). OSM Rel. FIVE does not support *juju-charms* full capabilities so it integrates the so-called *proxy charms*, which are pretty like *juju-charms,* but they map configurations to juju-actions via ssh or RESTful APIs.

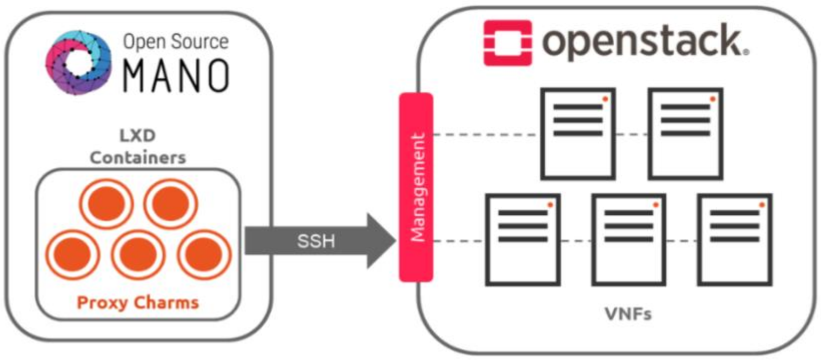


Figure 4-23. OSM's proxy charms.

Explained lightly, what occurs is that OSM's Service Orchestrator (SO) commands the RO to create a VM, when this process is over (the VM is instantiated) the SO will create an LXD container, managed by juju, with the *proxy charm* and this *proxy charms* will send the VM the proper configuration orders to carry out the Day-1 configuration:

```
+---------------------<----+---------------------+
|  Resource           |    |  Service            |
|  Orchestrator (RO)  +---->  Orchestrator (SO)  |
|                     |    |                     |
+-----------------+---+    +--------+----^--------+
                  |                 |    |
                  |                 |    |
            +-----v-----+<------+-v----+--+
            |  Virtual  |       |  Proxy  |
            |  Machine  |       |  Charm  |
            +-----------+------->---------+
```

Text Box 4-14. OSM Rel. FIVE Day-1 configuration process schema.

In summary, VNF configuration is developed through three different stages or "days":

- **Day-0:** Studied in the past test, is carried out when the machine is not completely ready, at the instantiation time. E.g. Create users/groups, generate pre-configuration files, etc.
- **Day-**1: Studied in this test, it is used when the machine is ready and running, this means that it is application-oriented. E.g. Command execution, update/upgrade packages, edit configuration files, etc.
- **Day-2:** Also mentioned in this section but slightly, as its usefulness in this project is not the best. Occurs once the machine is ready and configured, it allows to carry on-demand actions. E.g. Backups, dump logs, etc.

From these stages or "days", OSM's *juju-charms* covers the last two stages.

4.3.1 Test Characteristics

This section provides a summary of the current Test Characteristics.

Test 3	
Scenario	Testbed/Production 2
VNFs	1
VNFDs	1
NSs	1
NSDs	1
PNFs	0
PNFDs	0
NSTs	0
Networks	1
Total RAM	2 GB
Total vCPUS	1
Total Disk	20 GB
Day-0 Config	Yes
Day-1 Config	Yes
Day-2 Config	Yes

Table 4-5. Test 3 summary.

4.3.2 Test Description

Compared with the previous test, this one has a much simpler network configuration. Only one VNF is generated and this VNF is directly connected to OpenStack public network, through the VM management interface.

The main point in this test, as it had been mentioned is to understand the Day-1 config process and all the tools that imply its usage. Therefore, the first step is to understand what a juju-charm is, its main differences in respect to OSM's *proxy charms* and how does it work the layer structure of the least. The first two concepts have been explained in this same test introduction, consequently, here *proxy charms'* layers are detailed.

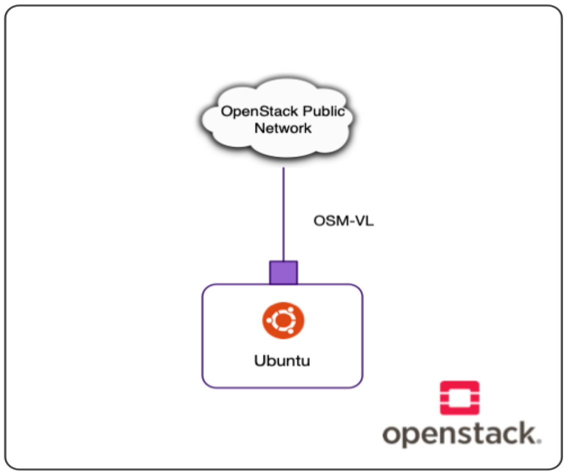


Figure 4-24. Test 3 High Level View.

A *juju-charm* layer is an individual component that, when combined with other layers, results in a unique charm. Layers are useful because they allow to reuse lots of code, especially when using proxy charms. Below is a figure that illustrates how a new OSM oriented charm is generated and how three layers are always used:

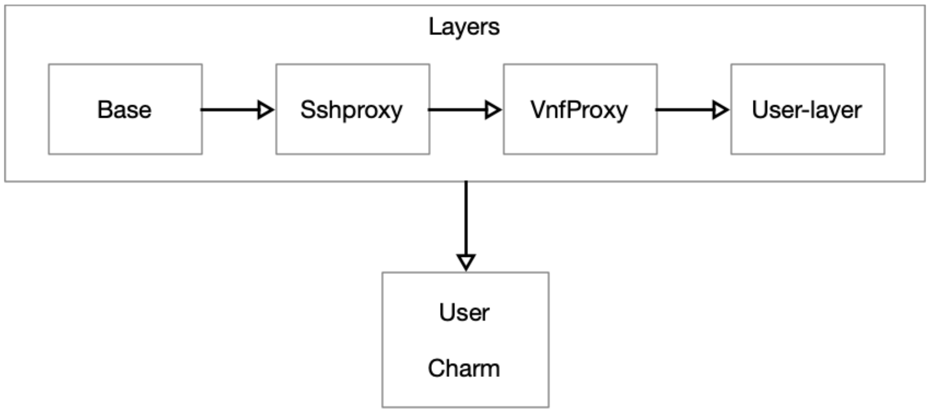


Figure 4-25. OSM charm layers.

Every single charm generated for OSM needs the above-shown layers:

- **Basic Layer:** Incorporates all the required SW to execute other layers.
- **sshproxy Layer:** Incorporates the required SW to manage ssh interchanges between the LXD container and the VM.
- **vnfproxy Layer:** Manages communication with the VNF and formats messages to be understood by OSM's RO and SO.

In this test, a mock charm layer that allows to create blank files in the VM is going to be developed in order to show the process of generating an OSM charm. However, direct proxy charm Day-1 configuration does not fit the requirements for an automated vEPC so, at the end of this section, another test is going to be implemented using an abstraction layer over proxy charms that allows to execute Day-1 configuration using "Ansible Playbooks".

To create a new OSM proxy charm is necessary to set up the proper environment inside the OSM machine, this is done with the following commands:

```
$ sudo snap install charm --classic # Charm building tool

$ mkdir -p ~/charms/layers

$ mkdir -p ~/charms/interfaces


# Add the following local variables to .bashrc file

$ export JUJU_REPOSITORY=~/charms

$ export CHARM_LAYERS_DIR=$JUJU_REPOSITORY/layers

$ export CHARM_INTERFACES_DIR=$JUJU_REPOSITORY/interfaces
```

Text Box 4-15. Juju charm environment set up.

The next step was to create a charm, in this case, it was called "simple":

```
$ cd JUJU_REPOSITORY/layers

$ charm create simple
```

Text Box 4-16. Charm generation.

These past commands will generate a directory with the following structure:

```
.
├── config.yaml
├── icon.svg
├── layer.yaml
├── metadata.yaml
├── reactive
│   └── simple.py
├── README.ex
└── tests
    ├── 00-setup
    └── 10-deploy
```

Text Box 4-17. Charm file structure.

Each file and directory are important and have their own functionality:

FILENAME	DESCRIPTION
LAYER.YAML	Addresses the layers that will compose the base for this new layer (always required basic and vnfproxy layers - the last one integrates the sshproxy layer).
METADATA.YAML	Contains the high-level information of the charm. E.g. Description, name, oriented OS, etc.
REACTIVE/SIMPLE.PY	Contains the reactive code that will define the behavior of the current charm.
ACTIONS.YAML	This file must be generated manually. It contains every action that the charm will be capable to perform.

Table 4-6. Charm files description.

Thereupon, as described in the table, it is necessary to create manually "actions.yaml" file in the root directory and edit it as follows:

```
touch:
  description: "Touch a file on the VNF."
  params:
    filename:
      description: "The name of the file to touch."
      type: string
      default: ""
  required:
  - filename
```

Text Box 4-18. "actions.yaml" charm file.

The above content tells the charm builder that this charm will be capable of developing an action called "touch" that requires a parameter, as a string, called "filename". Once this is done, the next step is to create a folder inside the root directory called "actions" and create a file inside it with the respective action (this file MUST have executing permissions). All the actions defined in any proxy charm use the same code, the actual behavior of each action is defined in the "/reactive" directory files. This code is shown in the next text-box:

```
#!/usr/bin/env python3

import sys

sys.path.append('lib')

from charms.reactive import main, set_state

from charmhelpers.core.hookenv import action_fail, action_name

set_state('actions.{}'.format(action_name()))

try:

    main()

except Exception as e:

    action_fail(repr(e))
```

Text Box 4-19. Actions generic code.

To end the charm generation process, the actual action behavior must be defined. This is done by appending, in this case, the following code inside the "*reactive/simple.py*" file:

```
@when('actions.touch')

def touch():

    """Touch a file."""

    err = ''

    try:

        filename = action_get('filename')

        cmd = ['touch {}'.format(filename)]

        result, err = charms.sshproxy._run(cmd)

    except:

        action_fail('command failed: {}'.format(err))

    else:

        action_set({'output': result})

    finally:

        clear_flag('actions.touch')
```

Text Box 4-20. Touch action behavior.

Finally, as all the required configuration has been done, it only remains to build the charm using the command "$ charm build" and copying its data to the "charms" directory of this test's VNF.

Furthermore, in order to use the newly created charm, additional configuration must be added to the VNFD descriptor:

```
vnf-configuration:
  juju:
    charm: simple
  initial-config-primitive:
  - seq: '1'
    name: config
    parameter:
    - name: ssh-hostname
      value: <rw_mgmt_ip>
    - name: ssh-username
      value: sysadm
    - name: ssh-password
      value: 5gvinni2019
  - seq: '2'
    name: touch
    parameter:
    - name: filename
      value: '/home/sysadm/test-file
```

Text Box 4-21. Test 3 VNFD additional metadata fields.

This new metadata fields request the VNF to use a charm called "simple" and it commands this charm to execute two actions: the first one allows the sshproxy layer to obtain the correct ssh credentials and the second one, generates a blank file named "*test-file*" in the *"home/sysadm"* directory inside the VM.

Moreover, although it is not going to be used in the rest of the project, a small Day-2 configuration was added to the VNFD through the following lines:

```
config-primitive:
- name: touch
  parameter:
  - name: filename
    data-type: STRING
    default-value: '/home/sysadm/touched'
```

Text Box 4-22. Test 3 VNFD Day-2 configuration.

These lines will allow any user to launch the touch action at any desired instant.

Additionally to this charm, as it does not really fit the requirements to automate a vEPC or, more clearly, it does but with a huge workload; it was decided to try a layer developed by the 5GinFIRE[61] project that adds an abstraction layer over the proxy charms and allows the usage of ansible playbooks. This layer has a lot of advantages: it works over a checked SW, it is an easiest approach as it is known by the author and does not require to generate, from scratch, an action for each of the required configuration processes needed to build up an entire vEPC with OAI.

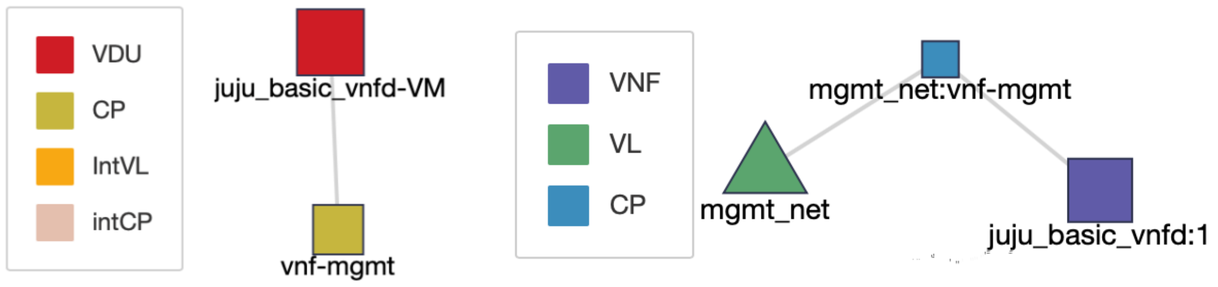


Figure 4-26. Test 3 VNFD/NSD graphs.

61 https://github.com/5GinFIRE/mano/tree/master/charms/ansible-charm

4.3.3 Test Results

4.3.3.1 Self-made proxy charm

In this section, the charm instantiation process and correct behavior of the described Network Service are checked. VNFD/NSD generation, uploading and launching processes will be ignored in this Test[62] as the process is the same than the one explained in the previous two tests. Consequently, the first verification is going to be OpenStack's network topology (only one network interface has been addressed it has to be connected to OpenStack's public provider network):

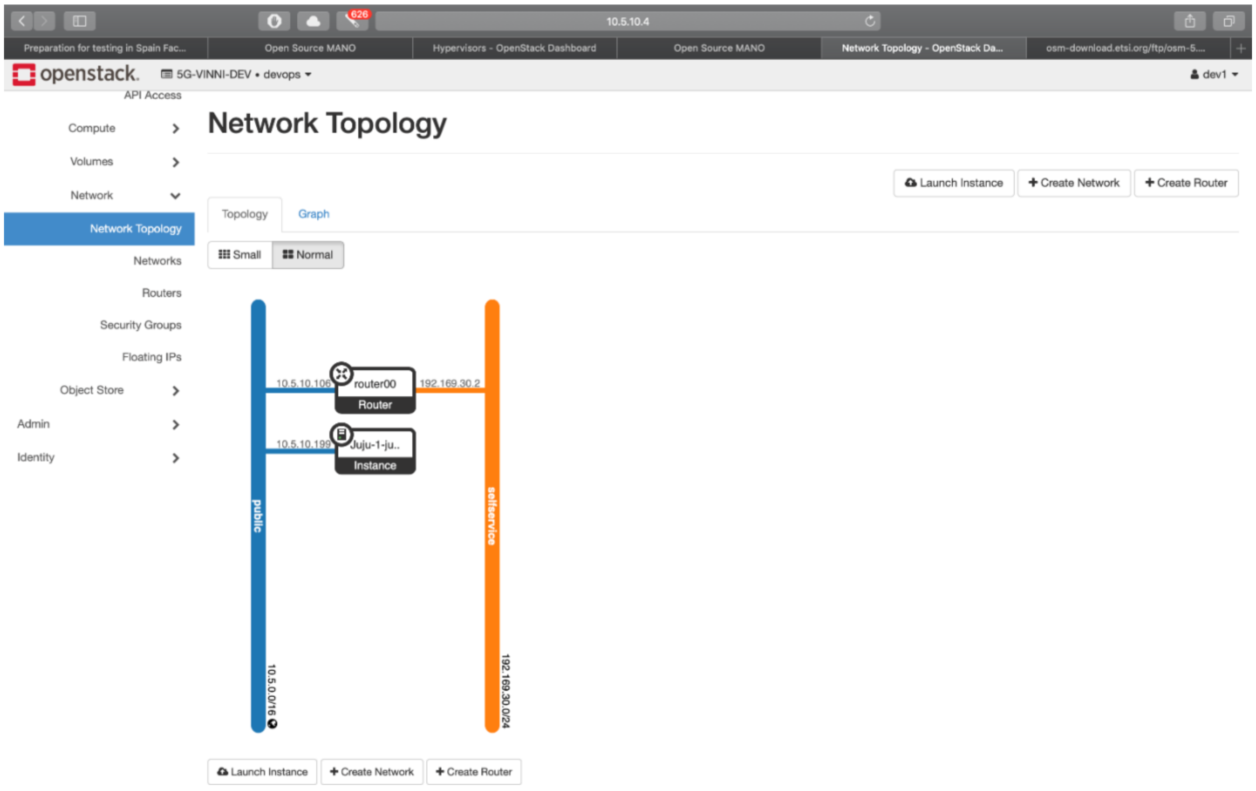
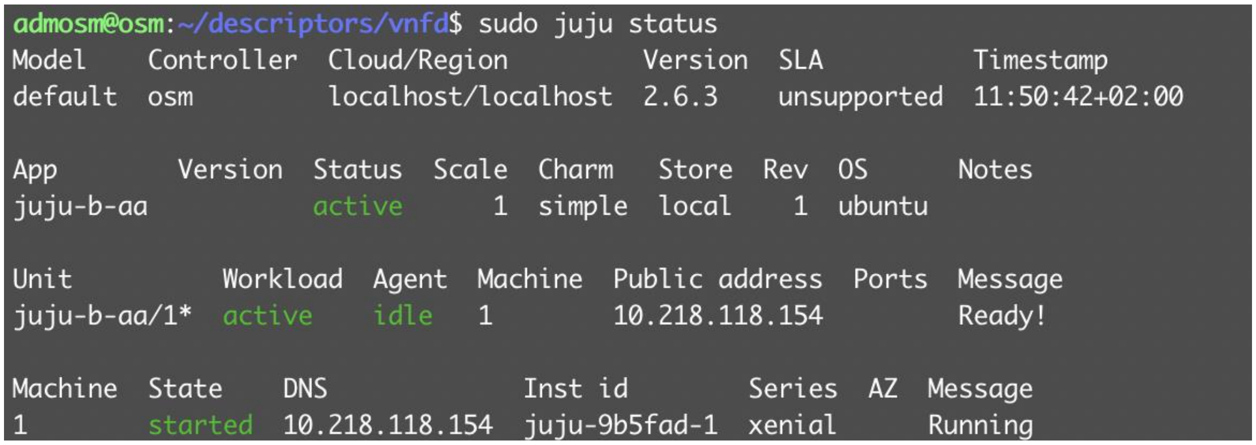


Figure 4-27. Test 3 OpenStack's Network Topology.

The next verification is located on the other side, the juju controller side. Once the VNF is completely instantiated, OSM SO sends an order to create a LXD container and from that container the commands for the Day-1 configuration are ssh-executed. To verify its correct state, the following command may be used:

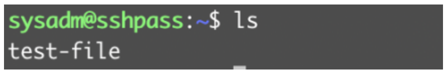
62 NOTE: For these two tests the cloud-init configuration file used, was the same than the one used in the "sshpass" VNF from the previous test.



```
admosm@osm:~/descriptors/vnfd$ sudo juju status
Model    Controller  Cloud/Region         Version  SLA          Timestamp
default  osm         localhost/localhost  2.6.3    unsupported  11:50:42+02:00

App        Version  Status  Scale  Charm   Store  Rev  OS      Notes
juju-b-aa           active      1  simple  local    1  ubuntu

Unit           Workload  Agent  Machine  Public address  Ports  Message
juju-b-aa/1*   active    idle   1        10.218.118.154         Ready!

Machine  State    DNS             Inst id        Series  AZ  Message
1        started  10.218.118.154  juju-9b5fad-1  xenial      Running
```

Figure 4-28. Test 3 Charms and LXD container status verification.

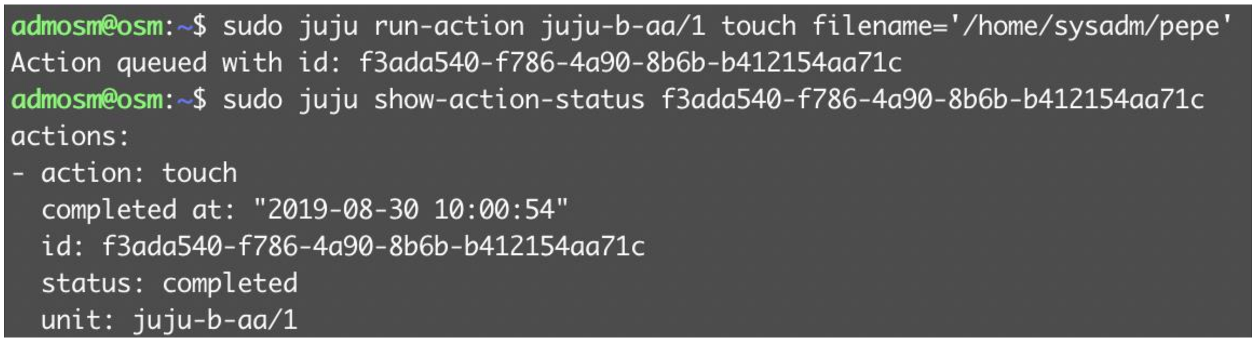
Thanks to the above figure, it can be observed that the creation of the LXD container and the execution of the "touch" action went as expected. To check if the blank file was created, login into the VM was required.



```
sysadm@sshpass:~$ ls
test-file
```

Figure 4-29. Touch action verification.

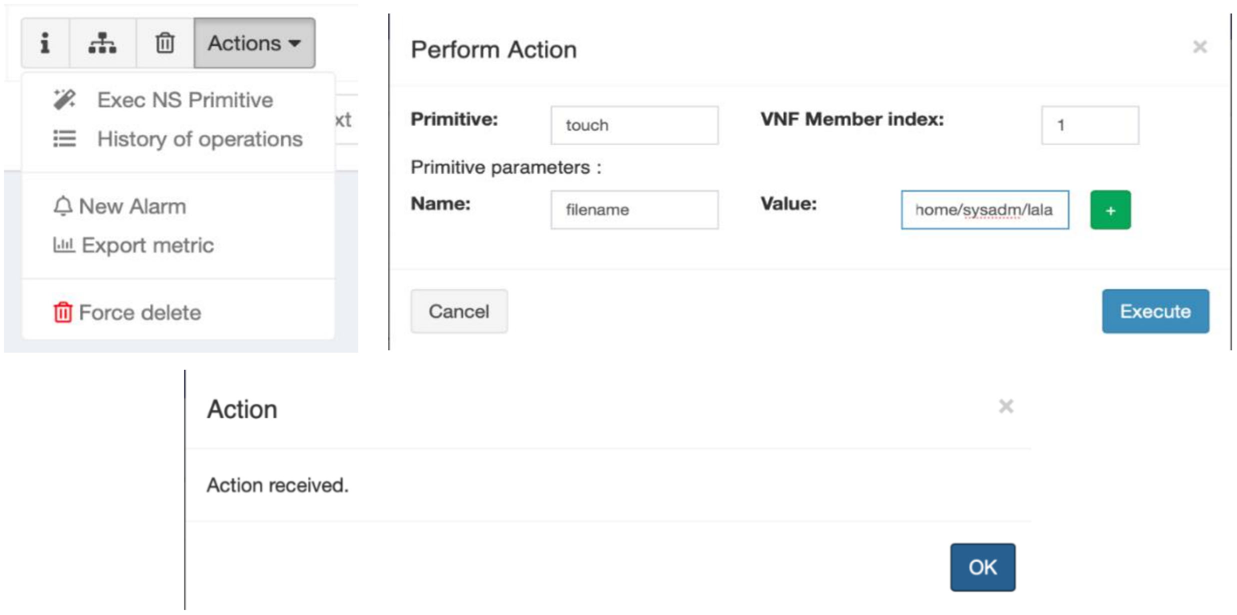
Another way of verifying if the Day-1 configuration actions are done correctly is to order the juju controller to launch them directly. To do so, follow these commands:



```
admosm@osm:~$ sudo juju run-action juju-b-aa/1 touch filename='/home/sysadm/pepe'
Action queued with id: f3ada540-f786-4a90-8b6b-b412154aa71c
admosm@osm:~$ sudo juju show-action-status f3ada540-f786-4a90-8b6b-b412154aa71c
actions:
- action: touch
  completed at: "2019-08-30 10:00:54"
  id: f3ada540-f786-4a90-8b6b-b412154aa71c
  status: completed
  unit: juju-b-aa/1
```

Figure 4-30. Juju controller launching actions.

Day-1 configuration went as expected, now it remains to verify if the Day-2 configuration actions can be done via GUI:



Perform Action

Primitive: touch **VNF Member index:** 1

Primitive parameters :

Name: filename **Value:** home/sysadm/lala

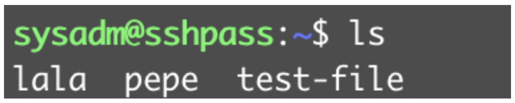
Action

Action received.

Figure 4-31. Test 3 Day-2 configuration.

To execute a Day-2 configuration action, the OSM portal has to be used, selecting a NSI actions tag and clicking over the "Exec NS Primitive" option. After doing so, a form will pop-up (see above's figure upper right screenshot), this form must be filled-up with the correct field information. Just after finalizing that action, a new form will appear confirming the action process.

Once again, after launching those "touching" actions, their results must be verified login into the VM:



```
sysadm@sshpass:~$ ls
lala  pepe  test-file
```

Figure 4-32. Test 3 touch files verification.

4.3.3.2 Ansible charm

This second part of the Test 3 is focused on using an existing juju-layer created by the 5GinFIRE 5GPPP European project that incorporates a new abstraction layer which allows the usage of Ansible Playbooks through the juju environment. This abstraction layer consists of, basically, a new juju-action that installs the latest available Ansible version on the LXD container after both, ssh credentials and ssh connection, have been verified. Although the objective of both SWs is the same, automating applications configuration, Ansible Playbooks is a SW, known by the author, that easies the challenge of automating a vEPC deployment.

Juju and proxy-charms would have required to create and specific set of actions inside a charm and programming the resulting reactive scripts to configure as desired each of the vEPC components. On the other hand, Ansible Playbooks only require an Ansible Playbook for each component with the correct configuration. The author chose the second option as a personal preference and time-saving decision.

Therefore, in order to give a try to this juju-layer, a new test was designed. This test is composed by a single VNF, with the same characteristics as the previous test VNF (juju-basic test), the same NSD networks and requirements but, with a Day-1 configuration that uses Ansible Playbooks. Its VNFD/NSD and Ansible Playbook can be consulted in the *Annexes 11.5 12.4 and 14.1.*

As it can be seen, what the Ansible Playbook does is downloading an ubuntu kernel version (5.2.0) into a directory and installs its dependencies, upgrading the system kernel. It is a simple Ansible Playbook, but it satisfies the objective of verifying the required functionalities for the vEPC test (creating directories, downloading files, using system commands such as “dpkg”, launching commands as root…) automation.

After composing the charm, building the complete VNF and Network Service descriptors, and after uploading and launching them from OSM, the following results were obtained:

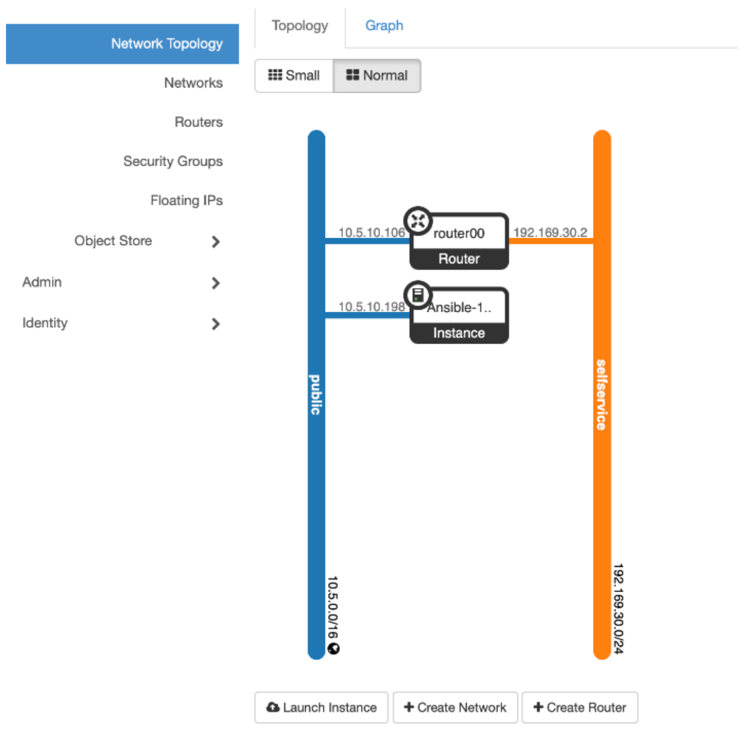
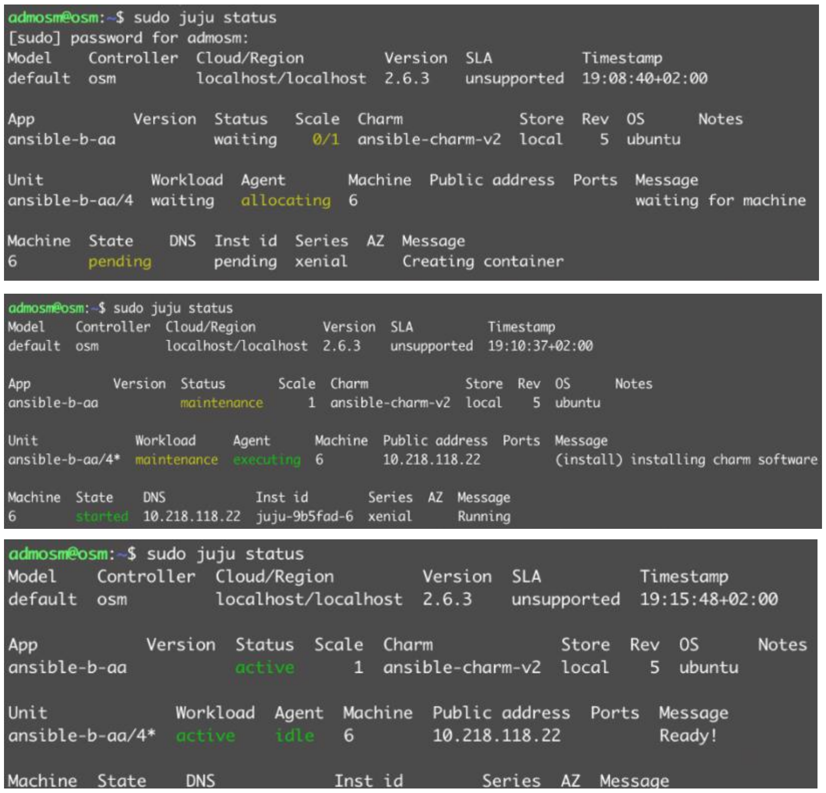


Figure 4-34. Ansible Basic OpenStack's Network Topology.

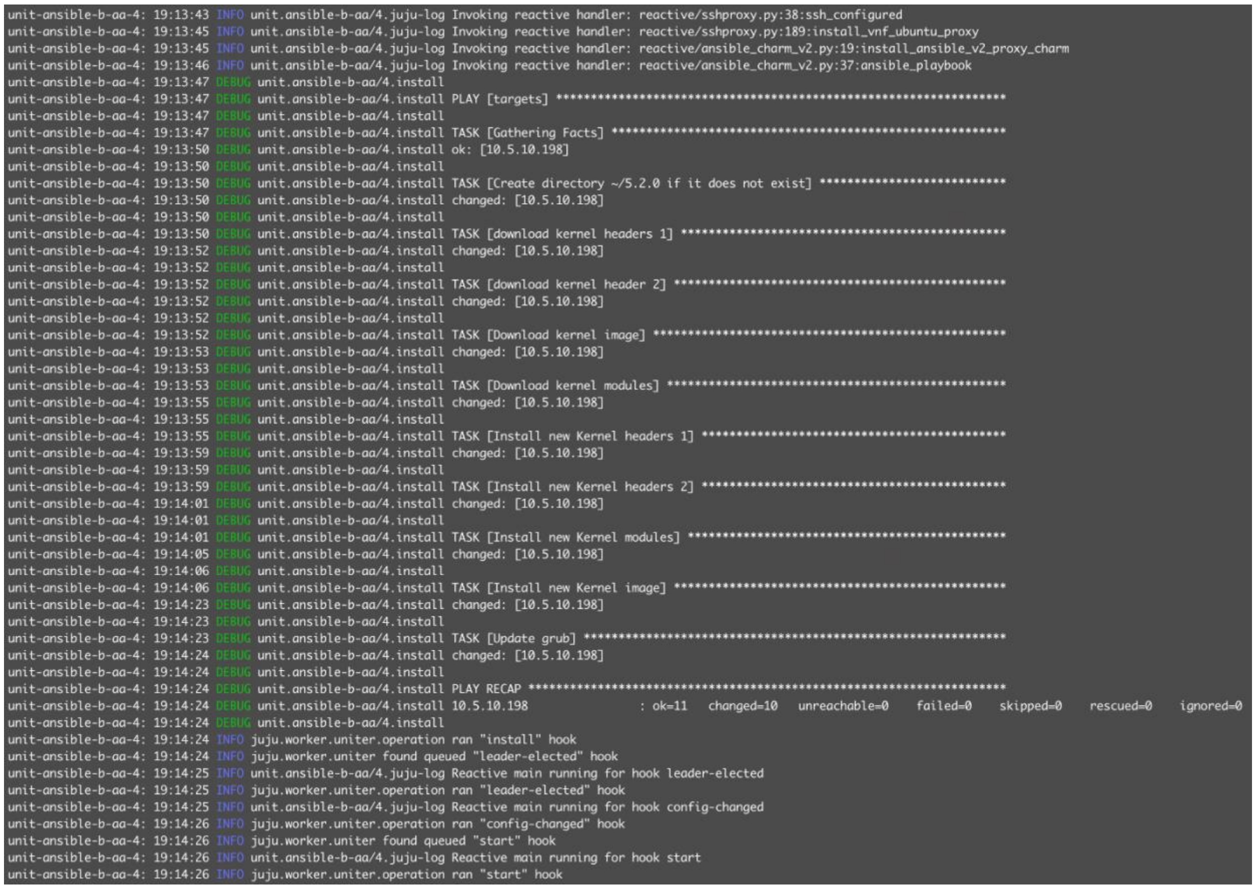

```
admosm@osm:~$ sudo juju status
[sudo] password for admosm:
Model    Controller  Cloud/Region         Version  SLA          Timestamp
default  osm         localhost/localhost  2.6.3    unsupported  19:08:40+02:00

App           Version  Status   Scale  Charm              Store  Rev  OS      Notes
ansible-b-aa           waiting    0/1  ansible-charm-v2   local    5  ubuntu

Unit            Workload  Agent       Machine  Public address  Ports  Message
ansible-b-aa/4  waiting   allocating  6                               waiting for machine

Machine  State    DNS  Inst id  Series  AZ  Message
6        pending       pending  xenial      Creating container
```

```
admosm@osm:~$ sudo juju status
Model    Controller  Cloud/Region         Version  SLA          Timestamp
default  osm         localhost/localhost  2.6.3    unsupported  19:10:37+02:00

App           Version  Status       Scale  Charm             Store  Rev  OS      Notes
ansible-b-aa           maintenance      1  ansible-charm-v2  local    5  ubuntu

Unit             Workload     Agent      Machine  Public address  Ports  Message
ansible-b-aa/4*  maintenance  executing  6        10.218.118.22          (install) installing charm software

Machine  State    DNS            Inst id       Series  AZ  Message
6        started  10.218.118.22  juju-9b5fad-6  xenial      Running
```

```
admosm@osm:~$ sudo juju status
Model    Controller  Cloud/Region         Version  SLA          Timestamp
default  osm         localhost/localhost  2.6.3    unsupported  19:15:48+02:00

App           Version  Status  Scale  Charm             Store  Rev  OS      Notes
ansible-b-aa           active      1  ansible-charm-v2  local    5  ubuntu

Unit             Workload  Agent  Machine  Public address  Ports  Message
ansible-b-aa/4*  active    idle   6        10.218.118.22          Ready!

Machine  State    DNS            Inst id             Series  AZ  Message
```

Figure 4-33. Ansible Basic Test juju status evolution.

To check the actual operational status inside the LXD container e.g. Check what commands are being executed, check what hooks are being carried out… The following command should be used: “*$ sudo juju debug-hooks ansible-b-aa/4*”



```
unit-ansible-b-aa-4: 19:13:43 INFO unit.ansible-b-aa/4.juju-log Invoking reactive handler: reactive/sshproxy.py:38:ssh_configured
unit-ansible-b-aa-4: 19:13:45 INFO unit.ansible-b-aa/4.juju-log Invoking reactive handler: reactive/sshproxy.py:189:install_vnf_ubuntu_proxy
unit-ansible-b-aa-4: 19:13:45 INFO unit.ansible-b-aa/4.juju-log Invoking reactive handler: reactive/ansible_charm_v2.py:19:install_ansible_v2_proxy_charm
unit-ansible-b-aa-4: 19:13:46 INFO unit.ansible-b-aa/4.juju-log Invoking reactive handler: reactive/ansible_charm_v2.py:37:ansible_playbook
unit-ansible-b-aa-4: 19:13:47 DEBUG unit.ansible-b-aa/4.install
unit-ansible-b-aa-4: 19:13:47 DEBUG unit.ansible-b-aa/4.install PLAY [targets] *****************************************************************
unit-ansible-b-aa-4: 19:13:47 DEBUG unit.ansible-b-aa/4.install
unit-ansible-b-aa-4: 19:13:47 DEBUG unit.ansible-b-aa/4.install TASK [Gathering Facts] *********************************************************
unit-ansible-b-aa-4: 19:13:50 DEBUG unit.ansible-b-aa/4.install ok: [10.5.10.198]
unit-ansible-b-aa-4: 19:13:50 DEBUG unit.ansible-b-aa/4.install
unit-ansible-b-aa-4: 19:13:50 DEBUG unit.ansible-b-aa/4.install TASK [Create directory ~/5.2.0 if it does not exist] ***************************
unit-ansible-b-aa-4: 19:13:50 DEBUG unit.ansible-b-aa/4.install changed: [10.5.10.198]
unit-ansible-b-aa-4: 19:13:50 DEBUG unit.ansible-b-aa/4.install
unit-ansible-b-aa-4: 19:13:50 DEBUG unit.ansible-b-aa/4.install TASK [download kernel headers 1] ***********************************************
unit-ansible-b-aa-4: 19:13:52 DEBUG unit.ansible-b-aa/4.install changed: [10.5.10.198]
unit-ansible-b-aa-4: 19:13:52 DEBUG unit.ansible-b-aa/4.install
unit-ansible-b-aa-4: 19:13:52 DEBUG unit.ansible-b-aa/4.install TASK [download kernel header 2] ************************************************
unit-ansible-b-aa-4: 19:13:52 DEBUG unit.ansible-b-aa/4.install changed: [10.5.10.198]
unit-ansible-b-aa-4: 19:13:52 DEBUG unit.ansible-b-aa/4.install
unit-ansible-b-aa-4: 19:13:52 DEBUG unit.ansible-b-aa/4.install TASK [Download kernel image] ***************************************************
unit-ansible-b-aa-4: 19:13:53 DEBUG unit.ansible-b-aa/4.install changed: [10.5.10.198]
unit-ansible-b-aa-4: 19:13:53 DEBUG unit.ansible-b-aa/4.install
unit-ansible-b-aa-4: 19:13:53 DEBUG unit.ansible-b-aa/4.install TASK [Download kernel modules] *************************************************
unit-ansible-b-aa-4: 19:13:55 DEBUG unit.ansible-b-aa/4.install changed: [10.5.10.198]
unit-ansible-b-aa-4: 19:13:55 DEBUG unit.ansible-b-aa/4.install
unit-ansible-b-aa-4: 19:13:55 DEBUG unit.ansible-b-aa/4.install TASK [Install new Kernel headers 1] ********************************************
unit-ansible-b-aa-4: 19:13:59 DEBUG unit.ansible-b-aa/4.install changed: [10.5.10.198]
unit-ansible-b-aa-4: 19:13:59 DEBUG unit.ansible-b-aa/4.install
unit-ansible-b-aa-4: 19:13:59 DEBUG unit.ansible-b-aa/4.install TASK [Install new Kernel headers 2] ********************************************
unit-ansible-b-aa-4: 19:14:01 DEBUG unit.ansible-b-aa/4.install changed: [10.5.10.198]
unit-ansible-b-aa-4: 19:14:01 DEBUG unit.ansible-b-aa/4.install
unit-ansible-b-aa-4: 19:14:01 DEBUG unit.ansible-b-aa/4.install TASK [Install new Kernel modules] **********************************************
unit-ansible-b-aa-4: 19:14:05 DEBUG unit.ansible-b-aa/4.install changed: [10.5.10.198]
unit-ansible-b-aa-4: 19:14:06 DEBUG unit.ansible-b-aa/4.install
unit-ansible-b-aa-4: 19:14:06 DEBUG unit.ansible-b-aa/4.install TASK [Install new Kernel image] ************************************************
unit-ansible-b-aa-4: 19:14:23 DEBUG unit.ansible-b-aa/4.install changed: [10.5.10.198]
unit-ansible-b-aa-4: 19:14:23 DEBUG unit.ansible-b-aa/4.install
unit-ansible-b-aa-4: 19:14:23 DEBUG unit.ansible-b-aa/4.install TASK [Update grub] *************************************************************
unit-ansible-b-aa-4: 19:14:24 DEBUG unit.ansible-b-aa/4.install changed: [10.5.10.198]
unit-ansible-b-aa-4: 19:14:24 DEBUG unit.ansible-b-aa/4.install
unit-ansible-b-aa-4: 19:14:24 DEBUG unit.ansible-b-aa/4.install PLAY RECAP *********************************************************************
unit-ansible-b-aa-4: 19:14:24 DEBUG unit.ansible-b-aa/4.install 10.5.10.198                : ok=11   changed=10   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
unit-ansible-b-aa-4: 19:14:24 DEBUG unit.ansible-b-aa/4.install
unit-ansible-b-aa-4: 19:14:24 INFO juju.worker.uniter.operation ran "install" hook
unit-ansible-b-aa-4: 19:14:24 INFO juju.worker.uniter found queued "leader-elected" hook
unit-ansible-b-aa-4: 19:14:25 INFO unit.ansible-b-aa/4.juju-log Reactive main running for hook leader-elected
unit-ansible-b-aa-4: 19:14:25 INFO juju.worker.uniter.operation ran "leader-elected" hook
unit-ansible-b-aa-4: 19:14:25 INFO unit.ansible-b-aa/4.juju-log Reactive main running for hook config-changed
unit-ansible-b-aa-4: 19:14:26 INFO juju.worker.uniter.operation ran "config-changed" hook
unit-ansible-b-aa-4: 19:14:26 INFO juju.worker.uniter found queued "start" hook
unit-ansible-b-aa-4: 19:14:26 INFO unit.ansible-b-aa/4.juju-log Reactive main running for hook start
unit-ansible-b-aa-4: 19:14:26 INFO juju.worker.uniter.operation ran "start" hook
```

Figure 4-35. Ansible Playbook Status.

As it can be observed, the Ansible Playbook configuration process went fine. Nonetheless, to double-check the process, it will be verified login into the VM:

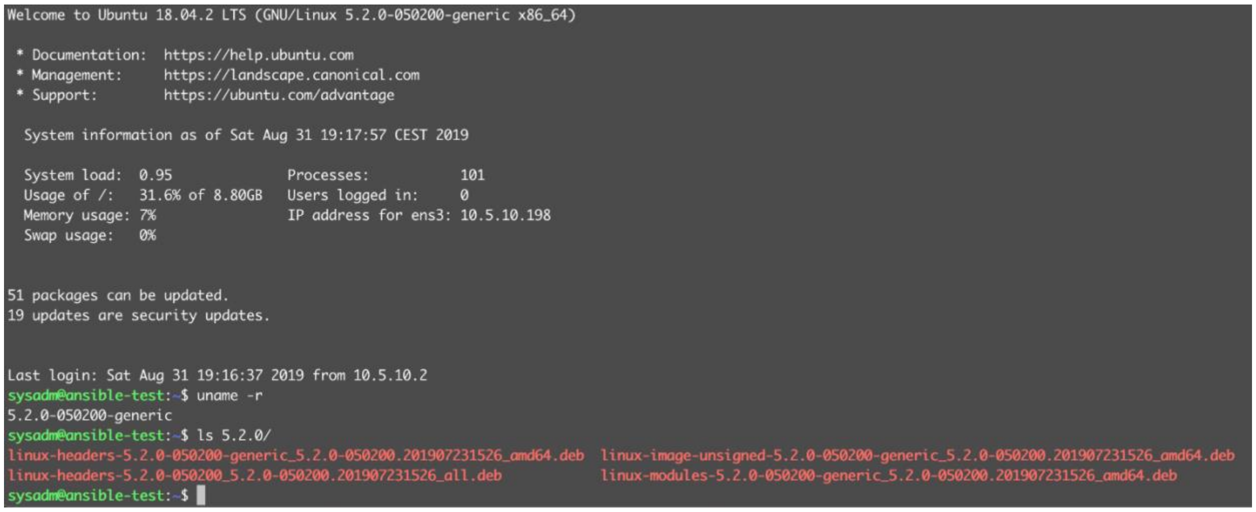

```
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 5.2.0-050200-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat Aug 31 19:17:57 CEST 2019

  System load:  0.95               Processes:           101
  Usage of /:   31.6% of 8.80GB    Users logged in:     0
  Memory usage: 7%                 IP address for ens3: 10.5.10.198
  Swap usage:   0%


51 packages can be updated.
19 updates are security updates.


Last login: Sat Aug 31 19:16:37 2019 from 10.5.10.2
sysadm@ansible-test:~$ uname -r
5.2.0-050200-generic
sysadm@ansible-test:~$ ls 5.2.0/
linux-headers-5.2.0-050200-generic_5.2.0-050200.201907231526_amd64.deb  linux-image-unsigned-5.2.0-050200-generic_5.2.0-050200.201907231526_amd64.deb
linux-headers-5.2.0-050200_5.2.0-050200.201907231526_all.deb            linux-modules-5.2.0-050200-generic_5.2.0-050200.201907231526_amd64.deb
sysadm@ansible-test:~$ 
```

Figure 4-36. Ansible Basic Test verification.

The test was completely successful, which means that, at this point more complex tests can be developed using the acquired knowledge.

4.3.4 Issues

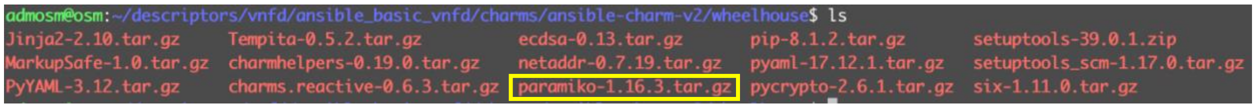
One major issue was found during the development of these latest tests which delayed learning OSM's Day-1 configuration for more than a week. The issue was discovered in the first trial of the juju-basic test, after the VNF was instantiated and the LXD container was deployed, just after a couple of minutes the command "*$ sudo juju status*" returned a result signaling that the ansible unit created for that test was blocked with the direct consequence that no "touch" action was triggered, no Day-2 configuration actions were available and no blank files were created inside the VNF VM.

After a small research the command "*$ sudo juju debug-log*" was discovered; this command returns the selected juju-unit console output so that the user can view the results of the hook execution steps. In this case, the hook "*update and upgrade*" ended successfully but the hook "*install*" returned an exception. Once again, after some research was done, it was discovered that the source of this exception was a python module called "*Paramiko*" when one of its functions tried to stablish a ssh connection to the VNF. This module is installed by the "sshproxy" juju layer in the "install " hook step.

Several inquiries were made to 5GinFIRE members in order to verify if they had experienced a similar problem, but none of them did. The major unknown was that, it a ssh connection was stablished from any other host or device to the VNF VM, no problem was encountered but these modules, Paramiko, was raising an exception when it tried to connect. VNFD ssh credentials were verified and re-verified but no problem was encountered…

After a while, "*$sudo juju debug-hooks <unit>*" command was discovered. This command opens a ssh connection with the respective LXD container in such a way that each hook action can be launched manually and that the python created environments can be accessed. Thereupon, the first step was to try and launch manually the "install" hook to see if the same exact result was obtained, it did. The second step was to try a normal ssh connection (using bash's ssh command) from the LXD container to the VNF VM, it worked. Finally, the third step was to verify if the problem was really in the python module and that it did not come from other source, to do so, a small python script was developed, using the same python sentences as the ones used in the sshproxy.py reactive file, to stablish a ssh connection; the connection couldn't be stablished and the same exception was raised from this module than the one obtained in the juju debug-log.

Once the problem source was found, it only remained to investigate the python module and its current exceptions. It was discovered that the problem came from the Paramiko module version, because the one used by the built juju proxy charms did not have the same KEX algorithms[63] than the VNF VM did. The Paramiko version installed by the juju proxy charms was *Paramiko v.1.16.3*.

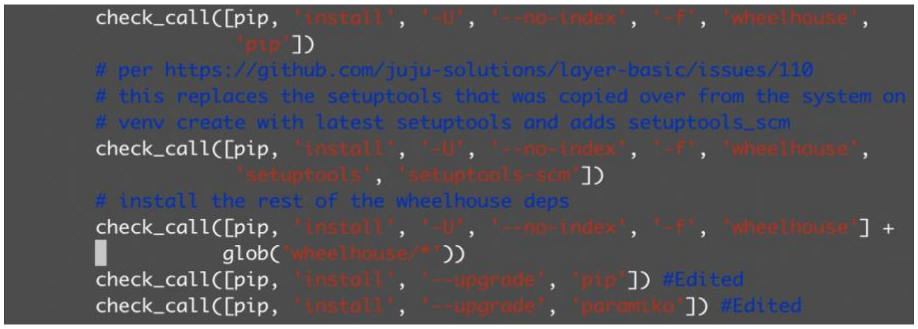


```
admosm@osm:~/descriptors/vnfd/ansible_basic_vnfd/charms/ansible-charm-v2/wheelhouse$ ls
Jinja2-2.10.tar.gz      Tempita-0.5.2.tar.gz            ecdsa-0.13.tar.gz      pip-8.1.2.tar.gz       setuptools-39.0.1.zip
MarkupSafe-1.0.tar.gz   charmhelpers-0.19.0.tar.gz      netaddr-0.7.19.tar.gz  pyaml-17.12.1.tar.gz   setuptools_scm-1.17.0.tar.gz
PyYAML-3.12.tar.gz      charms.reactive-0.6.3.tar.gz    paramiko-1.16.3.tar.gz pycrypto-2.6.1.tar.gz  six-1.11.0.tar.gz
```

Figure 4-37. Juju proxy charm Paramiko version.

The above figure shows where the LXD container Paramiko version came from. Basically, the "wheelhouse" directory is built by the first charm-composing layer, the "basic-layer", and it contains the dependencies required to execute the "install" hook. So, two solutions were accomplished:

1. Deleting the current Paramiko package and adding the latest one.
2. Editing "basic.py" code in order to upgrade the version of the pip and Paramiko modules when the "install" hook is launched. The following image shows the edited lines in this file:



```
    check_call([pip, 'install', '-U', '--no-index', '-f', 'wheelhouse',
                'pip'])
    # per https://github.com/juju-solutions/layer-basic/issues/110
    # this replaces the setuptools that was copied over from the system on
    # venv create with latest setuptools and adds setuptools_scm
    check_call([pip, 'install', '-U', '--no-index', '-f', 'wheelhouse',
                'setuptools', 'setuptools-scm'])
    # install the rest of the wheelhouse deps
    check_call([pip, 'install', '-U', '--no-index', '-f', 'wheelhouse'] +
                glob('wheelhouse/*'))
    check_call([pip, 'install', '--upgrade', 'pip']) #Edited
    check_call([pip, 'install', '--upgrade', 'paramiko']) #Edited
```

Figure 4-38. Basic.py file issue fixing.

63 https://github.com/paramiko/paramiko/issues/1455

It is important to add those lines just after the command that installs all the "wheelhouse" directory packages because if not, python2.7 and python3.5 dependencies problems will arise.

Once one of the proposed solutions is implemented, the experiments went as expected.

4.4 Test 4: Full OAI vEPC

At this point, knowledge about how to develop complex VNFDs/NSs has been acquired, as well as knowledge about how to add to the resulting VNFs and Network Services, Day-0 config and Day-1 configurations using Juju proxy charms and Ansible Playbooks. Thereupon, a test that will deploy a complete OAI automated vEPC can be carried out.

OAI has developed its SW dividing it into three main modules:

- **EPC:** Complete evolved packet core composed by MME, HSS, SPGW_U and SPGW_C modules. This module is intended to be run on a single machine but, as it is going to be explained later, in this test each sub-module will be launched on a dedicated VM.
- **eNB**
- **UE**

This test is going to be focused only in the first main module, the EPC. As mentioned before, this module is intended to be run on a single machine that hosts all the EPC sub-modules functionalities but, as 5G networks are being developed towards scalable, flexible and programmable environments; in this test each sub-module will be run on a dedicated VM. Thanks to this division, each VM has its own dedicated resources and, if at some point, one of the EPC sub-modules needs more resources, the respective VM/VNF can be escalated individually.

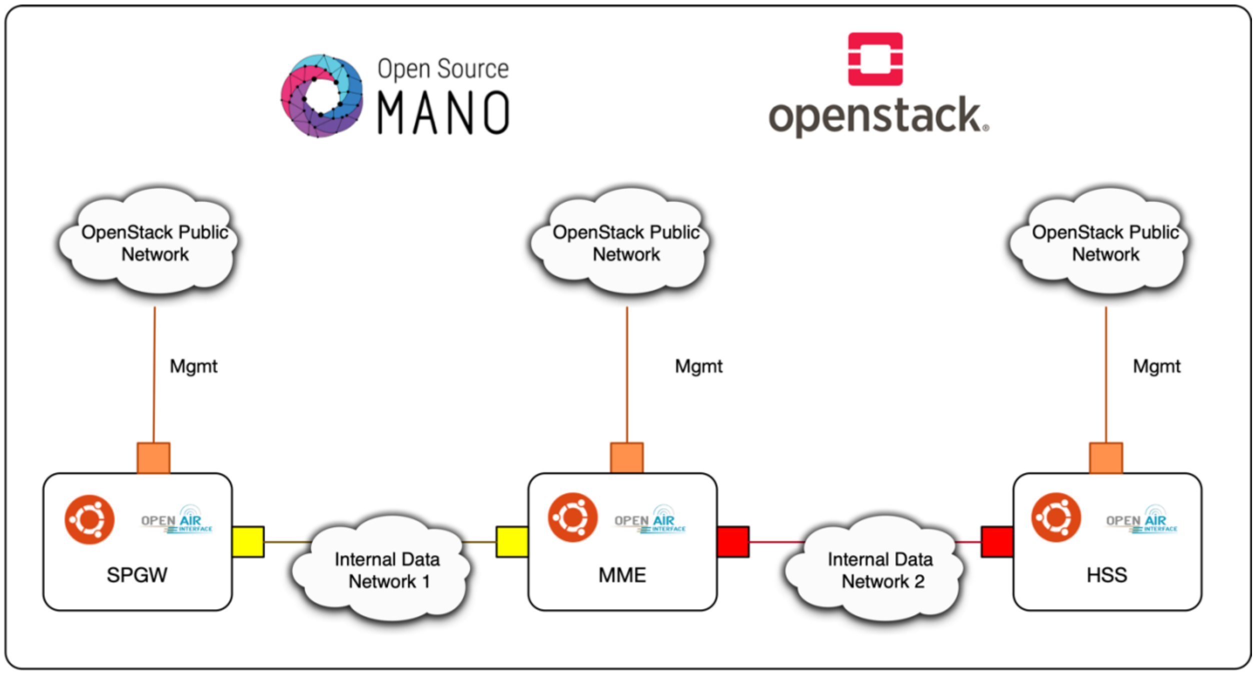


Figure 4-39. Test 4 High Level View.

4.4.1 Test Summary

This section provides a summary of the current Test Characteristics.

Test 3	
Scenario	Production 1/Production 2
VNFs	3
VNFDs	3
NSs	1
NSDs	1
PNFs	0
PNFDs	0
NSTs	0
Networks	3
Total RAM	12 GB
Total vCPUS	8
Total Disk	120 GB
Day-0 Config	Yes
Day-1 Config	Yes (Ansible)
Day-2 Config	No

Table 4-7. Test 4 summary.

4.4.2 Test Description

One of the big challenges of this test was to stablish the right network configuration to get the correct equivalence between OAIs EPC network interfaces and the required Network Service's management and internal data network interfaces. Several approaches were designed:

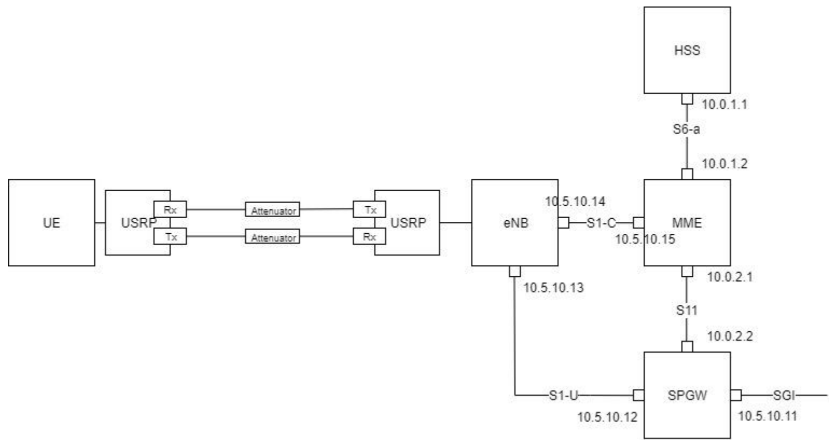


Figure 4-40. Test 4 network design 1.

The previous figure had two main problems, HSS VNF lacked a management interface which implies that no Internet connection was available and, consequently, no configuration could be carried out. In second place, eNB and SPGW VNFs had two different network interfaces connected to the same network which could derive in major routing issues.

So, a second design was developed:

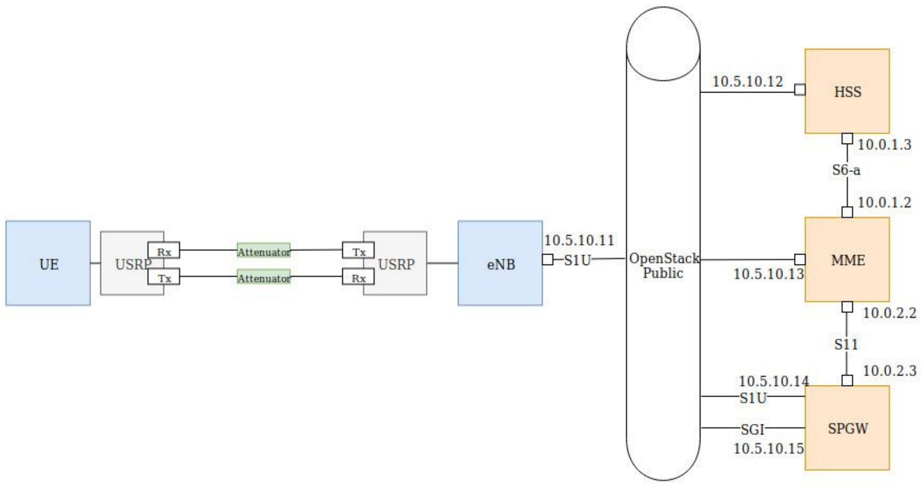


Figure 4-41. Test 4 network design 2.

In this case, a management interface was added to the HSS VNF, the double network interface was deleted from the eNB VNF but maintained in the SPGW VNF. The later one was maintained because OAI SW does not allow to run SPGW *SGI and S1-U* interfaces on a same network interface. Again, routing problems raised from this design and when the SPGW was running, Internet connection was lost.

Finally, a third and last design was done:

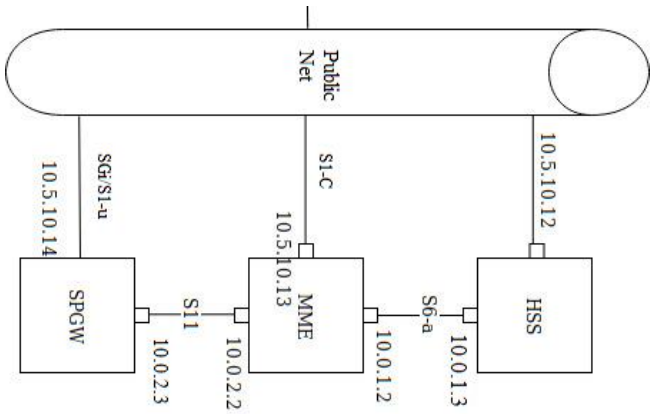


Figure 4-42. Test 4 network design 3.

Although it can seem as the same interface in MME and SPGW is acting as various types of EPC interfaces, Linux network sub-interfaces are used. In fact, OAI's SPGW is divided into two sub-modules (SPGW_U and SPGW_C) that will run, in this test, inside the same VNF-VM; this means than additional virtual network interfaces are required on that VNF. Therefore, a more detailed network architecture view was developed, mainly due to clarifying purposes (editing OAI's configuration files without a clear network view was a real mess). Below, it is shown:

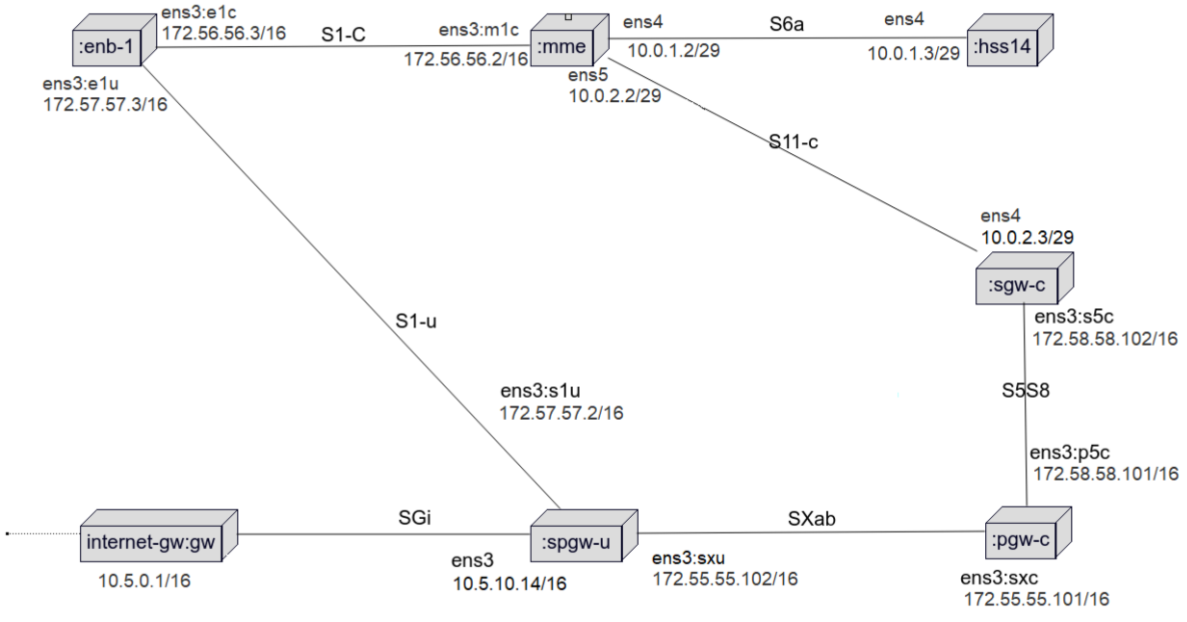


Figure 4-43Test 4 detailed network final design.

Only the OAI-EPC related network interfaces are shown, each one named with its functionality name and with three sub-modules acting as the whole SPGW.

Once the network architecture was clear, the next step was to define the VNFDs/NSDs in order to have an equivalent structure. To do so, the following metadata fields had to be added to the NSD to achieve self-defined fixed internal networks:

```
ip-profiles:
    - ip-profile-params:
        gateway-address: null
        ip-version: ipv4
        subnet-address: 10.0.1.0/29
      name: S6-a
    - ip-profile-params:
        gateway-address: null
        ip-version: ipv4
        subnet-address: 10.0.2.0/29
      name: S11
```

Text Box 4-23. NSD extra metadata fields.

These metadata fields allow OSM's RO to request self-defined fixed networks to OpenStack. It is important to remark that "/29" networks were used instead of "/30" despite only two hosts were going to be addressed inside these networks, because OpenStack deploys by default a DHCP agent on every "xxx.yyy.www.1/xx" IP and if it was deactivated from the VNFD, network issues raised in OSM Rel. FIVE. Other than that, no new metadata fields were added.

Moreover, in all the VNFs the same Day-0 configuration has been added with the only difference located in the VMs hostname:

```
#cloud-config
hostname: <spgw|hss|mme>
system_info:
  default_user:
    name: sysadm
password: 5gvinni2019
chpasswd: { expire: False }
ssh_pwauth: True
final_message: "The system is finally up, after $UPTIME seconds"
```

Text Box 4-24. Test 4 Day-0 configuration.

Finally, three Ansible playbooks have been developed, one for each EPC required VNF. As their extension is large, they are shown in *Annexes 14.2, 14.3 and 14.4*. Here, the configuration done by each of them is detailed:

- **HSS Playbook:** Ubuntu Bionic package repositories are updated, and packages upgraded, *"/etc/hosts"* file is edited in order to recognize "hss" as a host. OAI git repository is cloned and two building Cassandra files are edited: the first one to allow Cassandra service start/stop actions and log removal from the configuring OAI script and the second one, to avoid local environment variables to collide when executing the building command from Ansible. Afterwards, Cassandra and HSS_rel14 are compiled and build; and Cassandra database tables are populated to host Cassandra REALM on the localhost IP, to address a range of IMSIs and to allow a max number of 20 users. Finally, HSS configuration files are edited to achieve the above-mentioned network configuration and the required certificates and keys are downloaded.

- **MME Playbook:** Same first steps as the previous playbook. Afterwards, copies the required files from the building directory to the proper paths and edits them in order to have the following configuration:
  - **HSS** with a defined realm called *"OpenAir5G.Alliance"*.
  - **Mobile Country Code (MCC):** 34
  - **Mobile Network Code (MNC):** 67
  - **Tracking Area Code (TAC):** 1
  - **S10 Interface Address:** 127.0.0.100
  - **Above-mentioned network configuration**
  - **TCP/TLS enabled**
  - **No IPv6 available**
  - **Several extensions added**
  - **HSS configured as expected**

  At the end, a sub-interface is generated to hold the MME S1-C future interface connection.

- **SPGW Playbook:** Same first steps as the previous playbooks. This is the simplest playbook, both *SPGW_U and SPGW_C* are compiled and build and, when that process has ended, the proper files are modified in order to achieve the above-mentioned network configuration. Two DNS servers are added, and several sub-interfaces are generated in order to stablish all the required connection/interfaces.

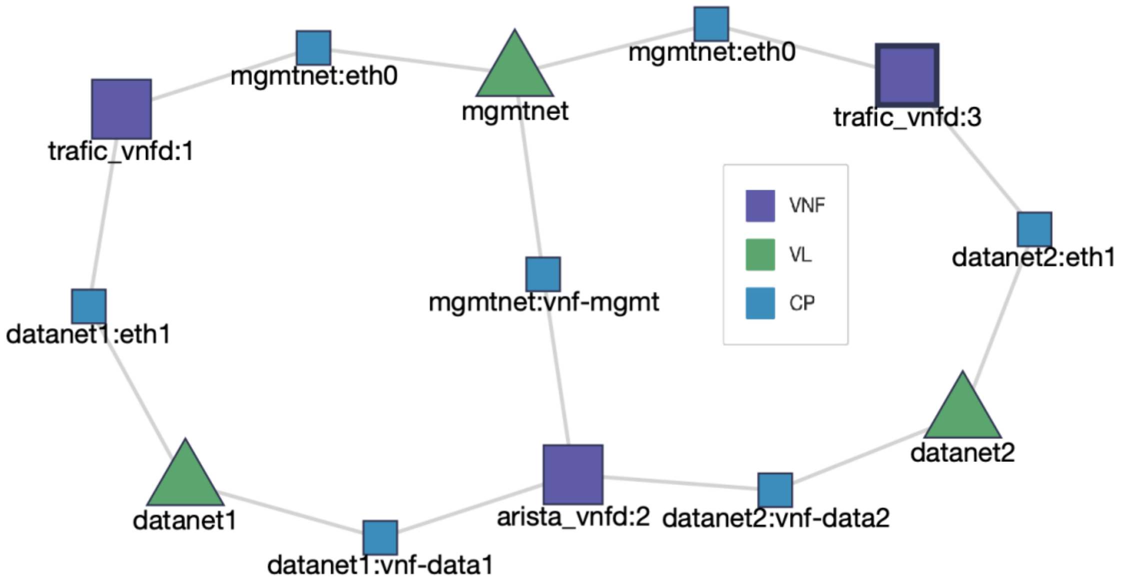


Figure 4-44. Test 4 OAI vEPC NSD graph.

4.4.3 Test Results

Although here are going to be shown the final and fine results, several trials were needed before achieving the expected and correct behavior without any issues.

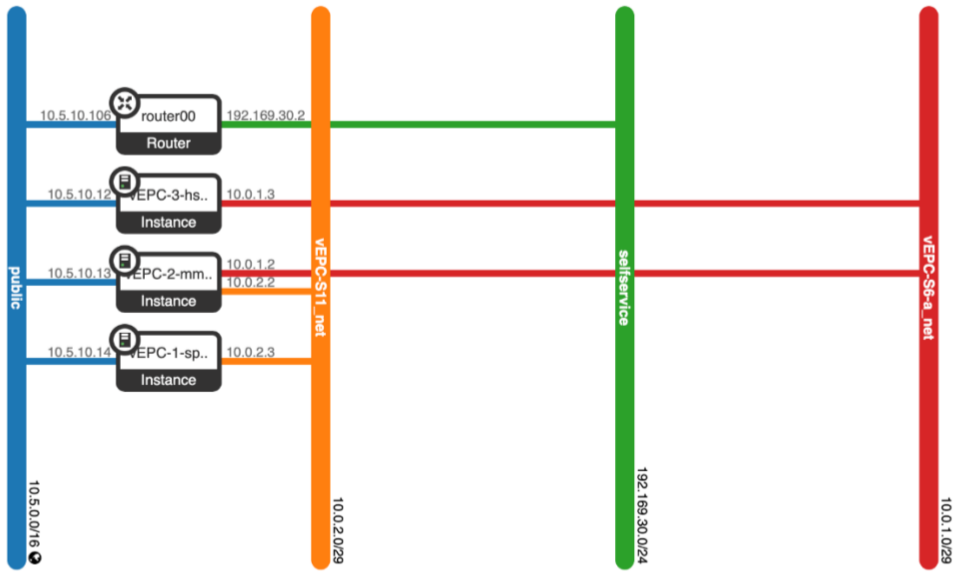
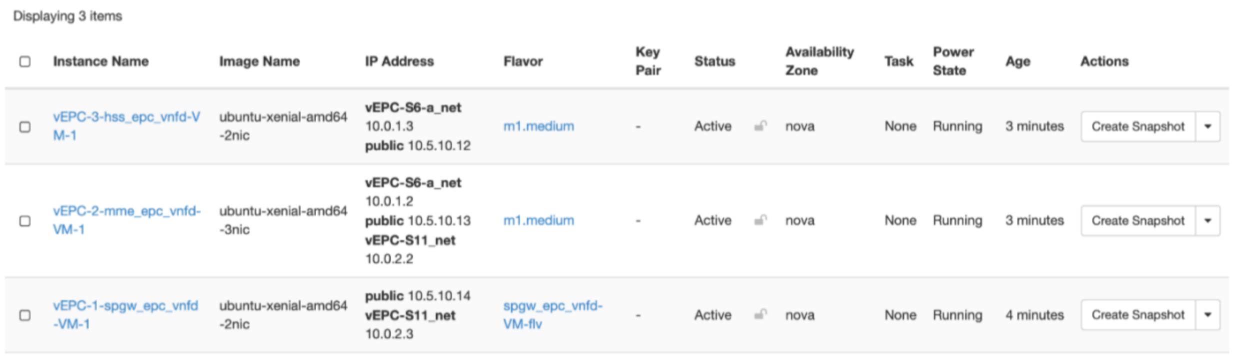


Figure 4-45. Test 4 OpenStack's network topology.

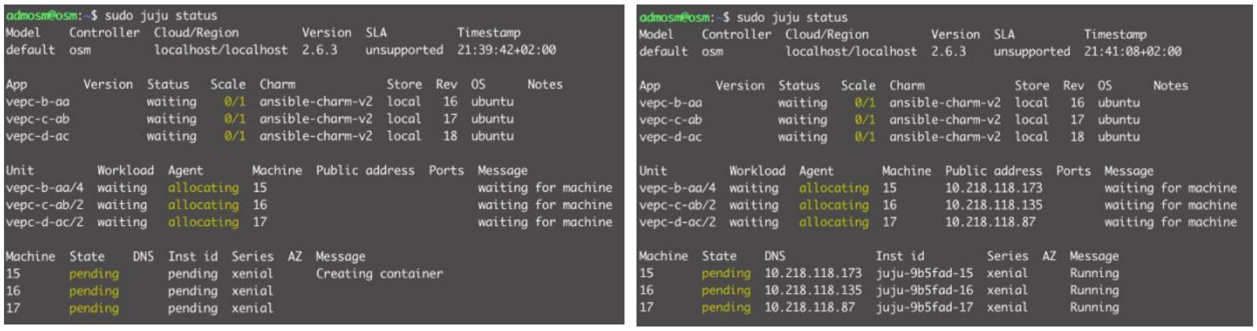


Displaying 3 items

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
vEPC-3-hss_epc_vnfd-VM-1	ubuntu-xenial-amd64-2nic	**vEPC-S6-a_net** 10.0.1.3 **public** 10.5.10.12	m1.medium	-	Active	nova	None	Running	3 minutes	Create Snapshot
vEPC-2-mme_epc_vnfd-VM-1	ubuntu-xenial-amd64-3nic	**vEPC-S6-a_net** 10.0.1.2 **public** 10.5.10.13 **vEPC-S11_net** 10.0.2.2	m1.medium	-	Active	nova	None	Running	3 minutes	Create Snapshot
vEPC-1-spgw_epc_vnfd-VM-1	ubuntu-xenial-amd64-2nic	**public** 10.5.10.14 **vEPC-S11_net** 10.0.2.3	spgw_epc_vnfd-VM-flv	-	Active	nova	None	Running	4 minutes	Create Snapshot

Figure 4-46. Test 4 OpenStack's VM information.

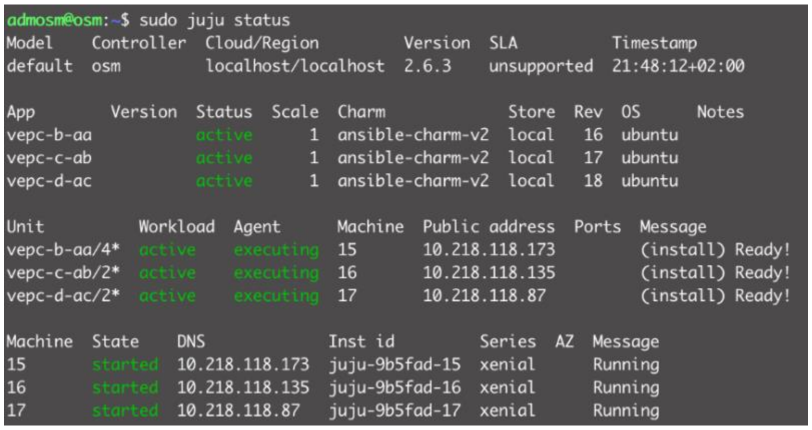
Once the VNFs VMs were instantiated and all the virtual networks generated, OpenStack's metadata agent sends a signal to OSM's RO signaling the end of the process and OSM's RO sends a signal to its SO in order to commence the deployment of the required LXD containers. This second phase has several stages: LXD containers generation, LXD containers IP addressing, LXD containers SW configuration (the required packages to communicate with OpenStack's VMs through the juju proxy ansible charm are installed) and finally, OpenStack's VMs configuration through Ansible Playbooks. The entire process has been followed and timed step by step:



```
admosm@osm:~$ sudo juju status
Model    Controller  Cloud/Region         Version  SLA          Timestamp
default  osm         localhost/localhost  2.6.3    unsupported  21:39:42+02:00

App        Version  Status   Scale  Charm              Store  Rev  OS      Notes
vepc-b-aa           waiting    0/1  ansible-charm-v2   local   16  ubuntu
vepc-c-ab           waiting    0/1  ansible-charm-v2   local   17  ubuntu
vepc-d-ac           waiting    0/1  ansible-charm-v2   local   18  ubuntu

Unit         Workload  Agent       Machine  Public address  Ports  Message
vepc-b-aa/4  waiting   allocating  15                              waiting for machine
vepc-c-ab/2  waiting   allocating  16                              waiting for machine
vepc-d-ac/2  waiting   allocating  17                              waiting for machine

Machine  State    DNS  Inst id  Series  AZ  Message
15       pending       pending  xenial      Creating container
16       pending       pending  xenial
17       pending       pending  xenial
```

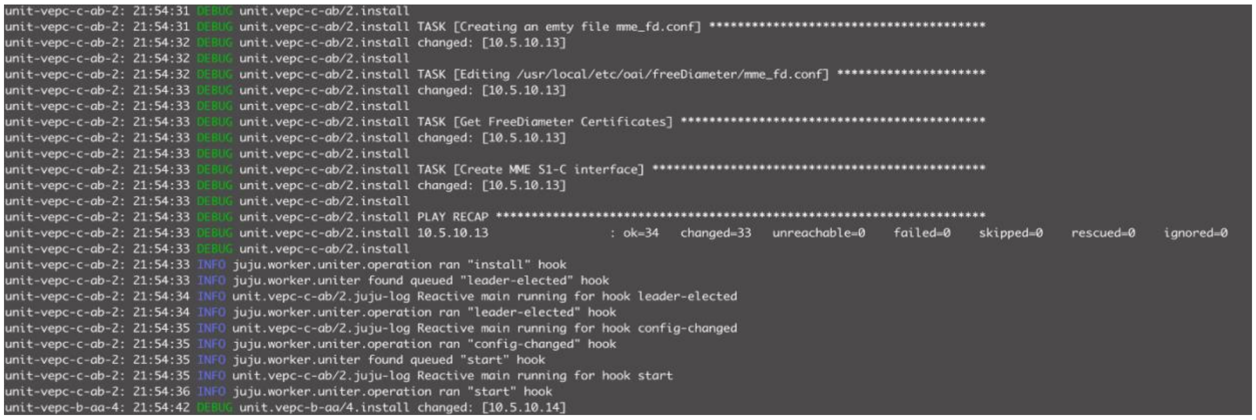
```
admosm@osm:~$ sudo juju status
Model    Controller  Cloud/Region         Version  SLA          Timestamp
default  osm         localhost/localhost  2.6.3    unsupported  21:41:08+02:00

App        Version  Status   Scale  Charm              Store  Rev  OS      Notes
vepc-b-aa           waiting    0/1  ansible-charm-v2   local   16  ubuntu
vepc-c-ab           waiting    0/1  ansible-charm-v2   local   17  ubuntu
vepc-d-ac           waiting    0/1  ansible-charm-v2   local   18  ubuntu

Unit         Workload  Agent       Machine  Public address  Ports  Message
vepc-b-aa/4  waiting   allocating  15       10.218.118.173         waiting for machine
vepc-c-ab/2  waiting   allocating  16       10.218.118.135         waiting for machine
vepc-d-ac/2  waiting   allocating  17       10.218.118.87          waiting for machine

Machine  State    DNS             Inst id          Series  AZ  Message
15       pending  10.218.118.173  juju-9b5fad-15   xenial      Running
16       pending  10.218.118.135  juju-9b5fad-16   xenial      Running
17       pending  10.218.118.87   juju-9b5fad-17   xenial      Running
```

Figure 4-47. Test 4 LXD containers creation and IP addressing.



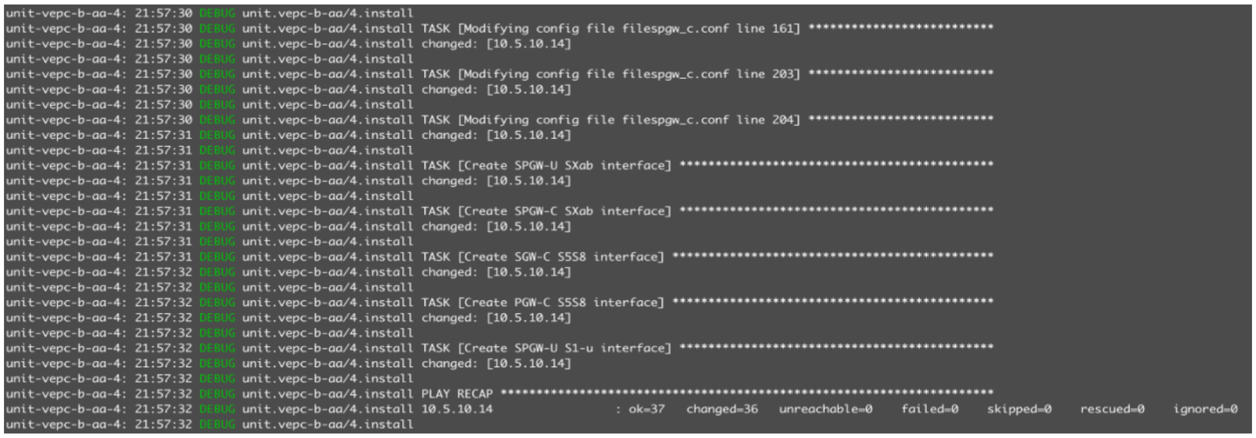
```
admosm@osm:~$ sudo juju status
Model    Controller  Cloud/Region         Version  SLA          Timestamp
default  osm         localhost/localhost  2.6.3    unsupported  21:48:12+02:00

App        Version  Status  Scale  Charm              Store  Rev  OS      Notes
vepc-b-aa           active      1  ansible-charm-v2   local   16  ubuntu
vepc-c-ab           active      1  ansible-charm-v2   local   17  ubuntu
vepc-d-ac           active      1  ansible-charm-v2   local   18  ubuntu

Unit          Workload  Agent      Machine  Public address  Ports  Message
vepc-b-aa/4*  active    executing  15       10.218.118.173         (install) Ready!
vepc-c-ab/2*  active    executing  16       10.218.118.135         (install) Ready!
vepc-d-ac/2*  active    executing  17       10.218.118.87          (install) Ready!

Machine  State    DNS             Inst id          Series  AZ  Message
15       started  10.218.118.173  juju-9b5fad-15   xenial      Running
16       started  10.218.118.135  juju-9b5fad-16   xenial      Running
17       started  10.218.118.87   juju-9b5fad-17   xenial      Running
```

Figure 4-48. Test 4 LXD container finished SW configuration.



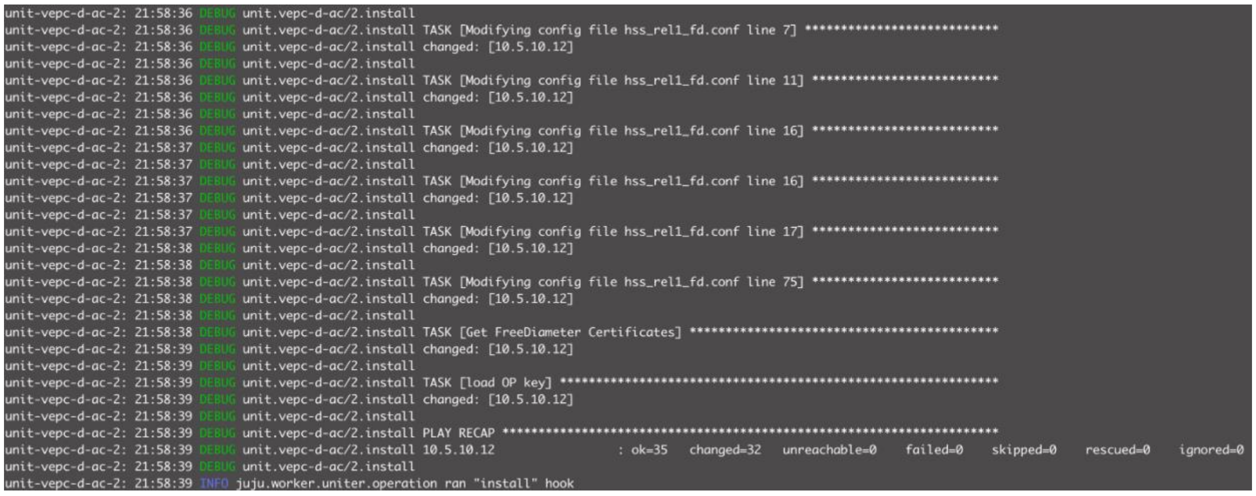
```
unit-vepc-c-ab-2: 21:54:31 DEBUG unit.vepc-c-ab/2.install
unit-vepc-c-ab-2: 21:54:31 DEBUG unit.vepc-c-ab/2.install TASK [Creating an emty file mme_fd.conf] ****************************************
unit-vepc-c-ab-2: 21:54:32 DEBUG unit.vepc-c-ab/2.install changed: [10.5.10.13]
unit-vepc-c-ab-2: 21:54:32 DEBUG unit.vepc-c-ab/2.install
unit-vepc-c-ab-2: 21:54:32 DEBUG unit.vepc-c-ab/2.install TASK [Editing /usr/local/etc/oai/freeDiameter/mme_fd.conf] *********************
unit-vepc-c-ab-2: 21:54:33 DEBUG unit.vepc-c-ab/2.install changed: [10.5.10.13]
unit-vepc-c-ab-2: 21:54:33 DEBUG unit.vepc-c-ab/2.install
unit-vepc-c-ab-2: 21:54:33 DEBUG unit.vepc-c-ab/2.install TASK [Get FreeDiameter Certificates] ********************************************
unit-vepc-c-ab-2: 21:54:33 DEBUG unit.vepc-c-ab/2.install changed: [10.5.10.13]
unit-vepc-c-ab-2: 21:54:33 DEBUG unit.vepc-c-ab/2.install
unit-vepc-c-ab-2: 21:54:33 DEBUG unit.vepc-c-ab/2.install TASK [Create MME S1-C interface] ************************************************
unit-vepc-c-ab-2: 21:54:33 DEBUG unit.vepc-c-ab/2.install changed: [10.5.10.13]
unit-vepc-c-ab-2: 21:54:33 DEBUG unit.vepc-c-ab/2.install
unit-vepc-c-ab-2: 21:54:33 DEBUG unit.vepc-c-ab/2.install PLAY RECAP **********************************************************************
unit-vepc-c-ab-2: 21:54:33 DEBUG unit.vepc-c-ab/2.install 10.5.10.13                : ok=34   changed=33   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
unit-vepc-c-ab-2: 21:54:33 DEBUG unit.vepc-c-ab/2.install
unit-vepc-c-ab-2: 21:54:33 INFO juju.worker.uniter.operation ran "install" hook
unit-vepc-c-ab-2: 21:54:33 INFO juju.worker.uniter found queued "leader-elected" hook
unit-vepc-c-ab-2: 21:54:34 INFO unit.vepc-c-ab/2.juju-log Reactive main running for hook leader-elected
unit-vepc-c-ab-2: 21:54:34 INFO juju.worker.uniter.operation ran "leader-elected" hook
unit-vepc-c-ab-2: 21:54:35 INFO unit.vepc-c-ab/2.juju-log Reactive main running for hook config-changed
unit-vepc-c-ab-2: 21:54:35 INFO juju.worker.uniter.operation ran "config-changed" hook
unit-vepc-c-ab-2: 21:54:35 INFO juju.worker.uniter found queued "start" hook
unit-vepc-c-ab-2: 21:54:35 INFO unit.vepc-c-ab/2.juju-log Reactive main running for hook start
unit-vepc-c-ab-2: 21:54:36 INFO juju.worker.uniter.operation ran "start" hook
unit-vepc-b-aa-4: 21:54:42 DEBUG unit.vepc-b-aa/4.install changed: [10.5.10.14]
```

Figure 4-49. Test 4 MME's Ansible Playbook configuration end.



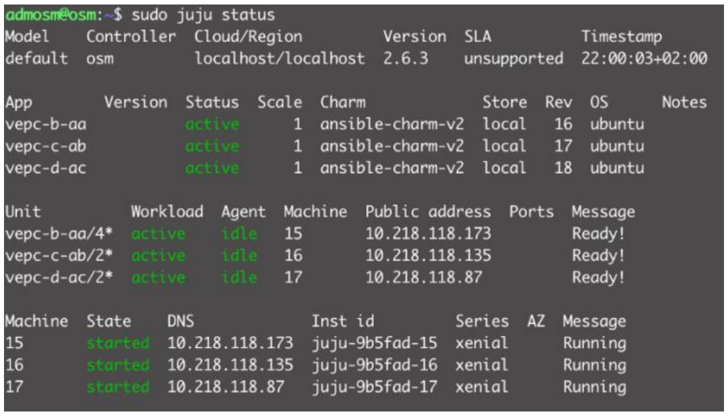
```
unit-vepc-b-aa-4: 21:57:30 DEBUG unit.vepc-b-aa/4.install
unit-vepc-b-aa-4: 21:57:30 DEBUG unit.vepc-b-aa/4.install TASK [Modifying config file filespgw_c.conf line 161] *************************
unit-vepc-b-aa-4: 21:57:30 DEBUG unit.vepc-b-aa/4.install changed: [10.5.10.14]
unit-vepc-b-aa-4: 21:57:30 DEBUG unit.vepc-b-aa/4.install
unit-vepc-b-aa-4: 21:57:30 DEBUG unit.vepc-b-aa/4.install TASK [Modifying config file filespgw_c.conf line 203] *************************
unit-vepc-b-aa-4: 21:57:30 DEBUG unit.vepc-b-aa/4.install changed: [10.5.10.14]
unit-vepc-b-aa-4: 21:57:30 DEBUG unit.vepc-b-aa/4.install
unit-vepc-b-aa-4: 21:57:30 DEBUG unit.vepc-b-aa/4.install TASK [Modifying config file filespgw_c.conf line 204] *************************
unit-vepc-b-aa-4: 21:57:31 DEBUG unit.vepc-b-aa/4.install changed: [10.5.10.14]
unit-vepc-b-aa-4: 21:57:31 DEBUG unit.vepc-b-aa/4.install
unit-vepc-b-aa-4: 21:57:31 DEBUG unit.vepc-b-aa/4.install TASK [Create SPGW-U SXab interface] *******************************************
unit-vepc-b-aa-4: 21:57:31 DEBUG unit.vepc-b-aa/4.install changed: [10.5.10.14]
unit-vepc-b-aa-4: 21:57:31 DEBUG unit.vepc-b-aa/4.install
unit-vepc-b-aa-4: 21:57:31 DEBUG unit.vepc-b-aa/4.install TASK [Create SPGW-C SXab interface] *******************************************
unit-vepc-b-aa-4: 21:57:31 DEBUG unit.vepc-b-aa/4.install changed: [10.5.10.14]
unit-vepc-b-aa-4: 21:57:31 DEBUG unit.vepc-b-aa/4.install
unit-vepc-b-aa-4: 21:57:31 DEBUG unit.vepc-b-aa/4.install TASK [Create SGW-C S5S8 interface] ********************************************
unit-vepc-b-aa-4: 21:57:32 DEBUG unit.vepc-b-aa/4.install changed: [10.5.10.14]
unit-vepc-b-aa-4: 21:57:32 DEBUG unit.vepc-b-aa/4.install
unit-vepc-b-aa-4: 21:57:32 DEBUG unit.vepc-b-aa/4.install TASK [Create PGW-C S5S8 interface] ********************************************
unit-vepc-b-aa-4: 21:57:32 DEBUG unit.vepc-b-aa/4.install changed: [10.5.10.14]
unit-vepc-b-aa-4: 21:57:32 DEBUG unit.vepc-b-aa/4.install
unit-vepc-b-aa-4: 21:57:32 DEBUG unit.vepc-b-aa/4.install TASK [Create SPGW-U S1-u interface] *******************************************
unit-vepc-b-aa-4: 21:57:32 DEBUG unit.vepc-b-aa/4.install changed: [10.5.10.14]
unit-vepc-b-aa-4: 21:57:32 DEBUG unit.vepc-b-aa/4.install
unit-vepc-b-aa-4: 21:57:32 DEBUG unit.vepc-b-aa/4.install PLAY RECAP **********************************************************************
unit-vepc-b-aa-4: 21:57:32 DEBUG unit.vepc-b-aa/4.install 10.5.10.14                : ok=37   changed=36   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
unit-vepc-b-aa-4: 21:57:32 DEBUG unit.vepc-b-aa/4.install
```

Figure 4-50. Test 4 SPGW's Ansible Playbook configuration end.



```
unit-vepc-d-ac-2: 21:58:36 DEBUG unit.vepc-d-ac/2.install
unit-vepc-d-ac-2: 21:58:36 DEBUG unit.vepc-d-ac/2.install TASK [Modifying config file hss_rel1_fd.conf line 7] ***************************
unit-vepc-d-ac-2: 21:58:36 DEBUG unit.vepc-d-ac/2.install changed: [10.5.10.12]
unit-vepc-d-ac-2: 21:58:36 DEBUG unit.vepc-d-ac/2.install
unit-vepc-d-ac-2: 21:58:36 DEBUG unit.vepc-d-ac/2.install TASK [Modifying config file hss_rel1_fd.conf line 11] *************************
unit-vepc-d-ac-2: 21:58:36 DEBUG unit.vepc-d-ac/2.install changed: [10.5.10.12]
unit-vepc-d-ac-2: 21:58:36 DEBUG unit.vepc-d-ac/2.install
unit-vepc-d-ac-2: 21:58:36 DEBUG unit.vepc-d-ac/2.install TASK [Modifying config file hss_rel1_fd.conf line 16] *************************
unit-vepc-d-ac-2: 21:58:37 DEBUG unit.vepc-d-ac/2.install changed: [10.5.10.12]
unit-vepc-d-ac-2: 21:58:37 DEBUG unit.vepc-d-ac/2.install
unit-vepc-d-ac-2: 21:58:37 DEBUG unit.vepc-d-ac/2.install TASK [Modifying config file hss_rel1_fd.conf line 16] *************************
unit-vepc-d-ac-2: 21:58:37 DEBUG unit.vepc-d-ac/2.install changed: [10.5.10.12]
unit-vepc-d-ac-2: 21:58:37 DEBUG unit.vepc-d-ac/2.install
unit-vepc-d-ac-2: 21:58:37 DEBUG unit.vepc-d-ac/2.install TASK [Modifying config file hss_rel1_fd.conf line 17] *************************
unit-vepc-d-ac-2: 21:58:38 DEBUG unit.vepc-d-ac/2.install changed: [10.5.10.12]
unit-vepc-d-ac-2: 21:58:38 DEBUG unit.vepc-d-ac/2.install
unit-vepc-d-ac-2: 21:58:38 DEBUG unit.vepc-d-ac/2.install TASK [Modifying config file hss_rel1_fd.conf line 75] *************************
unit-vepc-d-ac-2: 21:58:38 DEBUG unit.vepc-d-ac/2.install changed: [10.5.10.12]
unit-vepc-d-ac-2: 21:58:38 DEBUG unit.vepc-d-ac/2.install
unit-vepc-d-ac-2: 21:58:38 DEBUG unit.vepc-d-ac/2.install TASK [Get FreeDiameter Certificates] ********************************************
unit-vepc-d-ac-2: 21:58:39 DEBUG unit.vepc-d-ac/2.install changed: [10.5.10.12]
unit-vepc-d-ac-2: 21:58:39 DEBUG unit.vepc-d-ac/2.install
unit-vepc-d-ac-2: 21:58:39 DEBUG unit.vepc-d-ac/2.install TASK [load OP key] **************************************************************
unit-vepc-d-ac-2: 21:58:39 DEBUG unit.vepc-d-ac/2.install changed: [10.5.10.12]
unit-vepc-d-ac-2: 21:58:39 DEBUG unit.vepc-d-ac/2.install
unit-vepc-d-ac-2: 21:58:39 DEBUG unit.vepc-d-ac/2.install PLAY RECAP **********************************************************************
unit-vepc-d-ac-2: 21:58:39 DEBUG unit.vepc-d-ac/2.install 10.5.10.12                : ok=35   changed=32   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
unit-vepc-d-ac-2: 21:58:39 DEBUG unit.vepc-d-ac/2.install
unit-vepc-d-ac-2: 21:58:39 INFO juju.worker.uniter.operation ran "install" hook
```

Figure 4-51. Test 4 HSS's Ansible Playbook configuration end.



```
admosm@osm:~$ sudo juju status
Model    Controller  Cloud/Region         Version  SLA          Timestamp
default  osm         localhost/localhost  2.6.3    unsupported  22:00:03+02:00

App        Version  Status  Scale  Charm              Store  Rev  OS      Notes
vepc-b-aa           active      1  ansible-charm-v2  local   16  ubuntu
vepc-c-ab           active      1  ansible-charm-v2  local   17  ubuntu
vepc-d-ac           active      1  ansible-charm-v2  local   18  ubuntu

Unit           Workload  Agent  Machine  Public address  Ports  Message
vepc-b-aa/4*   active    idle   15       10.218.118.173         Ready!
vepc-c-ab/2*   active    idle   16       10.218.118.135         Ready!
vepc-d-ac/2*   active    idle   17       10.218.118.87          Ready!

Machine  State    DNS             Inst id        Series  AZ  Message
15       started  10.218.118.173  juju-9b5fad-15  xenial      Running
16       started  10.218.118.135  juju-9b5fad-16  xenial      Running
17       started  10.218.118.87   juju-9b5fad-17  xenial      Running
```

Figure 4-52. Test 4 Day-1 configuration completed.

After that, as several issues raise if a screen session is started from the Ansible Playbooks e.g. No installing termination signal is sent from the juju controller to OSM's SO and, consequently, the last stage is never reached; the process started in the screen session is aborted if the ansible playbook connection is closed. Each of the services MUST be run manually with the following commands in the respective VMs:

```
$ screen -S hss

$ oai_hss -j /usr/local/etc/oai/hss_rel14.json --onlyloadkey
```

Text Box 4-28. Test 4 HSS launch.

```
$ screen -S mme

$ /home/sysadm/openair-cn/scripts/run_mme
```

Text Box 4-27. Test 4 MME launch.

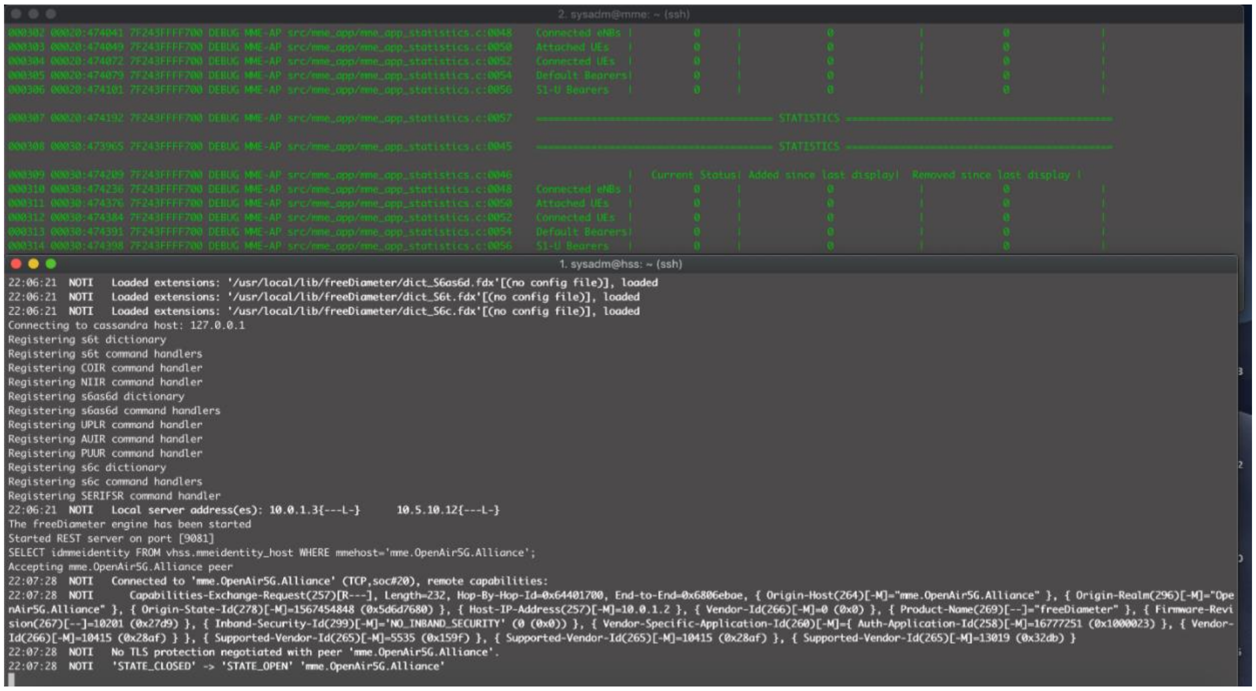
```
$ screen -S spgw_c

$ cd ~/openair-cn-cups/build/scripts

$ sudo spgwc -c /usr/local/etc/oai/spgw_c.conf
```

Text Box 4-26. Test 4 SPGW_C launch.

```
$ screen -S spgw_u

$ cd ~/openair-cn-cups/build/scripts

$ sudo spgwu -c /usr/local/etc/oai/spgw_u.conf
```

Text Box 4-25. Test 4 SPGW_U launch.

It is important to launch each of the services in the indicated order. After this has been done, the following terminals should appear:

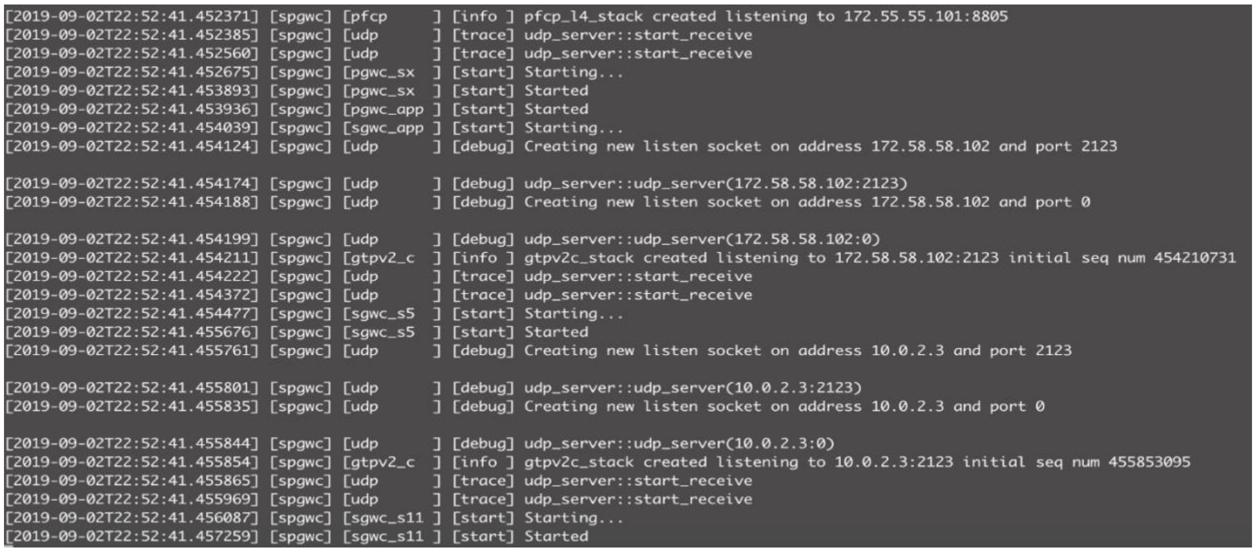


```
2. sysadm@mme: ~ (ssh)
000302 00020:474041 7F243FFFF700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0048   Connected eNBs |      0      |      0      |      0      |
000303 00020:474049 7F243FFFF700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0050   Attached UEs   |      0      |      0      |      0      |
000304 00020:474072 7F243FFFF700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0052   Connected UEs  |      0      |      0      |      0      |
000305 00020:474079 7F243FFFF700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0054   Default Bearers|      0      |      0      |      0      |
000306 00020:474101 7F243FFFF700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0056   S1-U Bearers   |      0      |      0      |      0      |

000307 00020:474192 7F243FFFF700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0057   ================================= STATISTICS =================================

000308 00030:473965 7F243FFFF700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0045   ================================= STATISTICS =================================

000309 00030:474209 7F243FFFF700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0046                  |  Current Status| Added since last display|  Removed since last display |
000310 00030:474236 7F243FFFF700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0048   Connected eNBs |      0      |      0      |      0      |
000311 00030:474376 7F243FFFF700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0050   Attached UEs   |      0      |      0      |      0      |
000312 00030:474384 7F243FFFF700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0052   Connected UEs  |      0      |      0      |      0      |
000313 00030:474391 7F243FFFF700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0054   Default Bearers|      0      |      0      |      0      |
000314 00030:474398 7F243FFFF700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0056   S1-U Bearers   |      0      |      0      |      0      |
```

```
1. sysadm@hss: ~ (ssh)
22:06:21  NOTI   Loaded extensions: '/usr/local/lib/freeDiameter/dict_S6as6d.fdx'[(no config file)], loaded
22:06:21  NOTI   Loaded extensions: '/usr/local/lib/freeDiameter/dict_S6t.fdx'[(no config file)], loaded
22:06:21  NOTI   Loaded extensions: '/usr/local/lib/freeDiameter/dict_S6c.fdx'[(no config file)], loaded
Connecting to cassandra host: 127.0.0.1
Registering s6t dictionary
Registering s6t command handlers
Registering COIR command handler
Registering NIIR command handler
Registering s6as6d dictionary
Registering s6as6d command handlers
Registering UPLR command handler
Registering AUIR command handler
Registering PUUR command handler
Registering s6c dictionary
Registering s6c command handlers
Registering SERIFSR command handler
22:06:21  NOTI   Local server address(es): 10.0.1.3{---L-}      10.5.10.12{---L-}
The freeDiameter engine has been started
Started REST server on port [9081]
SELECT idmmeidentity FROM vhss.mmeidentity_host WHERE mmehost='mme.OpenAir5G.Alliance';
Accepting mme.OpenAir5G.Alliance peer
22:07:28  NOTI   Connected to 'mme.OpenAir5G.Alliance' (TCP,soc#20), remote capabilities:
22:07:28  NOTI      Capabilities-Exchange-Request(257)[R---], Length=232, Hop-By-Hop-Id=0x64401700, End-to-End=0x6806ebae, { Origin-Host(264)[-M]="mme.OpenAir5G.Alliance" }, { Origin-Realm(296)[-M]="Ope
nAir5G.Alliance" }, { Origin-State-Id(278)[-M]=1567454848 (0x5d6d7680) }, { Host-IP-Address(257)[-M]=10.0.1.2 }, { Vendor-Id(266)[-M]=0 (0x0) }, { Product-Name(269)[--]="freeDiameter" }, { Firmware-Revi
sion(267)[--]=10201 (0x27d9) }, { Inband-Security-Id(299)[-M]='NO_INBAND_SECURITY' (0 (0x0)) }, { Vendor-Specific-Application-Id(260)[-M]={ Auth-Application-Id(258)[-M]=16777251 (0x1000023) }, { Vendor-
Id(266)[-M]=10415 (0x28af) } }, { Supported-Vendor-Id(265)[-M]=5535 (0x159f) }, { Supported-Vendor-Id(265)[-M]=10415 (0x28af) }, { Supported-Vendor-Id(265)[-M]=13019 (0x32db) }
22:07:28  NOTI   No TLS protection negotiated with peer 'mme.OpenAir5G.Alliance'.
22:07:28  NOTI   'STATE_CLOSED' -> 'STATE_OPEN' 'mme.OpenAir5G.Alliance'
```

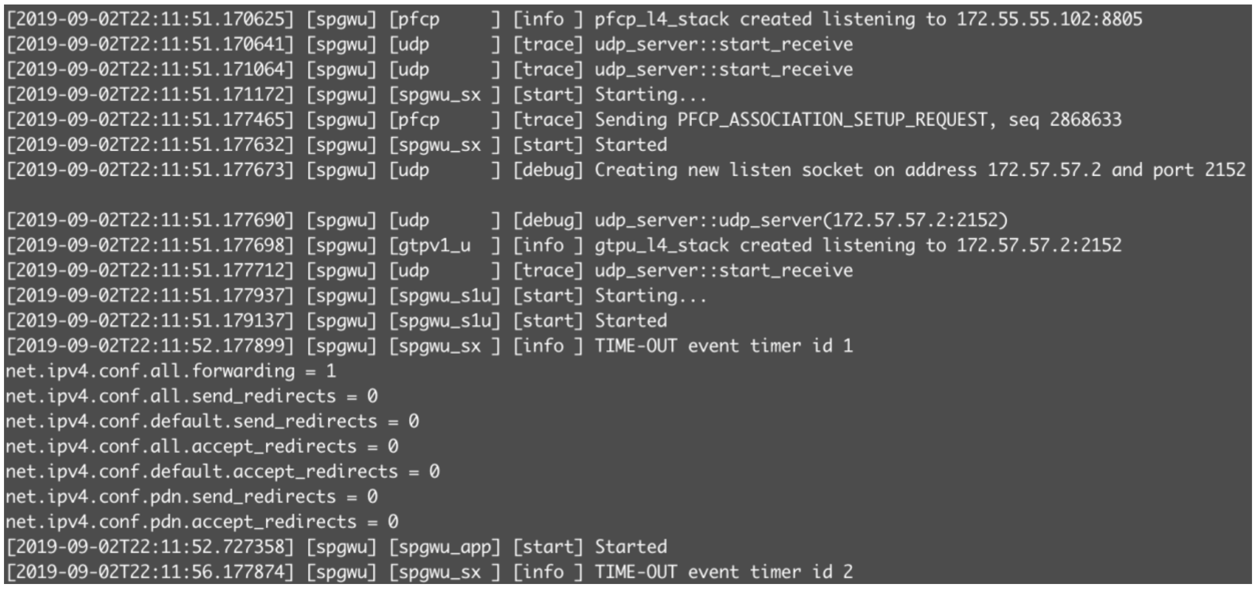
Text Box 4-29. Test 4 running HSS+MME.

In the above figure can be seen how the HSS VNF detects when the MME VNF is up and running as it turns its state from 'STATE_CLOSED' to 'STATE_OPEN - mme.OpenAir5G.Alliance'.



```
[2019-09-02T22:52:41.452371] [spgwc] [pfcp     ] [info ] pfcp_l4_stack created listening to 172.55.55.101:8805
[2019-09-02T22:52:41.452385] [spgwc] [udp      ] [trace] udp_server::start_receive
[2019-09-02T22:52:41.452560] [spgwc] [udp      ] [trace] udp_server::start_receive
[2019-09-02T22:52:41.452675] [spgwc] [pgwc_sx  ] [start] Starting...
[2019-09-02T22:52:41.453893] [spgwc] [pgwc_sx  ] [start] Started
[2019-09-02T22:52:41.453936] [spgwc] [pgwc_app ] [start] Started
[2019-09-02T22:52:41.454039] [spgwc] [sgwc_app ] [start] Starting...
[2019-09-02T22:52:41.454124] [spgwc] [udp      ] [debug] Creating new listen socket on address 172.58.58.102 and port 2123

[2019-09-02T22:52:41.454174] [spgwc] [udp      ] [debug] udp_server::udp_server(172.58.58.102:2123)
[2019-09-02T22:52:41.454188] [spgwc] [udp      ] [debug] Creating new listen socket on address 172.58.58.102 and port 0

[2019-09-02T22:52:41.454199] [spgwc] [udp      ] [debug] udp_server::udp_server(172.58.58.102:0)
[2019-09-02T22:52:41.454211] [spgwc] [gtpv2_c  ] [info ] gtpv2c_stack created listening to 172.58.58.102:2123 initial seq num 454210731
[2019-09-02T22:52:41.454222] [spgwc] [udp      ] [trace] udp_server::start_receive
[2019-09-02T22:52:41.454372] [spgwc] [udp      ] [trace] udp_server::start_receive
[2019-09-02T22:52:41.454477] [spgwc] [sgwc_s5  ] [start] Starting...
[2019-09-02T22:52:41.455676] [spgwc] [sgwc_s5  ] [start] Started
[2019-09-02T22:52:41.455761] [spgwc] [udp      ] [debug] Creating new listen socket on address 10.0.2.3 and port 2123

[2019-09-02T22:52:41.455801] [spgwc] [udp      ] [debug] udp_server::udp_server(10.0.2.3:2123)
[2019-09-02T22:52:41.455835] [spgwc] [udp      ] [debug] Creating new listen socket on address 10.0.2.3 and port 0

[2019-09-02T22:52:41.455844] [spgwc] [udp      ] [debug] udp_server::udp_server(10.0.2.3:0)
[2019-09-02T22:52:41.455854] [spgwc] [gtpv2_c  ] [info ] gtpv2c_stack created listening to 10.0.2.3:2123 initial seq num 455853095
[2019-09-02T22:52:41.455865] [spgwc] [udp      ] [trace] udp_server::start_receive
[2019-09-02T22:52:41.455969] [spgwc] [udp      ] [trace] udp_server::start_receive
[2019-09-02T22:52:41.456087] [spgwc] [sgwc_s11 ] [start] Starting...
[2019-09-02T22:52:41.457259] [spgwc] [sgwc_s11 ] [start] Started
```

Figure 4-53. Test 4 running SPGW_C.



```
[2019-09-02T22:11:51.170625] [spgwu] [pfcp     ] [info ] pfcp_l4_stack created listening to 172.55.55.102:8805
[2019-09-02T22:11:51.170641] [spgwu] [udp      ] [trace] udp_server::start_receive
[2019-09-02T22:11:51.171064] [spgwu] [udp      ] [trace] udp_server::start_receive
[2019-09-02T22:11:51.171172] [spgwu] [spgwu_sx ] [start] Starting...
[2019-09-02T22:11:51.177465] [spgwu] [pfcp     ] [trace] Sending PFCP_ASSOCIATION_SETUP_REQUEST, seq 2868633
[2019-09-02T22:11:51.177632] [spgwu] [spgwu_sx ] [start] Started
[2019-09-02T22:11:51.177673] [spgwu] [udp      ] [debug] Creating new listen socket on address 172.57.57.2 and port 2152

[2019-09-02T22:11:51.177690] [spgwu] [udp      ] [debug] udp_server::udp_server(172.57.57.2:2152)
[2019-09-02T22:11:51.177698] [spgwu] [gtpv1_u  ] [info ] gtpu_l4_stack created listening to 172.57.57.2:2152
[2019-09-02T22:11:51.177712] [spgwu] [udp      ] [trace] udp_server::start_receive
[2019-09-02T22:11:51.177937] [spgwu] [spgwu_s1u] [start] Starting...
[2019-09-02T22:11:51.179137] [spgwu] [spgwu_s1u] [start] Started
[2019-09-02T22:11:52.177899] [spgwu] [spgwu_sx ] [info ] TIME-OUT event timer id 1
net.ipv4.conf.all.forwarding = 1
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.pdn.send_redirects = 0
net.ipv4.conf.pdn.accept_redirects = 0
[2019-09-02T22:11:52.727358] [spgwu] [spgwu_app] [start] Started
[2019-09-02T22:11:56.177874] [spgwu] [spgwu_sx ] [info ] TIME-OUT event timer id 2
```

Figure 4-54. Test 4 running SPGW_U.

In conclusion, this experiment was a complete success as all the previous acquired knowledge could be applied and, additionally, a complete automated vEPC is deployed in a matter of minutes. In fact, the whole process was timed, obtaining the following results:

STAGE	STAGE TIME	TOTAL TIME
OPENSTACK VM INSTANTIATION	1 min 6.63 s	1 min 6.63 s
LXD CONTAINER DEPLOYMENT	33.35 s	1 min 39.99 s
LXD COMPLETE CONFIGURATION	4 min 9.6 s	5 min 49.59 s
DAY-1 CONFIGURATION	15 min 10.83 s	21 min 0.42 s

Table 4-8. Test 4 timing.

The total time from the first manual trials has been drastically diminished, from more than 8h to approximately 21min.

4.5 Test 5: NSA E2E 5G implementation

This final test aims at generating the final objective of this Master Thesis, an NSA E2E sliced 5G environment with the OAI components orchestrated by OSM. To achieve this task, Test 4 vEPC will be used as described in the previous section and it will be one of the two Network Services that will compose the NS. Besides, a Network Service will be generated containing the eNB module. The main difference with the previous tests is that the resulting environment will use not only virtualized devices (VNFs) but also one physical device as a PNF[64] to build the final test. Furthermore, OSM NS capabilities will be measured and tested in this test; this point is important because even though NS support is available in OSM Rel. FIVE it is the only element on its IM that does not follow 3GPP specifications, consequently, it is still not 3GPP aligned and not standardized. Basically, the ETSI-NFV and ETSI-MANO WGs have tried to stablish a relationship between the 3GPP NS concept and this same concept in the ETSI GR NFV-EVE 012 document, in order to implement OSM NST. In *Figure 4-55. 3GPP NS vs. ETSI-NFV NS*. the parallelism can be seen:

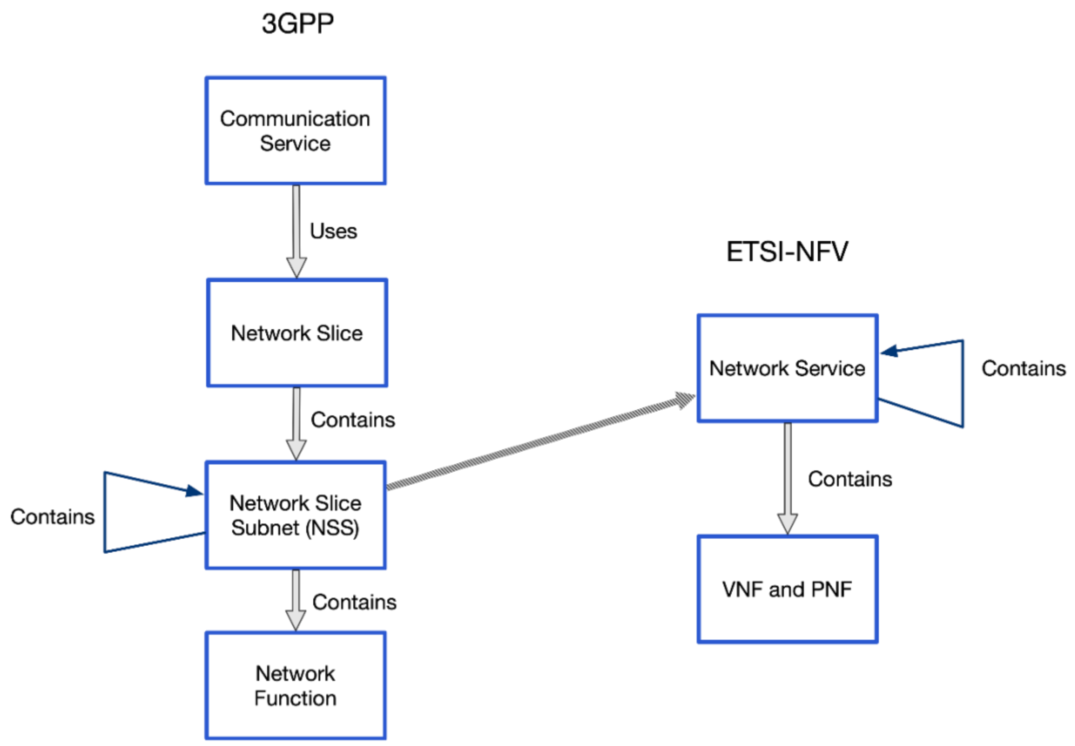


Figure 4-55. 3GPP NS vs. ETSI-NFV NS.

64 See section 9.2.1 Hardware “remote server” device, to obtain the machine characteristics.

Two relationships are directly stablished in order to implement a NS above the ETSI-NFV Network Service concept:

- NSS can be considered as a Network Service.
- NF is equivalent to VNFs and PNFs.

A NS definition has been defined in *section 2.4.1 Definition* as:

"Network Slicing is a technology that splits the E2E common/shared underlying network infrastructure, physical or virtual, into mutually isolated, optimized (NFs, topology, resources, management…), with independent control and management, logical networks that can be created on-demand to satisfy sets of given requirements and a negotiated service quality."

Taking this definition into account and having in mind the different types of NS that are already defined and the kind of usage that is pursued in this project, it can be said that, in this test an eMBB NS will be deployed.

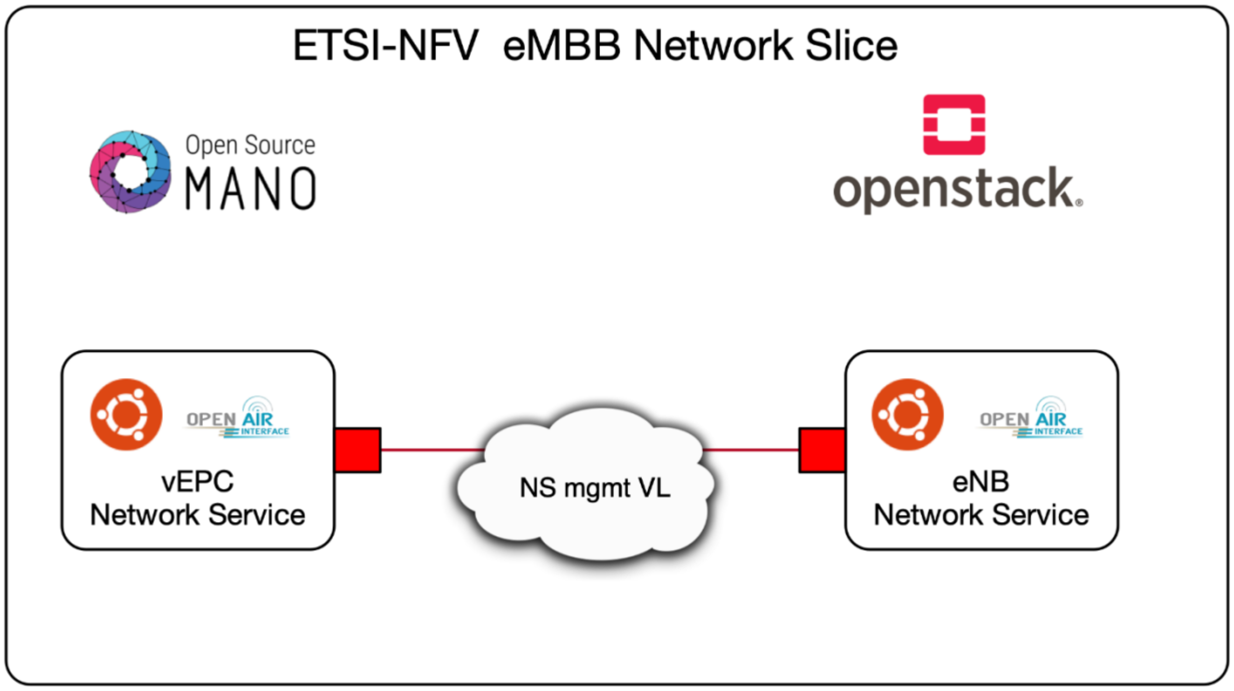


Figure 4-56. Test 5 eMBB NS high level view.

4.5.1 Test Summary

This section provides a summary of the current Test Characteristics.

Test 3	
Scenario	Production 2
VNFs	3
VNFDs	3
NSs	2
NSDs	2
PNFs	1
PNFDs	1
NSTs	1
Networks	5
Total RAM	20 GB
Total vCPUS	16
Total Disk	632 GB
Day-0 Config	Yes
Day-1 Config	Yes (Ansible)
Day-2 Config	No

Table 4-9. Test 5 summary.

It is important to clarify that the resources have been drastically increased not because the eNB really needs them but, because the server PNF total resources have been added to the summary (the PNF will have access to all the system resources).

4.5.2 Test Description

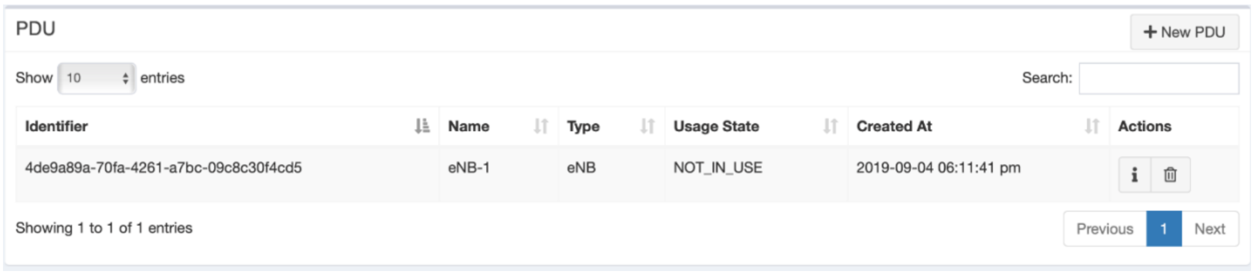
This last involves the usage of two new tools inside the OSM Rel. FIVE platform: a PNF instead of a VNF (acting as the OAI eNB) and a NST.

A PNF is sourced by a physical machine, consequently, to create a PDU in OSM Rel. FIVE no VDU should be used directly and, so, the user has to select the "*PDU Instances*" tag on the sidebar and press the upper-right button "*New PDU*" this will pop-up a window with a text editor where, in this case, the following content should be added:

```
name:            eNB-1
description:     Enhanced Node-B
type:            eNB
vim_accounts:    [ 17d82221-3782-4bf1-b78e-d2a7884e7a07 ]
shared:          false
interfaces:
 -  name:        eno1
    ip-address: 10.5.10.15
    vim-network-name: PUBLIC
    mgmt:        true
```

Text Box 4-30. Test 5 PDU Instance descriptor.

Basically, it is a formal document that indicates the number of network interfaces, their IPs, the VIM networks (in this case the "public" network) to which the machine is connected and allows the orchestrator to recognize the PNF as an accessible device. Once the document is edited and uploaded, the following should be seen in the "*PDU Instances*" tag:



Identifier	Name	Type	Usage State	Created At	Actions
4de9a89a-70fa-4261-a7bc-09c8c30f4cd5	eNB-1	eNB	NOT_IN_USE	2019-09-04 06:11:41 pm	

Text Box 4-31. Test 5 PDU upload verification.

In order to generate a descriptor for a PNF, the process is similar to the one of a VNF (it is also generated using "*devops tools*") but with some exceptions:

- VDU metadata field should not contain a *"flavor"* sub-metadata field.
- Day-0 configuration is not accepted as the machine is supposed to be already configured and with an OS.
- Day-1 configuration may be used.

Taking this into account, the proper PNFD was developed (see *Annex 11.9 eNB PNFD - Test 5*) and the following changes were added compared to a VNFD in the *VDU* section:

```
vdu:
        -   id: eNB_pdu
            description: eNB PDU
            interface:
            -   name: eno1
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
                external-connection-point-ref: mgmt
            pdu-type: eNB
```

Text Box 4-32. Test 5 PNFD.

The last line is the one that tells OSM RO to use the PDU unit named "eNB". Afterwards, the eNB NSD can be generated with no difference compared with a VNF NSD (see *Annex 12.6 eNB NSD - Test 5)*. In fact, the resulting NSD was simple and only one extra metadata field had to be added, in both NSDs (vEPC and eNB), in order to be connected by the NS VL:

```
connection-point:
        -   name: eNB_cp_mgmt
            vld-id-ref: eNB_pnf_nsd_mgmt
```

Text Box 4-33. Test 5 eNB NSD extra CP metadata field.

```
connection-point:
        -   name: epc_cp_mgmt
            vld-id-ref: management
```

Text Box 4-34. Test 5 EPC NSD extra CP metadata field.

Finally, as both of the required Network Services are prepared, the NST can be developed. In the following text-boxes its main features are going to be explained, if the full file content is desired (it includes an Apache v.2 License) see *Annex 13.1 5G NSA OAI NST - Test 5.*

```
nst:

-   id: 5g_na_nst

    name: 5g_na_nst

    SNSSAI-identifier:

        slice-service-type: eMBB

    quality-of-service:

        id: 1
```

Text Box 4-35. Test 5 NST part 1.

The section above allows the user to identify the NS by an ID and a name, quality of service policies may be defined (currently this metadata field was in development for OSM Rel. FIVE and implemented in OSM Rel. SIX) and the NS service type could be selected from: eMBB, uRLLC and mMTC.

```
netslice-subnet:

    -   id: eNB_ns

        is-shared-nss: 'false'

        description: NetSlice Subnet (service) composed by 1 pnf (eNB) and 1 cp
(mgmt)

        nsd-ref: eNB_pnf_nsd

    -   id: vEPC_ns

        is-shared-nss: 'false'

        description: NetSlice Subnet (service) composed by 3 vnfs and 8 cp (3 mgmt
and 5 data)

        nsd-ref: epc_ns
```

Text Box 4-36. Test 5 NST part 2.

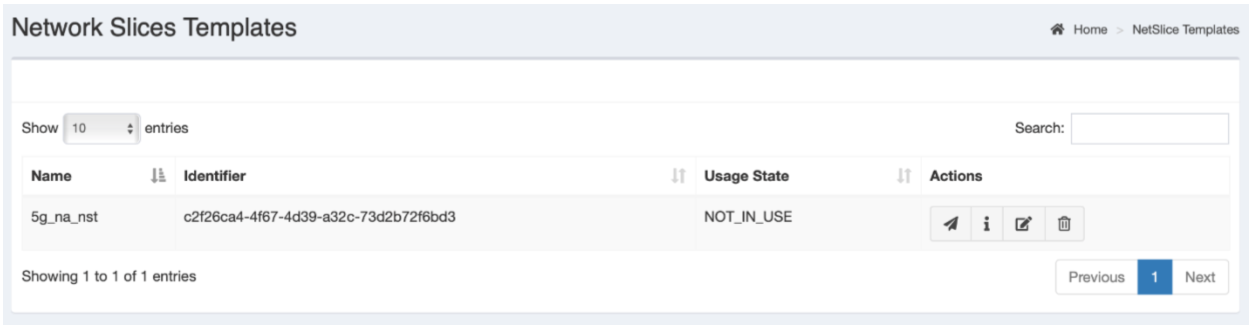
This past section is, probably, the most important one in the NST as it is the one that relates the ETSI-NFV/MANO NS concept with the 3GPP concept through the before-mentioned relation between NSSs and OSM Network Services. Here, each Network Service is “encapsulated” as a NSS.

To come to an end with the NST, it only remains to be defined how both NSSs are going to be connected. To achieve this objective, OSM treats them as a Network Service composed by other

Network Services so that it can connect them using a VL (that's the reason why the two CPs had to be defined in the NSDs):

```
netslice-vld:
    -   id: vld_mgmt
        name: vld_mgmt
        type: ELAN
        mgmt-network: 'true'
        nss-connection-point-ref:
        -   nss-ref: eNB
            nsd-connection-point-ref: eNB_cp_mgmt
        -   nss-ref: vEPC
            nsd-connection-point-ref: epc_cp_mgmt
```

Text Box 4-37. Test 5 NST part 3.



Name	Identifier	Usage State	Actions
5g_na_nst	c2f26ca4-4f67-4d39-a32c-73d2b72f6bd3	NOT_IN_USE	

Showing 1 to 1 of 1 entries

Figure 4-57. Test 5 OSM NST.

It is important to remark that this test was not intended to be launched, it was planned just to develop the proper code for each of the required files as the USRP SW radio cards were not available due to a huge delay with the provider and the *remote servers* devices were not expected to arrive until mid-September. However, both *remote servers* arrived much earlier than expected, by the last week of August, and as some time was left before this Master Thesis delivery date was over, it was decided to give it a quick try. Furthermore, it was discovered that, even though the 5TONIC datacenter optic fiber could support speeds up to 10Gbps, the ToR switches had network interfaces of only 1Gbps.

On the other hand, a new Ansible-Playbook was developed to perform the Day-1 configuration of the eNB PNF. Here its purpose and configuration functionalities are explained, to see the full document go to *Annex 26014.5 eNB Playbook -Test 5:*

1. Update and Upgrade package repositories and packages.

2. Install required USRP drivers and SW dependencies.
3. Download OAI GitHub repository.
4. Install and build eNB for USRP HW.
5. Modify USRP-210 configuration file:
    a. Debugging level is raised to "info" to receive more detailed information when the program is running.
    b. Configure MCC and MNC with the same data as in the MME VNF.
    c. Configure eNB S1-U and S1-C interfaces to match the desired Network architecture.
6. Create required network sub-interfaces.

The selected configuration file will run the eNB in LTE Band-7 with a channel Bandwidth of 10MHZ, consequently:

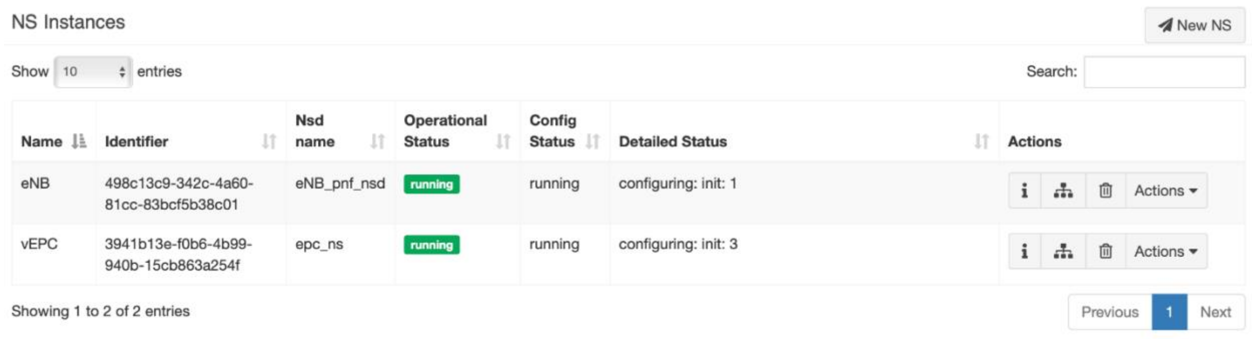
- 50 Physical Resource Block (PRB) are obtained per frame slot.
- 1000 PRBs are available per frame.
- 600 subcarriers may be used.
- FFT size: 1024.

4.5.3 Test Results

The results obtained in this test, as it is going to be shown below, are clearly unsatisfactory. OSM Rel. FIVE states that it has support for NS, but it was discovered that it was not prepared to support hybrid NS composed by PNFs and VNFs and similar problems were encountered when the same test was deployed using only a virtualized environment with the eNB defined as a VNF.

4.5.3.1 Trial 1: Launch both Network Services individually

The first step was to verify if both Network Services could be launched independently without any issue. OAI vEPC was verified in the previous test, nonetheless, it was launched again in this first trial to check OSM performance with multiple Network Services running Day-1 Configuration.


NS Instances

Show 10 entries   Search:

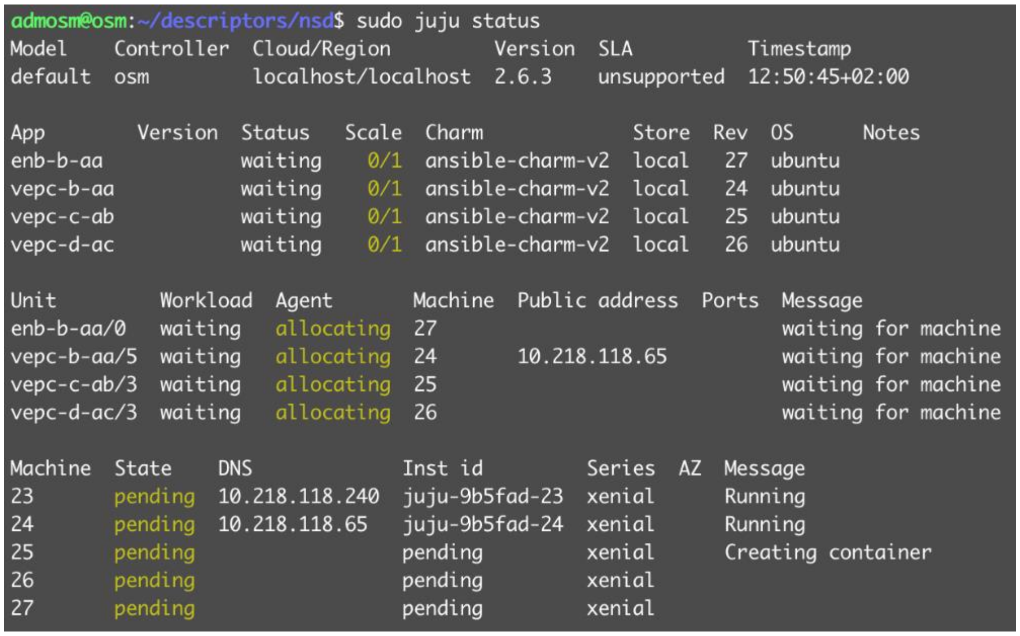
Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
eNB	498c13c9-342c-4a60-81cc-83bcf5b38c01	eNB_pnf_nsd	running	running	configuring: init: 1	Actions
vEPC	3941b13e-f0b6-4b99-940b-15cb863a254f	epc_ns	running	running	configuring: init: 3	Actions

Showing 1 to 2 of 2 entries

Text Box 4-38. Test 5 - Trial 1, Launching Network Services.

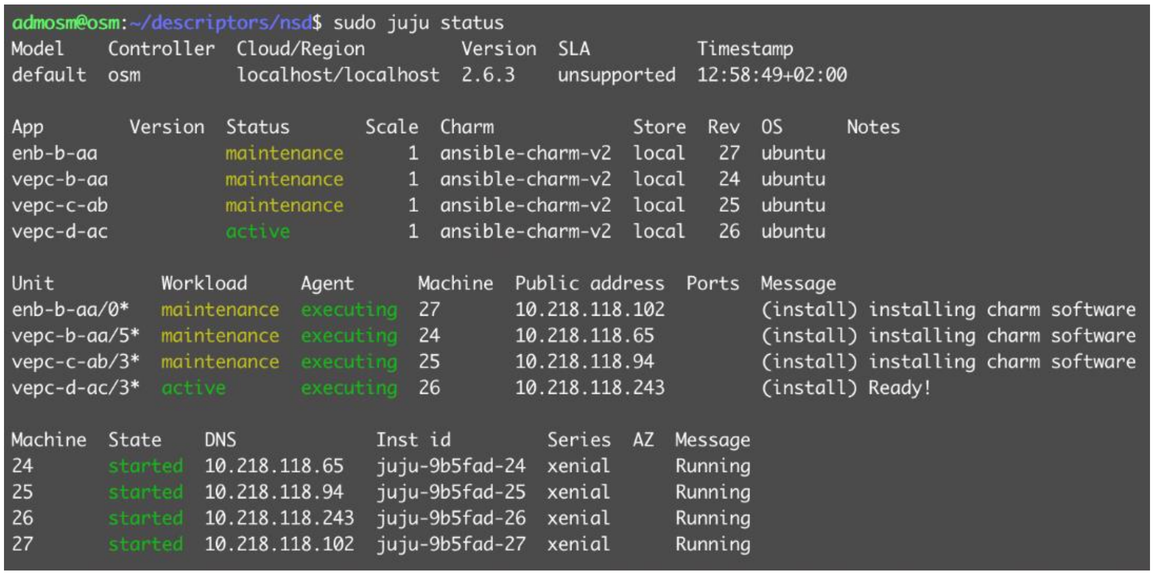
Both Network services were "instantiated" (the PNF cannot be instantiated as is it already running) by OSM RO correctly. However, the Day-1 configuration process suffered a huge delay compared with the previous test, it was observed that the LXD container deployment was much worse in time. The rest of the juju proxy charms stages seemed to be affected by the first stage delay[65].

[65] eNB compilation and building processes are, by far, the most time-consuming ones but, as different LXD containers are generated by the juju platform, it is expected that each configuration process runs in parallel and independently from other processes workload. Obtained results do not probe this behavior.



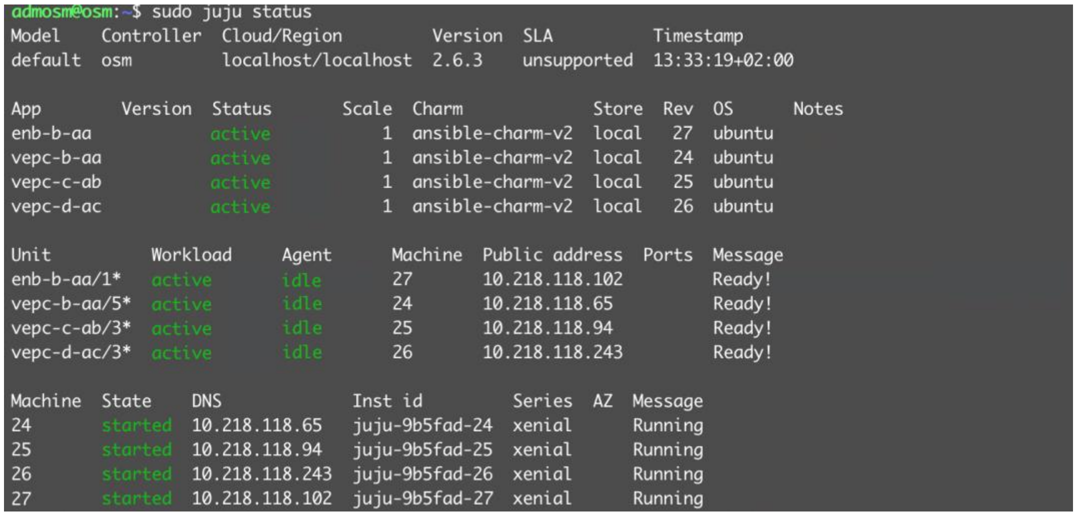
```
admosm@osm:~/descriptors/nsd$ sudo juju status
Model    Controller  Cloud/Region         Version  SLA          Timestamp
default  osm         localhost/localhost  2.6.3    unsupported  12:50:45+02:00

App        Version  Status   Scale  Charm            Store  Rev  OS      Notes
enb-b-aa            waiting    0/1  ansible-charm-v2  local   27  ubuntu
vepc-b-aa           waiting    0/1  ansible-charm-v2  local   24  ubuntu
vepc-c-ab           waiting    0/1  ansible-charm-v2  local   25  ubuntu
vepc-d-ac           waiting    0/1  ansible-charm-v2  local   26  ubuntu

Unit         Workload  Agent       Machine  Public address  Ports  Message
enb-b-aa/0   waiting   allocating  27                              waiting for machine
vepc-b-aa/5  waiting   allocating  24       10.218.118.65          waiting for machine
vepc-c-ab/3  waiting   allocating  25                              waiting for machine
vepc-d-ac/3  waiting   allocating  26                              waiting for machine

Machine  State    DNS            Inst id         Series  AZ  Message
23       pending  10.218.118.240 juju-9b5fad-23  xenial      Running
24       pending  10.218.118.65  juju-9b5fad-24  xenial      Running
25       pending                 pending         xenial      Creating container
26       pending                 pending         xenial
27       pending                 pending         xenial
```

Figure 4-58. Test 5 - Trial 1 LXD container deployment.



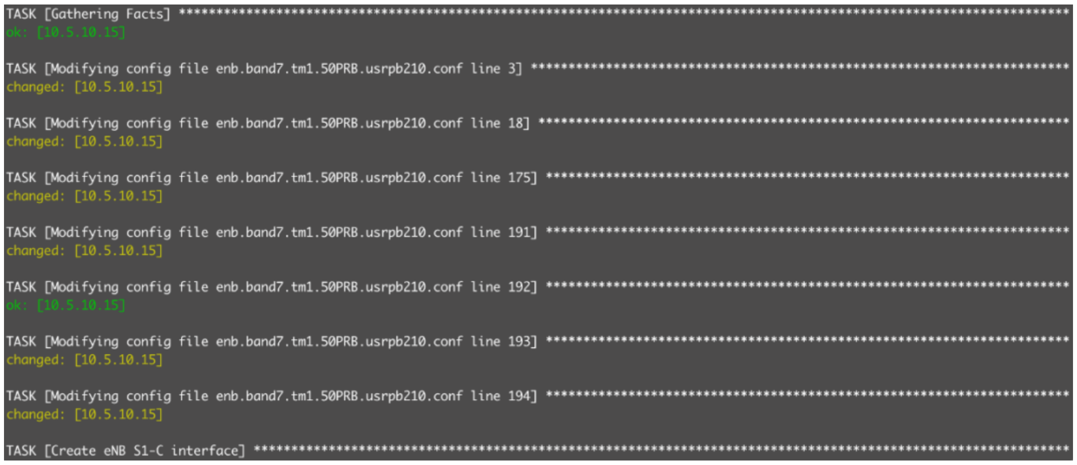
```
admosm@osm:~/descriptors/nsd$ sudo juju status
Model    Controller  Cloud/Region         Version  SLA          Timestamp
default  osm         localhost/localhost  2.6.3    unsupported  12:58:49+02:00

App        Version  Status       Scale  Charm             Store  Rev  OS      Notes
enb-b-aa            maintenance      1  ansible-charm-v2  local   27  ubuntu
vepc-b-aa           maintenance      1  ansible-charm-v2  local   24  ubuntu
vepc-c-ab           maintenance      1  ansible-charm-v2  local   25  ubuntu
vepc-d-ac           active           1  ansible-charm-v2  local   26  ubuntu

Unit          Workload     Agent      Machine  Public address  Ports  Message
enb-b-aa/0*   maintenance  executing  27       10.218.118.102         (install) installing charm software
vepc-b-aa/5*  maintenance  executing  24       10.218.118.65          (install) installing charm software
vepc-c-ab/3*  maintenance  executing  25       10.218.118.94          (install) installing charm software
vepc-d-ac/3*  active       executing  26       10.218.118.243         (install) Ready!

Machine  State    DNS             Inst id         Series  AZ  Message
24       started  10.218.118.65   juju-9b5fad-24  xenial      Running
25       started  10.218.118.94   juju-9b5fad-25  xenial      Running
26       started  10.218.118.243  juju-9b5fad-26  xenial      Running
27       started  10.218.118.102  juju-9b5fad-27  xenial      Running
```

Figure 4-59. Test 5 - Trial 1 SW installation.



```
admosm@osm:~$ sudo juju status
Model    Controller  Cloud/Region         Version  SLA          Timestamp
default  osm         localhost/localhost  2.6.3    unsupported  13:33:19+02:00

App        Version  Status       Scale  Charm              Store  Rev  OS      Notes
enb-b-aa            active           1  ansible-charm-v2  local   27  ubuntu
vepc-b-aa           active           1  ansible-charm-v2  local   24  ubuntu
vepc-c-ab           active           1  ansible-charm-v2  local   25  ubuntu
vepc-d-ac           active           1  ansible-charm-v2  local   26  ubuntu

Unit          Workload  Agent      Machine  Public address  Ports  Message
enb-b-aa/1*   active    idle       27       10.218.118.102         Ready!
vepc-b-aa/5*  active    idle       24       10.218.118.65          Ready!
vepc-c-ab/3*  active    idle       25       10.218.118.94          Ready!
vepc-d-ac/3*  active    idle       26       10.218.118.243         Ready!

Machine  State    DNS             Inst id         Series  AZ  Message
24       started  10.218.118.65   juju-9b5fad-24  xenial      Running
25       started  10.218.118.94   juju-9b5fad-25  xenial      Running
26       started  10.218.118.243  juju-9b5fad-26  xenial      Running
27       started  10.218.118.102  juju-9b5fad-27  xenial      Running
```

Figure 4-60. Test 5 - Trial 1 Day-1 configuration process end.



```
TASK [Gathering Facts] ****************************************************************************************************
ok: [10.5.10.15]

TASK [Modifying config file enb.band7.tm1.50PRB.usrpb210.conf line 3] *****************************************************
changed: [10.5.10.15]

TASK [Modifying config file enb.band7.tm1.50PRB.usrpb210.conf line 18] ****************************************************
changed: [10.5.10.15]

TASK [Modifying config file enb.band7.tm1.50PRB.usrpb210.conf line 175] ***************************************************
changed: [10.5.10.15]

TASK [Modifying config file enb.band7.tm1.50PRB.usrpb210.conf line 191] ***************************************************
changed: [10.5.10.15]

TASK [Modifying config file enb.band7.tm1.50PRB.usrpb210.conf line 192] ***************************************************
ok: [10.5.10.15]

TASK [Modifying config file enb.band7.tm1.50PRB.usrpb210.conf line 193] ***************************************************
changed: [10.5.10.15]

TASK [Modifying config file enb.band7.tm1.50PRB.usrpb210.conf line 194] ***************************************************
changed: [10.5.10.15]

TASK [Create eNB S1-C interface] ******************************************************************************************
```

Figure 4-61. Test 5 - Trial 1 eNB Ansible Playbook debug-log.

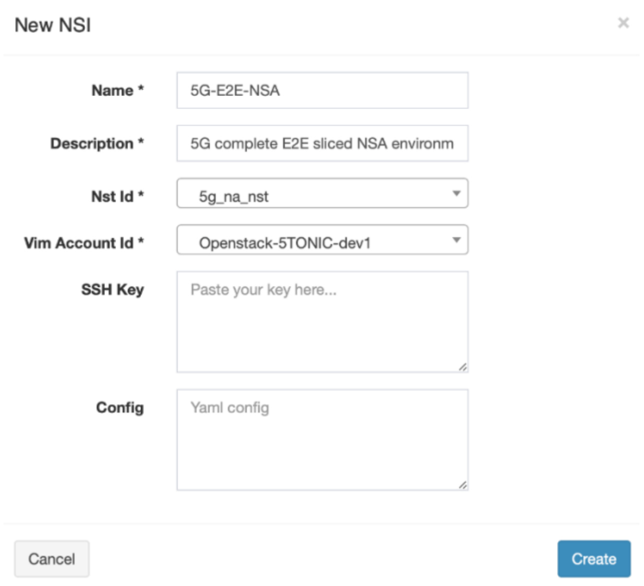
As it can be seen, even without an exact timing report, the difference between the timestamp of the *Figure 4-58. Test 5 - Trial 1 LXD container deployment*. and the one of the *Figure 4-60. Test 5 - Trial 1 Day-1 configuration process end*. exceeds the 40 minutes length which is a remarkable difference if compared with the 21 minutes of the solo vEPC Network service. Nonetheless, both Network Services work so, in theory, the NS is ready to be deployed.

NOTE: The eNB, although configured, cannot be run because it crashes if a USRP is not attached to one of the PNF USB-3 ports.

4.5.3.2 Trial 2: Launch hybrid NS

As the previous trial was satisfactory and both Network Services work properly, finally, a NS could be launched. The objective of this trial is to test OSM hybrid NS capabilities.

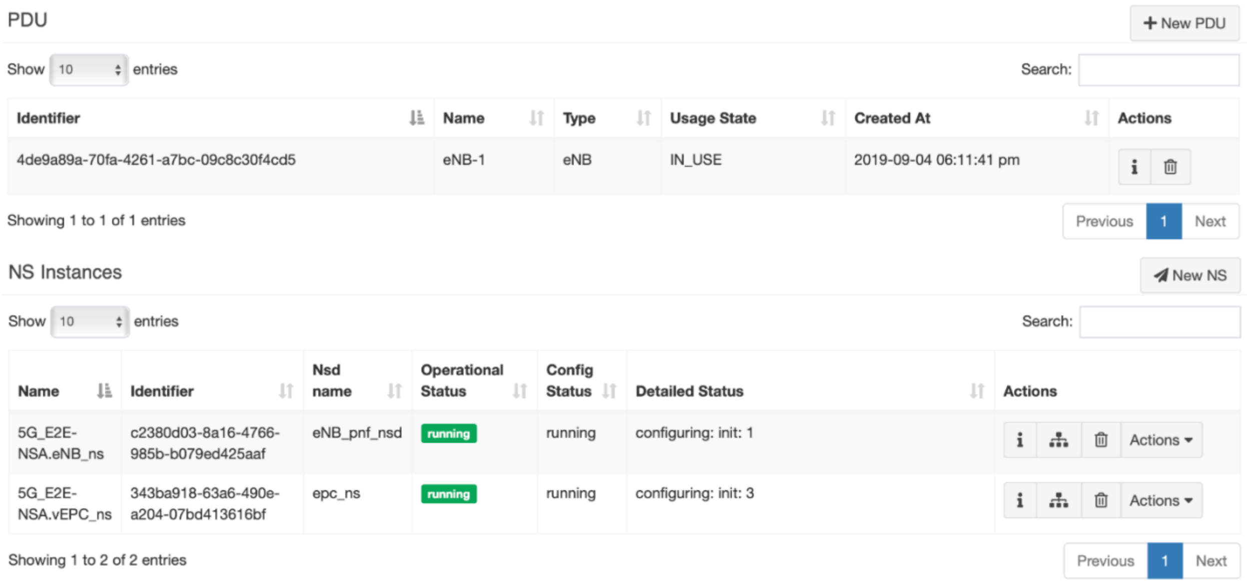
In order to launch a NS in OSM Rel. FIVE the user has to go to the NST tag and select the desired NST, pressing the "*Launch Button*". Once this action is carried out, the following dialogue will pop-up:



New NSI

Name *	5G-E2E-NSA
Description *	5G complete E2E sliced NSA environm
Nst Id *	5g_na_nst
Vim Account Id *	Openstack-5TONIC-dev1
SSH Key	Paste your key here...
Config	Yaml config

Figure 4-63. Test 5 - Trial 2: NST Launching Form.



PDU

Show 10 entries

Identifier	Name	Type	Usage State	Created At	Actions
4de9a89a-70fa-4261-a7bc-09c8c30f4cd5	eNB-1	eNB	IN_USE	2019-09-04 06:11:41 pm	

Showing 1 to 1 of 1 entries

NS Instances

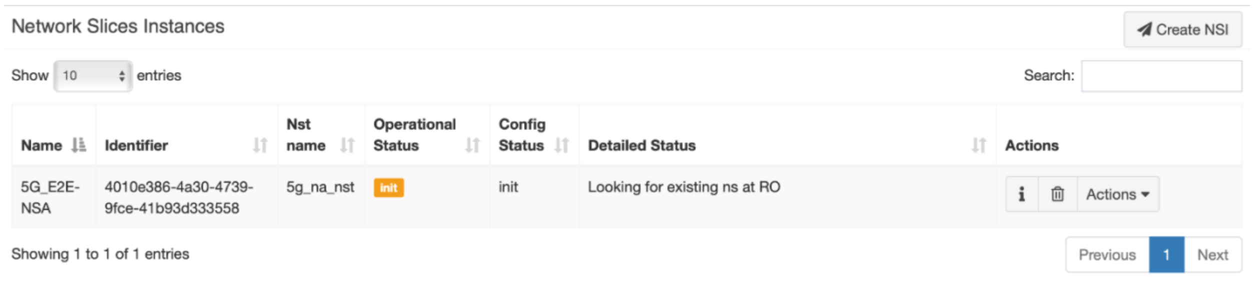
Show 10 entries

Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
5G_E2E-NSA.eNB_ns	c2380d03-8a16-4766-985b-b079ed425aaf	eNB_pnf_nsd	running	running	configuring: init: 1	Actions
5G_E2E-NSA.vEPC_ns	343ba918-63a6-490e-a204-07bd413616bf	epc_ns	running	running	configuring: init: 3	Actions

Showing 1 to 2 of 2 entries

Figure 4-62. Test 5 - Trial 1: Running PDU and Network Services.

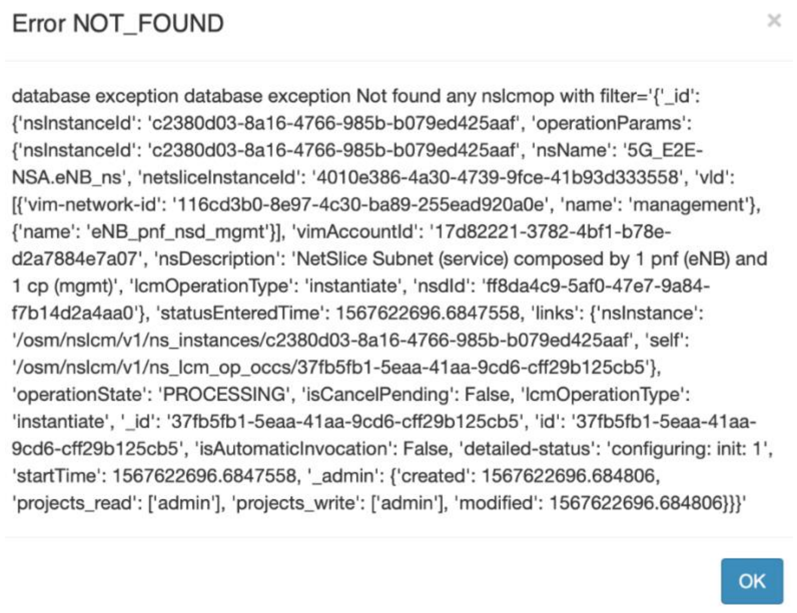
In the above figure can be observed that both Network Services have instantiated correctly and that the defined PDU is being used by one of those services (*"5G_E2E-NSA.eNB_ns"*). OpenStack created the proper networks, and everything seemed to work as expected; nevertheless, after a couple of minutes the "*sudo juju status*" command showed no activity, as if no LXD container was being deployed or in the process to be deployed. The OSM NST Instance tag was revisited to verify if something went wrong and the following status was observed for more than half an hour before deleting the NS:



Name	Identifier	Nst name	Operational Status	Config Status	Detailed Status	Actions
5G_E2E-NSA	4010e386-4a30-4739-9fce-41b93d333558	5g_na_nst	init	init	Looking for existing ns at RO	Actions

Figure 4-64. Test 5 - Trial 2: NSI status.

The status reports a state were one or both Network Services had not been instantiated by OSM RO but, as it can be checked in the previous figure, both Network Services were deployed and running. Besides, it looked like if the NST state does not evolve, no Day-1 configuration actions are triggered as no juju proxy charms or LXD containers were deployed. After a long while waiting and expecting something, nothing new occurred and it was decided to delete the NS, just after doing so, the following error popped-up:



```
database exception database exception Not found any nslcmop with filter='{'_id':
{'nsInstanceId': 'c2380d03-8a16-4766-985b-b079ed425aaf', 'operationParams':
{'nsInstanceId': 'c2380d03-8a16-4766-985b-b079ed425aaf', 'nsName': '5G_E2E-
NSA.eNB_ns', 'netsliceInstanceId': '4010e386-4a30-4739-9fce-41b93d333558', 'vld':
[{'vim-network-id': '116cd3b0-8e97-4c30-ba89-255ead920a0e', 'name': 'management'},
{'name': 'eNB_pnf_nsd_mgmt'}], 'vimAccountId': '17d82221-3782-4bf1-b78e-
d2a7884e7a07', 'nsDescription': 'NetSlice Subnet (service) composed by 1 pnf (eNB) and
1 cp (mgmt)', 'lcmOperationType': 'instantiate', 'nsdId': 'ff8da4c9-5af0-47e7-9a84-
f7b14d2a4aa0'}, 'statusEnteredTime': 1567622696.6847558, 'links': {'nsInstance':
'/osm/nslcm/v1/ns_instances/c2380d03-8a16-4766-985b-b079ed425aaf', 'self':
'/osm/nslcm/v1/ns_lcm_op_occs/37fb5fb1-5eaa-41aa-9cd6-cff29b125cb5'},
'operationState': 'PROCESSING', 'isCancelPending': False, 'lcmOperationType':
'instantiate', '_id': '37fb5fb1-5eaa-41aa-9cd6-cff29b125cb5', 'id': '37fb5fb1-5eaa-41aa-
9cd6-cff29b125cb5', 'isAutomaticInvocation': False, 'detailed-status': 'configuring: init: 1',
'startTime': 1567622696.6847558, '_admin': {'created': 1567622696.684806,
'projects_read': ['admin'], 'projects_write': ['admin'], 'modified': 1567622696.684806}}}'
```

Figure 4-65. Test 5 - Trial 2: NS Instance error.

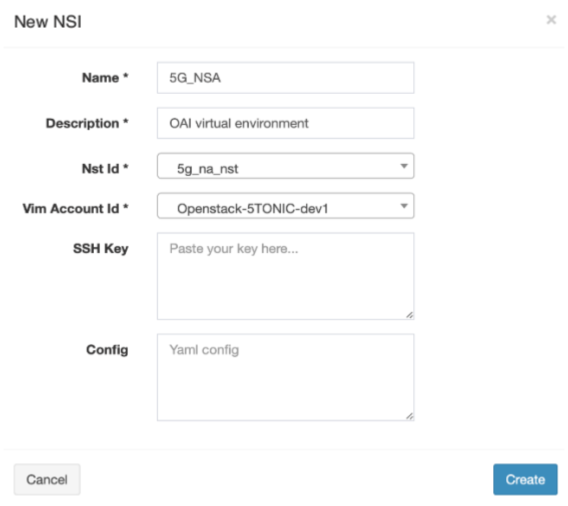
This error reports that the eNB Network Service was not found in the OSM RO, again, this made no sense as the eNB Network Service was clearly up and running. After a day trying to the ghost issue: syntax error, VLDs errors, descriptors errors… None was found and, therefore, an email was sent to the ETSI-MANO technical e-mail list to verify if OSM Rel. FIVE supports hybrid NS.

A vague answer was received recommending the author of this project to migrate OSM Rel. FIVE to OSM Rel. SIX because several issues were reported for OSM Rel. FIVE and its recent NS supporting capabilities (no distinction was made between hybrid NS and virtualized NS).

Thereupon, some digging was carried out throughout OSM files, but no solution was found and, consequently, this trial has ended as a FAIL, as the program used, OSM Rel. FIVE does not support the expected features and capabilities.

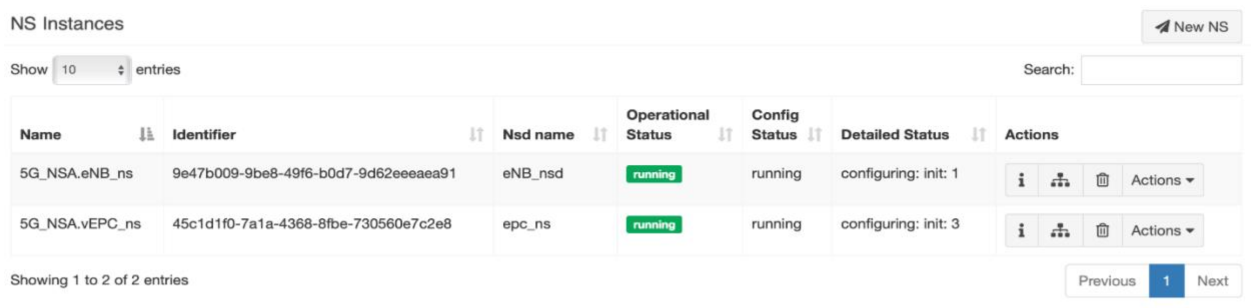
4.5.3.3 Trial 3: Launch virtualized NS

This trial was carried out merely to verify if a completely virtualized NS could be launched replacing the eNB PNF with a VNF with similar resources.


New NSI

Name *	5G_NSA
Description *	OAI virtual environment
Nst Id *	5g_na_nst
Vim Account Id *	Openstack-5TONIC-dev1
SSH Key	Paste your key here...
Config	Yaml config

Figure 4-66. Test 5 - Trial 3: NS launching form.

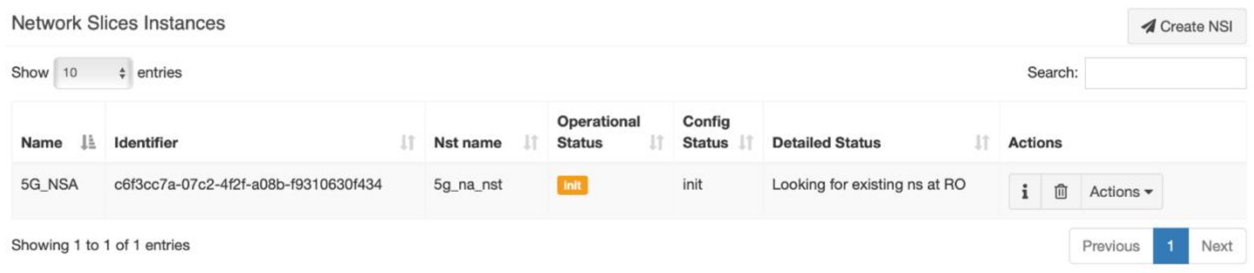

NS Instances

Show 10 entries

Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
5G_NSA.eNB_ns	9e47b009-9be8-49f6-b0d7-9d62eeeaea91	eNB_nsd	running	running	configuring: init: 1	Actions
5G_NSA.vEPC_ns	45c1d1f0-7a1a-4368-8fbe-730560e7c2e8	epc_ns	running	running	configuring: init: 3	Actions

Showing 1 to 2 of 2 entries

Figure 4-67. Test 5 - Trial 3: Network Services Status.



Name	Identifier	Nst name	Operational Status	Config Status	Detailed Status	Actions
5G_NSA	c6f3cc7a-07c2-4f2f-a08b-f9310630f434	5g_na_nst	init	init	Looking for existing ns at RO	Actions

Figure 4-68. Test 3 - Trial 3: NS status bug.

Once again, the same error was obtained, and no NS could be deployed even though both Network Services were up and running. This time, the issue was reported to OSM slack channel[66] and there, several developers reported similar problems which were answered equally with "Migrate to OSM Rel. SIX".

This trial is also considered to have FAILED as no NS could be deployed.

66 https://opensourcemano.slack.com/

5 Conclusions

From the very first moment this project started, two things were clear: it was going to be a great challenge due to the enormous quantity of new technologies, concepts and SW that were needed in order to achieve the final objectives and build the complete desired environment; and, besides, it was a great opportunity to come to know this new "world" that it is starting to arise around the 5G concept. The first week was, indeed, a real mess since the author had small-to-no knowledge about almost all the topics that were required in this project. However, although the learning-curve was steep at the beginning, the required final knowledge and the development and self-improving possibilities that this project provided, were more of a huge motivation than an impediment to achieve its goals.

This Master Thesis has tried to present a research about the current state of art of the 5G development status in Europe and a solution, with different scenarios and tests, which allows to build a 5G NSA E2E Sliced testbed with Open Source platforms. These goals were ambitious since 5G is a technology which is still being actively standardized and, therefore, the proper state of art could (and would) vary all along the development of the project. Furthermore, there were no previous Open Source 5G testbeds from which to have a base to design and implement the current 5G testbed. However, although almost all the fixed goals have been achieved, the final objective, building an E2E sliced environment, could not be fulfilled due to the problems encountered in the last test with the current support for NS in OSM Rel. FIVE. Nonetheless, even though this test is considered a failure, there is a clear path towards implementing the desired objective with OSM Rel. SIX with the developed VNFDs, NSDs and NSTs.

Despite what has been achieved and the excellent results obtained in the various experiments and scenarios, it is important to remark that the final test, the one that aimed at building an eMBB NS from Open Source platforms, failed as no real support was enabled in OSM Rel. FIVE for the desired scenario, more concretely, for hybrid/virtual NSs. Besides, one of the major conclusions that have been extracted from this Master Thesis is that, still nowadays, there is no clear difference (from a designing point of view) between building the OAI LTE environment as independent Network Services and building it as a unique, integrated NS because, currently, what the last option does is just adding a new abstraction layer above the already implemented functionality (Network Services are capable of connecting between them as they operate in the same network, in this case). On the other hand, the vast potential of the NS technology inside the 5G environment was also discovered, opening a whole new world of use cases, re-designing possibilities and major optimizing options in existing developments or implementations.

Additionally, even though the author had previous experience working in the "researching" world with other universities, working side-by-side with the 5G-VINNI European project and knowing that the results obtained in this project were going to be used as a main input inside the Spanish facility of this project, was both: a great source of motivation and a great opportunity to come to know how this big-scale projects work and how an edge-technology is developed and managed by different partners.

Finally, looking backwards, a tone of workload and effort was done in the path towards these words but, consequently, the objectives that seemed a challenge at the beginning have become a usable reality. A reality of which the author is very proud.

6 Future Work

Every single step and every single discovery that has been taken/found in this Master Thesis has opened doors towards new development possibilities inside the project. There are still lots of un-researched research topics inside the 5G environment, here are going to be mentioned those that are directly related to this project and will improve its performance and those to which the author feels more interest.

The first and obvious upgrade is migrating OSM Rel. FIVE to OSM Rel. SIX in the *Production Level Scenario 2* in order to verify the compatibility of this version of the SW with the NS concept. This migration could not be completed during the development of this project because while the latest tests were being carried out, one of the 5G-VINNI partners was testing the Spanish facility infrastructure, including the MANO platform and, thereupon, no changes could be done to the orchestrator machine. Furthermore, there is a clear path to follow in order to achieve a 5G SA implementation scenario which is, adding 5G NR and 5G NC. Introducing 5G NR in the current infrastructure could be done as 5TONIC has brought inside its laboratory an Ericsson NR massive MIMO antenna:

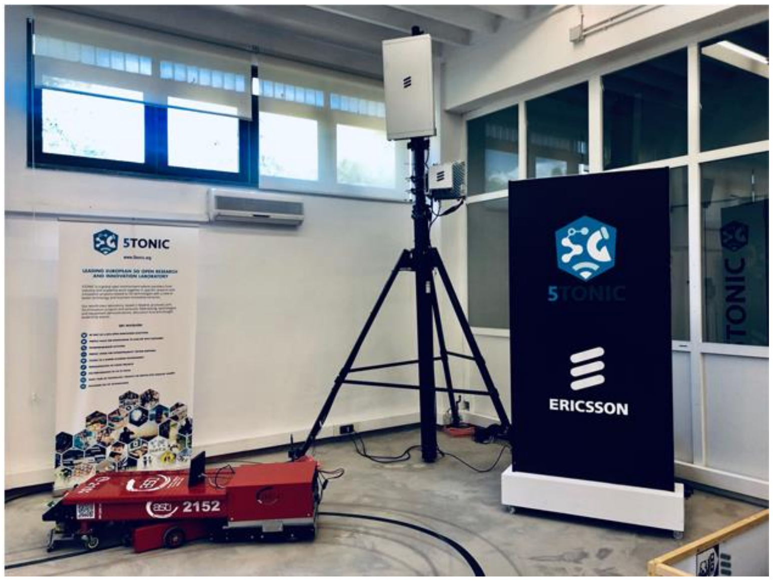


Figure 6-1. Ericsson 5G antenna.[67]

67 https://www.5tonic.org/news/ericsson-activates-5g-nsa-technology-5tonic-open-innovation-lab

The main problem was that this resource arrived one week before the end of this Master Thesis and, besides, it is a common resource shared between several of the European projects hosted in 5TONIC. Therefore, a vast research can be carried out around the 5G NR capabilities.

On the other hand, from an automation and optimizing point of view of the management processes involved in the MANO infrastructure operations various paths can be explored:

1. Generating more complex NSTs/NSDs/VNFDs using auto-scalable metadata fields that consider VNFs indicators (CPU, memory or disk usage, throughput, etc.).
2. Generating more complex NSTs/NSDs/VNFDs using auto-scalable metadata fields that consider VIM KPIs (OpenStack Ceilometer module data).
3. Verifying OSM Rel. SIX capabilities towards SW Image management at the VIM.
4. LCM automation using ML techniques and AI modules inside the 5G architecture components.

As it can be seen, a lot of future work can be done but, probably, the most promising one is researching the automation of the NS LCM using ML techniques.

7 References

[1] 5GPPP, Architecture Working Group, "5G-PPP.eu/white-papers," 15 December 2017. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2018/01/5G-PPP-5G-Architecture-White-Paper-Jan-2018-v2.0.pdf. [Accessed 28 July 2019].

[2] NGMN Alliance, "NGMN-Network Slicing," 14 September 2016. [Online]. Available: https://www.ngmn.org/fileadmin/user_upload/161010_NGMN_Network_Slicing_framework_v1.0.8.pdf. [Accessed 07 July 2019].

[3] 5G-VINNI, TID, "5G-VINNI Deliverable 3.1," March 2019. [Online]. Available: https://zenodo.org/record/3345612. [Accessed 07 July 2019].

[4] NGMN Alliance, "NGMN NWMO," 12 March 2019. [Online]. Available: https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2019/190312_5G_Network_and_Service_Management__including_Orchestration_3.14.0.pdf. [Accessed 28 July 2019].

[5] NGNM Alliance, "NGMN 5G White Paper," 17 February 2015. [Online]. Available: https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2015/NGMN_5G_White_Paper_V1_0.pdf. [Accessed 07 July 2019].

[6] NGNM Alliance, "5G E2E Architecture Framework," 04 October 2017. [Online]. Available: https://www.ngmn.org/fileadmin/user_upload/171006_NGMN_E2EArchFramework_v0.8.1_01.pdf. [Accessed 07 July 2019].

[7] NGNM Alliance, "Testing Framework for 5G pre-commercial trials," January 2018. [Online]. Available: https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2018/180220_NGMN_PreCommTrials_Framework_definition_v1_0.pdf. [Accessed 07 July 2019].

[8] NGNM Alliance, "NGNM 5G exposure Security Aspects," 21 September 2018. [Online]. Available: https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2018/180921_NGMN-NCEsec_white_paper_v1.0.pdf. [Accessed 07 July 2019].

[9] GSMA, "GSMA Operator Req.5G Core Connectivity," May 2019. [Online]. Available: https://www.gsma.com/futurenetworks/wp-content/uploads/2019/05/20190515-GSMA-Operator-Requirements-for-5G-Core-Connectivity-Options.pdf. [Accessed 08 July 2019].

[10] GSMA, "GSMA 5G Implementation Guidelines," July 2019. [Online]. Available: https://www.gsma.com/futurenetworks/wp-content/uploads/2019/03/5G-Implementation-Guideline-v2.0-July-2019.pdf. [Accessed 08 July 2019].

[11] GSMA, "GSMA An Introduction to NS," 2017. [Online]. Available: https://www.gsma.com/futurenetworks/wp-content/uploads/2017/11/GSMA-An-Introduction-to-Network-Slicing.pdf. [Accessed 08 July 2019].

[12] 3GPP SA WG2, "3GPP System Architecture for 5G systems (v.16.1.0)," June 2019. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.501/. [Accessed 08 July 2019].

[13] 3GPP SA WG2, "Procedures for the 5G system (v.15.4.0)," March 2018. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.502/. [Accessed 08 July 2019].

[14] 3GPP SA WG5, "MANO Concepts, requirements and use cases," December 2018. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/28_series/28.530/. [Accessed 08 July 2019].

[15] 5G PPP, "5G PPP Architecture WG," June 2019. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2019/07/5G-PPP-5G-Architecture-White-Paper_v3.0_PublicConsultation.pdf. [Accessed 30 July 2019].

[16] 5G PPP, "5G PPP Automotive WG," February 2019. [Online]. Available: https://bscw.5g-ppp.eu/pub/bscw.cgi/d293672/5G%20PPP%20Automotive%20WG_White%20Paper_Feb2019.pdf. [Accessed 30 July 2019].

[17] 5G PPP, "5G PPP Security WG," 2017. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP_White-Paper_Phase-1-Security-Landscape_June-2017.pdf. [Accessed 30 July 2019].

[18] ETSI ZSM, "ZSM Proof of concept Framework," May 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/006/01.01.01_60/gs_zsm006v010101p.pdf. [Accessed 08 July 2019].

[19] ETSI NFV, "Architectural Framework," December 2014. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf. [Accessed 08 July 2019].

[20] ETSI NFV-IFA, "Acceleration Technologies; VNF Interfaces Specification," March 2016. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/002/02.01.01_60/gs_NFV-IFA002v020101p.pdf. [Accessed 08 July 2019].

[21] ETSI MEC, "Technical Requirements," March 2016. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/01.01.01_60/gs_MEC002v010101p.pdf. [Accessed 29 July 2019].

[22] MEF, "MEF LSO," MEF, March 2019. [Online]. Available: http://www.mef.net/lso/lifecycle-service-orchestration. [Accessed 30 July 2019].

[23] 5G Transformer, "Report on vertical requirements and use cases," December 2017. [Online]. Available: http://5g-transformer.eu/wp-content/uploads/2017/12/Report_on_vertical_requirements_and_use_cases.pdf. [Accessed 30 July 2019].

[24] K. Ali, A. Alkhatar, N. Jawad and J. Cosmas, "IoRL Indoor Location Based Data Access, Indoor Location Monitoring & Guiding Interaction Applications," IEEE International Symposium on Broadband Multimedia Systems and, Valencia, 2018.

[25] IETF Network WG, "On the Difference between IMs and DMs," January 2013. [Online]. Available: https://tools.ietf.org/html/rfc3444. [Accessed 09 August 2019].

[26] 5G-VINNI, "D3.1 Specification of servicer delivered by each of the 5G-VINNI facilities," 28 June 2019. [Online]. Available: https://zenodo.org/record/3345612. [Accessed 09 August 2019].

[27] OASIS, "TOSCA Simple Profile for NFV," 11 May 2017. [Online]. Available: https://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.pdf. [Accessed 09 August 2019].

[28] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini and H. Flinck, "Network Slicing & Sowftwarization: A Survey," *IEE Communications Surveys and Tutorials,* vol. 20, no. 3, pp. 2429-2453, 2018.

[29] J. Ordonez Lucena, P. Ameigeiras, D. López, J. J. Ramos-Muñoz, J. Lorca and J. Folgueira, "Network Slicing for 5G with SDN/NFV: Concepts, Architectures and Challenges," *IEEE Communications Magazine,* vol. 55, no. 5, pp. 80-87, 2017.

[30] 5G-VINNI, "5G-VINNI Solution Facility-sites High Level Design," 31 March 2019. [Online]. Available: https://zenodo.org/record/2668791. [Accessed 09 August 219].

[31] A. Boubendir, F. Guillemin, S. Kerboeuf, B. Orlandi, F. Faucheux and J.-L. Lafragette, "Network Slice LifeCycle Management Towards Automation," 2019. [Online]. Available: http://dl.ifip.org/db/conf/im/im2019demo/191804.pdf. [Accessed 09 August 2019].

[32] 3GPP, "Technical Specification Group Services and System Aspects; Study on the Wireless and Wireline Convergence for the 5G system architecture Rel.16," 3GPP TR, 2018.

[33] L. Da Silva, A. Kaloxylos and G. Zimmermann, "On the impact of network slicing on 5G radio access networks," in *European conference on Networks and communications (EuCNC)*, Athens, Greece, 2016.

[34] P. S. Khodashenas, H. Khalili, D. Guija and M. S. Siddiqui, "TALENT: Towards Integration of Satellite and Terrestrial Networks," in *EUCNC*, Valencia, 2019.

8 Annex A: Planification and Budget

This chapter aims at informing the reader about the planification process of this project as well as the total costs of it. Chapter seven, consequently, is divided into two sections, one for each of the before-mentioned topics.

8.1 Planification

This section is dedicated to show the sketching process prior to the actual development of this project, being a key at the starting point as it gave the author an idea of the estimated time required for each stage of the ongoing Master Thesis. Five functional stages have been distinguished; each stage may be sub-divided in several sub-stages:

1. **Research&Analysis:** This stage is focused on researching the potential requirements of the different phases of the project, acquiring the necessary knowledge to start approaching the development and design of this Master Thesis (research, study and application of knowledge).
2. **Planification:** This stage aims at estimating the required amount of time for each of the next stages based on the knowledge acquired on the previous stage. It is an important phase as it will help to stablish chronological objectives and project milestones.
3. **Design and Development:** Phase that is dedicated to the execution of the project, achieving the different milestones is its main goal. It involves designing scenarios, updating the previous gathered information, developing tests and correcting issues.
4. **Testing:** This stage is mainly focused on testing the available infrastructure/scenarios and extracting conclusions, issues and potential upgrades from the obtained results.
5. **Documentation:** Final stage whose objective is to document and reflect, on a written document, the results, conclusions, development and evolution of the whole project.

Moreover, a comparison, using Gantt charts, between the estimated time and the actual time used for each of the different stages will be shown.

8.1.1 Estimated Time

Generally, estimating the time that a project is going to last may turn to be utterly difficult, especially if no previous experience over the field/topic is gotten, as, from the beginning, not all the information, requirements, risks and issues are known. For this reason, the actual real times have some slight delays in comparison with the estimated times (unforeseen errors, modifications, order issues, etc.). In the following Gantt Diagram, the estimated times for each of the previously mentioned stages are shown:

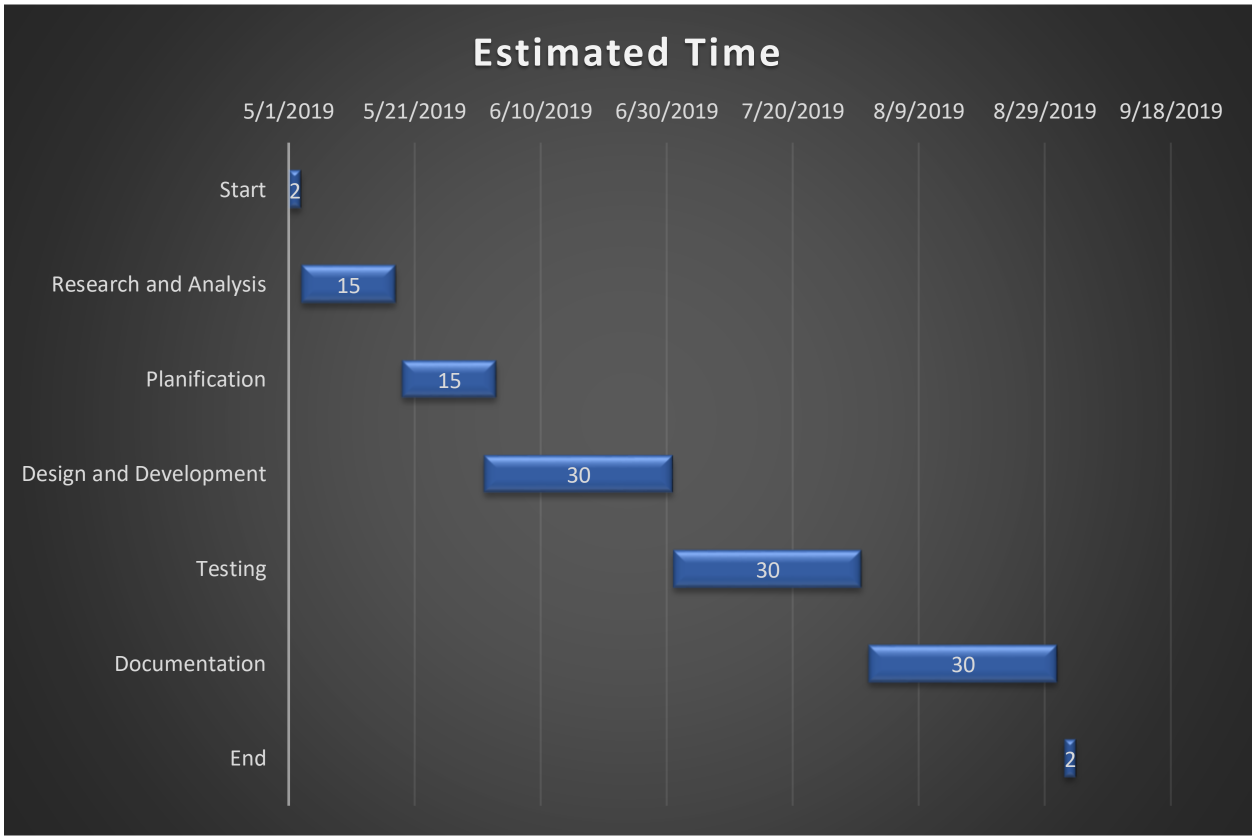


Figure 8-1. Estimated Time Gantt's Diagram

8.1.2 Actual time

Although some kind of parallelism was expected in the real time scenario as many of the tasks were going to require inputs from the others, it was not expected to have a complete need of developing almost all of the main tasks at same time because of the used SW new releases, SDOs new specifications and HW arrival problems. Thereupon, the "*Research&Analysis*" phase has been one of the most important ones in this project due to the highly-varying technological environment.

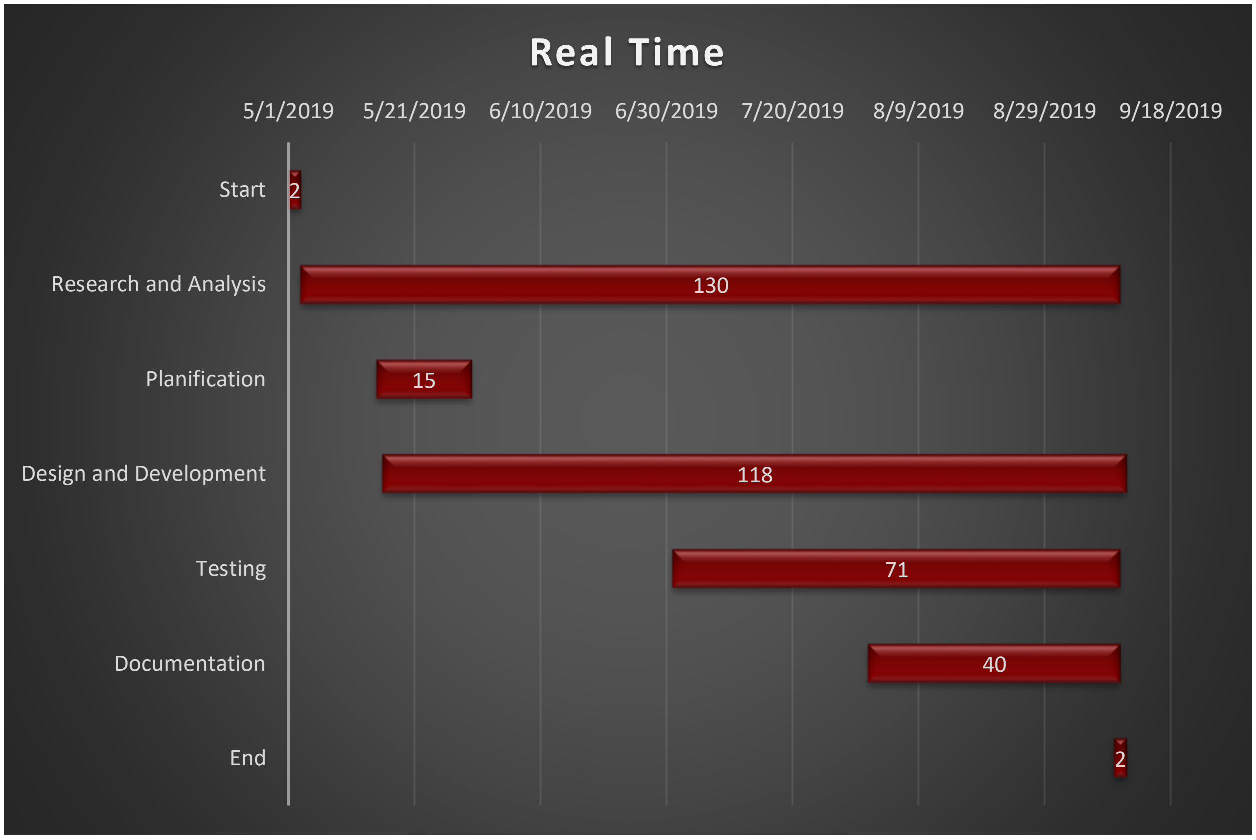


Figure 8-2. Real Time Gantt's Diagram.

8.2 Budget

This section exposes the execution estimated cost of this project. It includes prices that refer to semi-new[68] products and, in turn, includes the prices of the required licenses for those programs whose SW requires a paying license. It is important to remark that most of the HW shown in this table, except for the Generic Desktop and the Laptop, is not solely dedicated to this Master Thesis but it is shared infrastructure with the 5G-VINNI project.

Table 8-1. Estimated Budget.

Concept	Description	Quantity	Cost (Unit)	Subtotal
Engineer Salary	Software Engineer Price per month gross (22.5 weekly hours).	5	**929.89 €**	**5,579.34€**
Generic Desktop	PC +Desktop 24'' Price Semi-new Unit.	1	**242.00€**	**242.00€**
Laptop	MacBook Pro 13'' Price new Unit.	1	**2,542.94€**	**2,542.94€**
Portable Multiport Server	MiniPC ITX MC500G Price new Unit.	4	**2,843.50€**	**11,374.00€**
Remote Server		2	**1701.66€**	**3403.32€**
Datacenter Main Server	Datacenter rack server Price new Unit.	1	**7,263.61€**	**7,263.61€**
Datacenter Compute Node	Datacenter rack server Price new Unit.	1	**14,493.65€**	**14,493.65€**
USRP B200 mini-i	Digital Radio card Price new Unit.	2	**917.00€**	**1,834.00€**
Loop Back Cable Kit	Cable + attenuator kit Price new Unit.	2	**118.00€**	**236.00€**
Office 365	Microsoft Office Price Annual License.	1	**149.00€**	**149.00€**
Windows 10 Pro	OS Pre-installed SW.	1	**0.00€**	**0.00€**
Ubuntu 16.04	OS Installed SW.	1	**0.00€**	**0.00€**
Ubuntu 18.04	OS Installed SW.	1	**0.00€**	**0.00€**
OpenStack	Cloud-computing SW platform Apache 2.0 License.	1	**0.00€**	**0.00€**
OSM	Management and Orchestration SW Apache 2.0 License.	1	**0.00€**	**0.00€**

68 All products that are, at least, one year old since purchase, and whose amortization has not expired.

OAI	Open LTE/5G radio platform OAI public 1.1 License.	**1**	**0.00€**	**0.00€**
Juju Charms	Open Source modelling tool GNU-Affero GPL-v3 LGPL License.	**1**	**0.00€**	**0.00€**
Ansible	Open Source automating SW provisioning platform Proprietary / GPLv2 License.	**1**	**0.00€**	**0.00€**
KVM	Virtualization Infrastructure manager LGPL License.	**1**	**0.00€**	**0.00€**
VirtualBox	Hypervisor GPLv2 License	**1**	**0.00€**	**0.00€**

Disc.: (0.31%)*	**149.00€**
Tax Base	**38,817.24€**
IVA (21%)	**8,151.62€**
Total (*)	**46,968.86€**

NOTE: The indicated discount has been applied because the license for Office 365 was obtained through the UAH's student program, therefore, its value was paid by this entity.

The main economic source of this Master Thesis has been the European 5GPPP 5G-VINNI project.

9 Annex B: Specifications

9.1 General Specification

This Specification section is the characteristics' summary that must be met in the installation, execution and analysis of the project described in this Master Thesis, as well as the devices used in the development and design of it.

For any specification outside those contemplated here, the author of this Master Thesis takes no responsibility if the resulting environment does not behave as described and expected.

For a detailed description on how to install, meet the necessary software requirements, execute and test the different scenarios, see "*section 0.*

Technical Development".

9.2 Technical Specifications

In this subsection the technical specifications (SW and HW) required to reproduce exactly the project as the author, are exposed. Besides, this subsection is split in two sections, one for each technical specification set.

9.2.1 Hardware

Due to the high computational consumption that is required to develop this project and due to the wide variety of explored and proposed scenarios, a high number of physical devices is required. The functionality, purpose and main characteristics of each of the devices is exposed in each Scenario section on this document; except for the "Generic HP Workstation" which is a personal computer used, basically, to write this Master Thesis and the Laptop which is the organizations' computer. Bellow, detailed characteristics are exposed:

Device	Information
Generic HP Workstation	***OS:*** *Windows 10 Pro.* ***CPU:*** *Intel® CoreTM i7-2600 (3.40GHz).* ***GPU:*** *NVIDIA GForce GT 530 (2GB RAM).* ***RAM:*** *2x 4GB DDR3 (1600MHz).* ***DD:*** *Seagate Barracuda HDD (1TB-SATA3).* ***Desktop 1:*** *Samsung HD 22'' (1920x1080).* ***Desktop 2:*** *HP LP2465 HD 24'' (1920x1200).*
MacBook Pro 13''	***OS:*** *macOS Mojave (v. 10.14.6)* ***CPU:*** *Intel® CoreTM i7-2720QM (2.20GHz – 4 cores).* ***GPU:*** *Intel Iris Plus Graphics 640 (1536MB VRAM).* ***RAM:*** *2x 8GB LPDDR3 (2133MHz).* ***DD:*** *APPLE SSD AP1024J (1TB-SSD).* ***Pantalla 1:*** *Built-In Retina LCD 13'' (2560x1600).*
Portable Multiport Server (x4)	***Model:*** *MiniPC ITX MC500G* ***OS:*** *Ubuntu 16.04| Ubuntu 18.04* ***CPU:*** *Intel® CoreTM i5-3210ME (2.7GHz).* ***GPU:*** *IntelHD Graphics 630 (3GB RAM).* ***RAM: 1****x 8GB DDR3 (1600MHz).* ***DD:*** *SSD (128GB-SATA3).*
Remote Server (x2)	***SO:*** *Ubuntu 18.04.2 LTS-Bionic* ***CPU:*** *Intel® CoreTM i7-8700k (4.7GHz).* ***GPU:*** *NVIDIA Quadro P620 (2GB RAM).* ***RAM:*** *2x 8GB DDR4 (2666MHz).* ***DD-1:*** *Z Turbo SSD (512GB).* ***DD-2:*** *HDD (2TB - SATA3).* ***Network Card:*** *Intel X710-DA2 (10GbE -dualPort)*
DataCenter Remote Main Server	***OS:*** *Ubuntu 18.04.2 LTS-Bionic* ***CPU:*** *Intel® Xeon® Silver 4114 (2.20GHz, 40 cores).* ***IDRAC:*** *v.9 Enterprise* ***RAM:*** *2x 32GB DDR4 (2666MHz).* ***DD:*** *x4 3TB HDD SAS (7200rpm)* ***Cache:*** *Non-volatile (2GB)* ***RAID PERC:*** *H730+ driver* ***Power Source:*** *500W, 1+1* ***Network Card 1:*** *Broadcom 5720 (1Gb)* ***Network Card 2:*** *Intel Eth.i350 (1Gb, 4 ports)* ***Network Card 3:*** *Integrated LOM (1Gb, 2 ports)*

Datacenter Remote Compute Node	***OS:*** *Ubuntu 18.04.2 LTS-Bionic* ***CPU:*** *Intel® Xeon® Gold 6152 (2.10GHz, 88 cores).* ***IDRAC:*** *v.9 Enterprise* ***RAM:*** *2x 32GB DDR4 (2666MHz).* ***DD:*** *x4 2TB SDD SAS (7200rpm)* ***Cache:*** *Non-volatile (2GB)* ***RAID PERC:*** *H730+ driver* ***Power Source:*** *550W, 1+1* ***Network Card 1:*** *Broadcom 5720 (1Gb)* ***Network Card 2:*** *Intel Eth.i350 (1Gb, 4 ports)* ***Network Card 3:*** *Integrated LOM (1Gb, 2 ports)*
USRP B200 mini-i (x2)	***Frequency Range:*** *70MHz-6GHz* ***Bandwidth:*** *Up to 56MHz* ***FPGA:*** Xilinx Spartan-6 XC6SLX75 ***GNU Radio Support***
Loop Back Cable kit (x2)	***SMA cable and 30dB Attenuator***

Table 9-1. Project's HW requirements.

9.2.2 Software

Most of the SW used in this project has been selected from scratch in order to obtain the best and most efficient overall performance results, as well as to achieve the ultimate goal of the project using Open Source SW. Moreover, as the experiments are run over virtualized infrastructure and specific programs, those has to be maintained if the same behavior is wanted, but the rest of the SW, including the Operating Systems (OSs) of the main server and personal computer, and text processors MAY be changed by the person who tries to reproduce or extend this project if other personal preferences are present. Bellow, a table with a schematic of the SW is shown:

Table 9-2. Project's SW requirements.

Software	Information
Windows 10 Pro	OS installed on the *Workstation* personal computer. Used mainly for writing and documentation purposes.
Ubuntu 16.04	OS installed on the OSM VM/computers and used in several of the trial and scenarios.
Ubuntu 18.04	OS installed on the DataCenter Servers, the eNB/UE servers and on several of the scenario VMs.
OpenStack Stein	See section 2.6.1.1 *OpenStack.*
OSM Rel. FIVE	See section 2.6.2.3 *Open Source MANO.*

Juju Charms	Open Source application and service modelling tool that helps to deploy, manage and scale cloud applications. It is used for Day-1 configurations on OSM.
Ansible	Open Source SW that automates SW provisioning, configuration management and application deployment. Used as a bonus tool to simplify Day-1 configurations processes on OSM.
OAI	See section 2.6.3 *Radio.*
KVM	Full Open Source virtualization solution for Linux on x86 HW. It is used to generate several VMs on the *DataCenter Remote Main Server* and to generate virtualized networks on the first production-scenario.
VirtualBox	Open source hypervisor for x86 and AMD64/Intel64 computers for enterprise and home use. It is used as the virtualization tool in the first scenario (mock Scenario).
Office 365	Microsoft Office SW set that enables Access to two of the applications used on this Master Thesis for documentation purposes: Word, PowerPoint and OneDrive.

10 Annex C: Configuration Files

10.1 Initial Scenario DevStack "local.conf" file

```
# The ``localrc`` section replaces the old ``localrc`` configuration file.
# Note that if ``localrc`` is present it will be used in favor of this section.
[[local|localrc]]
ADMIN_PASSWORD=admin1234
DATABASE_PASSWORD=admindb
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
HOST_IP=192.168.56.108
#HOST_IPV6=2001:db8::7
FLOATING_RANGE=10.4.0.0/27
FIXED_RANGE=10.0.3.0/24

# Image Downloads
IMAGE_URLS="http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img"
IMAGE_URLS+="http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img"
IMAGE_URLS+="https://cloud-images.ubuntu.com/xenial/current/xenial-server-cloudimg-
amd64-disk1.img"

# Logging
LOGFILE=$DEST/logs/stack.sh.log
LOGDAYS=2

# Swift
SWIFT_HASH=66a3d6b56c1f479c8b4e70ab5c2000f5
SWIFT_REPLICAS=1
SWIFT_DATA_DIR=$DEST/data
```

10.2 Physical server Netplan config scenario 3

```
network:
    ethernets:
        eno1:
            dhcp4: false
            dhcp6: false
        eno2:
            dhcp4: true
        enp175s0f0:
            dhcp4: true
        enp175s0f1:
            dhcp4: true
        enp59s0f0:
            dhcp4: true
        enp59s0f1:
            dhcp4: true
        enp59s0f2:
            dhcp4: true
        enp59s0f3:
            dhcp4: true
    version: 2

    bridges:
          br0:
            interfaces: [eno1]
            addresses: [10.5.10.10/16]
            gateway4: 10.5.0.1
            mtu: 1500
            nameservers:
                addresses: [8.8.8.8]
            dhcp4: no
            dhcp6: no
```

10.3 Physical server libvirt bridge br0 definition

```
<network>
  <name>host-bridge</name>
  <uuid>17e2484b-39c1-440f-94ca-706bc786ca54</uuid>
  <forward mode='bridge'/>
  <bridge name='br0'/>
</network>
```

10.4 Physical server libvirt NAT network definition

```
<network>
  <name>NAT_mgmt</name>
  <uuid>0935f331-7570-4ff8-a7b2-ee37ffe33774</uuid>
  <forward mode='nat'/>
  <bridge name='virbr_vm' stp='on' delay='0'/>
  <mac address='52:54:00:4a:cc:eb'/>
  <ip address='10.0.0.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='10.0.0.2' end='10.0.0.254'/>
    </dhcp>
  </ip>
</network>
```

10.5 Scenario 2 OSM machine /etc/network/interfaces file

```
source /etc/network/interfaces.d/*

auto lo
iface lo inet loopback

auto eno1
iface eno1 inet static
      address 163.117.140.197
      netmask 255.255.255.0
      network 163.117.140.0
      gateway 163.117.140.2
      dns-nameservers 8.8.8.8

auto enp6s0
iface enp6s0 inet static
      address 10.10.10.1
      netmask 255.255.255.0
      network 10.10.10.0
      dns-nameservers 8.8.8.8

auto enp2s0
iface enp2s0 inet static
      address 192.168.200.1
      netmask 255.255.255.0
      network 192.168.200.0
      dns-nameservers 8.8.8.8
```

10.6 Scenario 2 *local.conf* DevStack File

```
[[local|localrc]]

ADMIN_PASSWORD=admin1234

DATABASE_PASSWORD=admindb

RABBIT_PASSWORD=$ADMIN_PASSWORD

SERVICE_PASSWORD=$ADMIN_PASSWORD

HOST_IP =10.10.10.2

SERVICE_HOST=$HOST_IP

PUBLIC_INTERFACE=enp2s0

FLOATING_RANGE =192.168.200.0/24

Q_FLOATING_ALLOCATION_POOL=start=192.168.200.150 ,end=192.168.200.250


# Image Downloads

IMAGE_URLS="http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-
disk.img"

IMAGE_URLS+=”https://cloud-images.ubuntu.com/xenial/current/xenial-server-
cloudimg-amd64-disk1.img”


# Logging

LOGFILE=$DEST/logs/stack.sh.log

LOGDAYS=2


# Swift

SWIFT_HASH=66a3d6b56c1f479c8b4e70ab5c2000f5

SWIFT_REPLICAS=1

SWIFT_DATA_DIR=$DEST/data
```

11 Annex D: VNFDs

11.1 Cirros VNFD - Test 1

```
vnfd:vnfd-catalog:

    vnfd:

    -   id: cirros_vnfd

        name: cirros_vnf

        short-name: cirros_vnf

        description: Cirros Basic VNF example

        vendor: AGS

        version: '1.1'


        # Place the logo as png in icons directory and provide the name here

        logo: cirros-64.png


        # Management interface

        mgmt-interface:

            cp: eth0


        # At least one VDU needs to be specified

        vdu:

        -   id: cirros_vnfd-VM

            name: cirros_vnfd-VM

            description: cirros_vnfd-VM

            count: 1


            # Flavour of the VM to be instantiated for the VDU

            vm-flavor:

                vcpu-count: 1

                memory-mb: 256

                storage-gb: 2


            # Image/checksum or image including the full path

            image: cirros034
    #checksum:

    interface:
    # Specify the external interfaces
    # There can be multiple interfaces defined
    -   name: eth0
        type: EXTERNAL
        virtual-interface:
            type: VIRTIO
        external-connection-point-ref: eth0

connection-point:
    -   name: eth0
        type: VPORT
```

11.2 Cloud-init sshkey VNFD - Test 2

```
vnfd:vnfd-catalog:

    vnfd:

    -   id: cloud_init_sshkey_vnfd

        name: cloud_init_sshkey_vnfd

        short-name: cloud_init_sshkey_vnfd

        description: Ubuntu Cloud-init test with SSH key

        vendor: AGS

        version: '1.0'

        logo: uc3m.jpg


        # Management interface

        mgmt-interface:

            cp: mgmt


        vdu:

        -   cloud-init-file: c_init_sshkey_vnfd.cfg

            id: cloud_init_sshkey_vnfd-VM

            name: cloud_init_sshkey_vnfd-VM

            description: cloud_init_sshkey_vnfd-VM

            count: 1

            vm-flavor:

                vcpu-count: 1

                memory-mb: 2048

                storage-gb: 20


            image: 'ubuntu-xenial-amd64-2nic'


            interface:

            -   name: ens3

                type: EXTERNAL

                virtual-interface:

                    type: VIRTIO
        external-connection-point-ref: mgmt
    -   name: ens4
        type: EXTERNAL
        virtual-interface:
            type: VIRTIO
        external-connection-point-ref: data

connection-point:
-   name: mgmt
    type: VPORT
-   name: data
    type: VPORT
```

11.3 Cloud-init sshpass VNFD - Test 2

```
vnfd:vnfd-catalog:
    vnfd:
    -   id: cloud_init_sshpass_vnfd
        name: cloud_init_sshpass_vnfd
        short-name: cloud_init_sshpass_vnfd
        description: VNFD with cloud-init password credentials (no ssh key)
        vendor: AGS
        version: '1.0'
        logo: uc3m.jpg

        mgmt-interface:
            cp: mgmt

        vdu:
        -   cloud-init-file: cloud_init_sshpass.cfg
            id: cloud_init_sshpass_vnfd-VM
            name: cloud_init_sshpass_vnfd-VM
            description: cloud_init_sshpass_vnfd-VM
            count: 1
            vm-flavor:
                vcpu-count: 1
                memory-mb: 2048
                storage-gb: 20

            image: 'ubuntu-xenial-amd64-2nic'

            interface:
            -   name: ens3
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
                external-connection-point-ref: mgmt
    -   name: ens4
        type: EXTERNAL
        virtual-interface:
            type: VIRTIO
        external-connection-point-ref: data

connection-point:
-   name: mgmt
    type: VPORT
-   name: data
    type: VPORT
```

11.4 Juju Basic VNFD - Test 3

```
vnfd:vnfd-catalog:

    vnfd:

    -   id: juju_basic_vnfd

        name: juju_basic_vnfd

        short-name: juju_basic_vnfd

        description: Basic VNFD mod testing

        vendor: UC3M

        version: '1.1'

        logo: 'uc3m.jpg'


        mgmt-interface:

            cp: mgmt


        vdu:

        -   cloud-init-file: cloud-init.cfg

            id: juju_basic_vnfd-VM

            name: juju_basic_vnfd-VM

            description: juju_basic_vnfd-VM

            count: 1

            image: ubuntu-bionic-3nic


            vm-flavor:

                vcpu-count: 1

                memory-mb: 2048

                storage-gb: 20


            interface:

            -   name: ens3

                type: EXTERNAL

                virtual-interface:

                    type: VIRTIO

                external-connection-point-ref: mgmt
```

```
connection-point:
-   name: mgmt
    type: VPORT

vnf-configuration:
    juju:
        charm: simple
    initial-config-primitive:
    -   seq: '1'
        name: config
        parameter:
        -   name: ssh-hostname
            value: <rw_mgmt_ip>
        -   name: ssh-username
            value: sysadm
        -   name: ssh-password
            value: 5gvinni2019
    -   seq: '2'
        name: touch
        parameter:
        -   name: filename
            value: '/home/sysadm/test-file'

    config-primitive:
    -  name: touch
       parameter:
       -   name: filename
           data-type: STRING
           default-value: '/home/sysadm/touched'
```

11.5 Ansible Basic VNFD - Test 3

```
vnfd:vnfd-catalog:
    vnfd:
    -   id: ansible_basic_vnfd
        name: ansible_basic_vnfd
        short-name: ansible_basic_vnfd
        description: Basic VNFD mod testing
        vendor: UC3M
        version: '1.1'
        logo: 'uc3m.jpg'

        mgmt-interface:
            cp: mgmt

        vdu:
        -   cloud-init-file: cloud-init.cfg
            id: ansible_basic_vnfd-VM
            name: ansible_basic_vnfd-VM
            description: ansible_basic_vnfd-VM
            count: 1
            image: ubuntu-bionic-3nic

            vm-flavor:
                vcpu-count: 2
                memory-mb: 2048
                storage-gb: 20

            interface:
            -   name: ens3
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
                external-connection-point-ref: mgmt
```

```
connection-point:
-   name: mgmt
    type: VPORT

vnf-configuration:
    juju:
        charm: ansible-charm-v2
    initial-config-primitive:
    -   seq: '1'
        name: config
        parameter:
        -   name: ssh-hostname
            value: <rw_mgmt_ip>
        -   name: ssh-username
            value: sysadm
        -   name: ssh-password
            value: 5gvinni2019
```

11.6 HSS VNFD - Test 4

```
vnfd:vnfd-catalog:
  vnfd:
  - connection-point:
    - name: mgmt
      port-security-enabled: false
      type: VPORT
    - name: data
      port-security-enabled: false
      type: VPORT
    description: HSS-EPC VNFD descriptor
    id: hss_epc_vnfd
    mgmt-interface:
      cp: mgmt
    name: hss_epc_vnfd
    short-name: hss_epc_vnfd
    vdu:
    - cloud-init-file: c_init_sshkey_vnfd.cfg
      count: 1
      description: hss_epc_vnfd-VM
      id: hss_epc_vnfd-VM
      image: ubuntu-bionic-3nic #ubuntu-xenial-amd64-2nic
      interface:
      - external-connection-point-ref: mgmt
        name: ens3
        type: EXTERNAL
        virtual-interface:
          type: VIRTIO
      - external-connection-point-ref: data
        name: ens4
        type: EXTERNAL
        virtual-interface:
          type: VIRTIO
    name: hss_epc_vnfd-VM
    vm-flavor:
      memory-mb: 4096
      storage-gb: 40
      vcpu-count: 2
  vendor: UC3M
  version: '1.1'
  vnf-configuration:
          initial-config-primitive:
          -   name: config
              parameter:
              -   name: ssh-hostname
                  value: <rw_mgmt_ip>
              -   name: ssh-username
                  value: sysadm
              -   name: ssh-password
                  value: 5gvinni2019
              seq: '1'
          juju:
              charm: ansible-charm-v2
```

11.7 MME VNFD - Test 4

```
vnfd:vnfd-catalog:
  vnfd:
  - connection-point:
    - name: mgmt
      port-security-enabled: false
      type: VPORT
    - name: data1
      port-security-enabled: false
      type: VPORT
    - name: data2
      port-security-enabled: false
      type: VPORT
    description: MME-EPC VNF Descriptor
    id: mme_epc_vnfd
    mgmt-interface:
      cp: mgmt
    name: mme_epc_vnfd
    short-name: mme_epc_vnfd
    vdu:
    - cloud-init-file: c_init_sshkey_vnfd.cfg
      count: 1
      description: mme_epc_vnfd-VM
      id: mme_epc_vnfd-VM
      image: ubuntu-bionic-3nic  #ubuntu-xenial-amd64-3nic
      interface:
      - external-connection-point-ref: mgmt
        name: ens3
        type: EXTERNAL
        virtual-interface:
          type: VIRTIO
      - external-connection-point-ref: data1
        name: ens4
      type: EXTERNAL
      virtual-interface:
        type: VIRTIO
    - external-connection-point-ref: data2
      name: ens5
      type: EXTERNAL
      virtual-interface:
        type: VIRTIO
    name: mme_epc_vnfd-VM
    vm-flavor:
      memory-mb: 4096
      storage-gb: 40
      vcpu-count: 2
  vendor: UC3M
  version: '1.1'
  vnf-configuration:
          initial-config-primitive:
          -   name: config
              parameter:
              -   name: ssh-hostname
                  value: <rw_mgmt_ip>
              -   name: ssh-username
                  value: sysadm
              -   name: ssh-password
                  value: 5gvinni2019
              seq: '1'
          juju:
              charm: ansible-charm-v2
```

11.8 SPGW VNFD - Test 4

```
vnfd:vnfd-catalog:
  vnfd:
  - connection-point:
    - name: mgmt
      port-security-enabled: false
      type: VPORT
    - name: data
      port-security-enabled: false
      type: VPORT
    description: SPGW-EPC LTE VNF Descriptor
    id: spgw_epc_vnfd
    mgmt-interface:
      cp: mgmt
    name: spgw_epc_vnfd
    short-name: spgw_epc_vnfd
    vdu:
    - cloud-init-file: c_init_sshkey_vnfd.cfg
      count: 1
      description: spgw_epc_vnfd-VM
      id: spgw_epc_vnfd-VM
      image: ubuntu-bionic-3nic  #ubuntu-xenial-amd64-2nic
      interface:
      - external-connection-point-ref: mgmt
        name: ens3
        type: EXTERNAL
        virtual-interface:
          type: VIRTIO
      - external-connection-point-ref: data
        name: ens4
        type: EXTERNAL
        virtual-interface:
          type: VIRTIO
  name: spgw_epc_vnfd-VM
  vm-flavor:
    memory-mb: 4096
    storage-gb: 40
    vcpu-count: 4
vendor: UC3M
version: '1.1'
vnf-configuration:
        initial-config-primitive:
        -   name: config
            parameter:
            -   name: ssh-hostname
                value: <rw_mgmt_ip>
            -   name: ssh-username
                value: sysadm
            -   name: ssh-password
                value: 5gvinni2019
            seq: '1'
        juju:
            charm: ansible-charm-v2
```

11.9 eNB PNFD - Test 5

```
vnfd:vnfd-catalog:
    vnfd:
    -   id: eNB_pnfd
        name: eNB_pnfd
        short-name: eNB_pnfd
        description: eNB PNF
        vendor: UC3M
        version: '1.0'
        mgmt-interface:
            cp: mgmt
        vdu:
        -   id: eNB_pdu
            description: eNB PDU
            interface:
            -   name: eno1
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
                external-connection-point-ref: mgmt
            pdu-type: eNB
        connection-point:
        -   name: mgmt
            type: VPORT
        vnf-configuration:
            initial-config-primitive:
            -   name: config
                parameter:
                -   name: ssh-hostname
                    value: <rw_mgmt_ip>
                -   name: ssh-username
                    value: sysadm
                -   name: ssh-password
            value: 5gvinni2019
        seq: '1'
    juju:
        charm: ansible-charm-v2
```

12 Annex E: NSDs

12.1 Test-1 NSD

```
nsd:nsd-catalog:

    nsd:

    -   id: cirros_2vnf_nsd

        name: cirros_2vnf_ns

        short-name: cirros_2vnf_ns

        description: Cirros Example Network Service

        vendor: AGS

        version: '1.1'


        # Place the logo as png in icons directory and provide the name here

        logo: osm_2x.png


        # Specify the VNFDs that are part of this NSD

        constituent-vnfd:

        -   member-vnf-index: 1

            vnfd-id-ref: cirros_vnfd

        -   member-vnf-index: 2

            vnfd-id-ref: cirros_vnfd


        vld:

        # Networks for the VNFs

            -   id: vld1

                name: cirros_2vnf_nsd_vld1

                short-name: vld1

                type: ELAN

                mgmt-network: 'true'

                vnfd-connection-point-ref:

                -   member-vnf-index-ref: 1

                    vnfd-id-ref: cirros_vnfd

                    vnfd-connection-point-ref: eth0

                -   member-vnf-index-ref: 2
```

```
vnfd-id-ref: cirros_vnfd

vnfd-connection-point-ref: eth0
```

12.2 Test 2 NSD (Cloud-Init)

```
nsd:nsd-catalog:
    nsd:
    -   id: cloud_init-trial_nsd
        name: cloud_init-trial_nsd
        short-name: cloud_init-trial_nsd
        description: CLoud init login network service test
        vendor: AGS
        version: '1.0'
        logo: uc3m.jpg

        constituent-vnfd:
        -   member-vnf-index: 1
            vnfd-id-ref: cloud_init_sshpass_vnfd
        -   member-vnf-index: 2
            vnfd-id-ref: cloud_init_sshkey_vnfd

        vld:
        # Networks for the VNFs
        -   id: mgmt_net
            name: management
            short-name: management
            type: ELAN
            mgmt-network: 'true'
            vim-network-name: public
            vnfd-connection-point-ref:
            -   member-vnf-index-ref: 1
                vnfd-id-ref: cloud_init_sshpass_vnfd
                vnfd-connection-point-ref: mgmt
            -   member-vnf-index-ref: 2
                vnfd-id-ref: cloud_init_sshkey_vnfd
                vnfd-connection-point-ref: mgmt
```

```
-   id: datanet
    name: data
    short-name: data
    type: ELAN
    vnfd-connection-point-ref:
    -   member-vnf-index-ref: 1
        vnfd-id-ref: cloud_init_sshpass_vnfd
        vnfd-connection-point-ref: data
    -   member-vnf-index-ref: 2
        vnfd-id-ref: cloud_init_sshkey_vnfd
        vnfd-connection-point-ref: data
```

12.3 Juju-basic NSD

```
nsd:nsd-catalog:

    nsd:

    -   id: juju_basic_nsd

        name: juju_basic_nsd

        short-name: juju_basic_nsd

        description: Basic NS with day-1 vnf config test

        vendor: UC3M

        version: '1.1'

        logo: 'uc3m.jpg'


        constituent-vnfd:

        -   member-vnf-index: 1

            vnfd-id-ref: juju_basic_vnfd


        vld:

        -   id: mgmt_net

            name: management

            short-name: management

            type: ELAN

            mgmt-network: 'true'

            vim-network-name: public

            vnfd-connection-point-ref:

            -   ip-address: 10.5.10.199

                member-vnf-index-ref: 1

                vnfd-id-ref: juju_basic_vnfd

                vnfd-connection-point-ref: mgmt
```

12.4 Ansible Basic NSD

```
nsd:nsd-catalog:
    nsd:
    -   id: ansible_basic_nsd
        name: ansible_basic_nsd
        short-name: ansible_basic_nsd
        description: Basic NS with day-1 vnf config test
        vendor: UC3M
        version: '1.1'
        logo: 'uc3m.jpg'

        constituent-vnfd:
        -   member-vnf-index: 1
            vnfd-id-ref: ansible_basic_vnfd

        vld:
        -   id: mgmt_net
            name: management
            short-name: management
            type: ELAN
            mgmt-network: 'true'
            vim-network-name: public
            vnfd-connection-point-ref:
            -   ip-address: 10.5.10.198
                member-vnf-index-ref: 1
                vnfd-id-ref: ansible_basic_vnfd
                vnfd-connection-point-ref: mgmt
```

12.5 EPC NSD - Test 4

```
nsd:nsd-catalog:
  nsd:
  - constituent-vnfd:
    - member-vnf-index: 1
      vnfd-id-ref: spgw_epc_vnfd
    - member-vnf-index: 2
      vnfd-id-ref: mme_epc_vnfd
    - member-vnf-index: 3
      vnfd-id-ref: hss_epc_vnfd
    description: EPC LTE NS descriptor
    id: epc_ns
    ip-profiles:
    - ip-profile-params:
        gateway-address: null
        ip-version: ipv4
        subnet-address: 10.0.1.0/29
      name: S6-a
    - ip-profile-params:
        gateway-address: null
        ip-version: ipv4
        subnet-address: 10.0.2.0/29
      name: S11
    name: epc_ns
    short-name: epc_ns
    vendor: UC3M
    version: '1.2'
    vld:
    - id: management
      mgmt-network: 'true'
      name: mgmt_net
      type: ELAN
      vim-network-name: public
      vnfd-connection-point-ref:
      - ip-address: 10.5.10.14
        member-vnf-index-ref: '1'
        vnfd-connection-point-ref: mgmt
        vnfd-id-ref: spgw_epc_vnfd
      - ip-address: 10.5.10.13
        member-vnf-index-ref: '2'
        vnfd-connection-point-ref: mgmt
        vnfd-id-ref: mme_epc_vnfd
      - ip-address: 10.5.10.12
        member-vnf-index-ref: '3'
        vnfd-connection-point-ref: mgmt
        vnfd-id-ref: hss_epc_vnfd
    - id: S6-a_net
      ip-profile-ref: S6-a
      name: S6-a_net
      type: ELAN
      vnfd-connection-point-ref:
      - ip-address: 10.0.1.2
        member-vnf-index-ref: '2'
        vnfd-connection-point-ref: data1
        vnfd-id-ref: mme_epc_vnfd
      - ip-address: 10.0.1.3
        member-vnf-index-ref: '3'
        vnfd-connection-point-ref: data
        vnfd-id-ref: hss_epc_vnfd
    - id: S11_net
      ip-profile-ref: S11
      name: S11_net
      type: ELAN
      vnfd-connection-point-ref:
      - ip-address: 10.0.2.2
        member-vnf-index-ref: '2'
        vnfd-connection-point-ref: data2
        vnfd-id-ref: mme_epc_vnfd
      - ip-address: 10.0.2.3
        member-vnf-index-ref: '1'
        vnfd-connection-point-ref: data
        vnfd-id-ref: spgw_epc_vnfd
```

12.6 eNB NSD - Test 5

```
nsd:nsd-catalog:

    nsd:

    -   id: eNB_pnf_nsd

        name: eNB_pnf_nsd

        short-name: eNB_pnf_nsd

        description: NSD for eNB PNF

        vendor: UC3M

        version: '1.1'

        constituent-vnfd:

        -   member-vnf-index: 1

            vnfd-id-ref: eNB_pnfd


        connection-point:

        -   name: eNB_cp_mgmt

            vld-id-ref: eNB_pnf_nsd_mgmt


        vld:

        -   id: eNB_pnf_nsd_mgmt

            name: management

            short-name: management

            type: ELAN

            mgmt-network: 'true'

            vim-network-name: public

            vnfd-connection-point-ref:

            -   member-vnf-index-ref: 1

                vnfd-id-ref: eNB_pnfd

                vnfd-connection-point-ref: mgmt
```

13 Annex F: NSTs

13.1 5G NSA OAI NST - Test 5

```
# Copyright [2019] [Adrián Gallego]

#

# Licensed under the Apache License, Version 2.0 (the "License");

# you may not use this file except in compliance with the License.

# You may obtain a copy of the License at

#

#     http://www.apache.org/licenses/LICENSE-2.0

#

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS,

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

# See the License for the specific language governing permissions and

# limitations under the License.


#NST to deploy a Full 5G NSA OAI hybrid architecture composed by two Network Services:
eNB and EPC

nst:

-   id: 5g_na_nst

    name: 5g_na_nst

    SNSSAI-identifier:

        slice-service-type: eMBB

    quality-of-service:

        id: 1


    netslice-subnet:

    -   id: eNB_ns

        is-shared-nss: 'false'

        description: NetSlice Subnet (service) composed by 1 pnf (eNB) and 1 cp (mgmt)

        nsd-ref: eNB_pnf_nsd

    -   id: vEPC_ns

        is-shared-nss: 'false'
        description: NetSlice Subnet (service) composed by 3 vnfs and 8 cp (3 mgmt and
5 data)
        nsd-ref: epc_ns


    netslice-vld:
    -   id: vld_mgmt
        name: vld_mgmt
        type: ELAN
        mgmt-network: 'true'
        nss-connection-point-ref:
        -   nss-ref: eNB
            nsd-connection-point-ref: eNB_cp_mgmt
        -   nss-ref: vEPC
            nsd-connection-point-ref: epc_cp_mgmt
```

14 Annex G: Ansible Playbooks

14.1 Ansible Basic - Test 3

```
- hosts: targets

  tasks:

  - name: Create directory ~/5.2.0 if it does not exist

    file:

     path: ~/5.2.0

     state: directory


  - name: download kernel headers 1

    get_url:

      url:     https://kernel.ubuntu.com/~kernel-ppa/mainline/v5.2/linux-headers-5.2.0-
050200_5.2.0-050200.201907231526_all.deb

      dest: ~/5.2.0/linux-headers-5.2.0-050200_5.2.0-050200.201907231526_all.deb


  - name: download kernel header 2

    get_url:

      url:     https://kernel.ubuntu.com/~kernel-ppa/mainline/v5.2/linux-headers-5.2.0-
050200-generic_5.2.0-050200.201907231526_amd64.deb

      dest:                          ~/5.2.0/linux-headers-5.2.0-050200-generic_5.2.0-
050200.201907231526_amd64.deb


  - name: Download kernel image

    get_url:

      url:    https://kernel.ubuntu.com/~kernel-ppa/mainline/v5.2/linux-image-unsigned-
5.2.0-050200-generic_5.2.0-050200.201907231526_amd64.deb

      dest:                   ~/5.2.0/linux-image-unsigned-5.2.0-050200-generic_5.2.0-
050200.201907231526_amd64.deb


  - name: Download kernel modules

    get_url:

      url:     https://kernel.ubuntu.com/~kernel-ppa/mainline/v5.2/linux-modules-5.2.0-
050200-generic_5.2.0-050200.201907231526_amd64.deb

      dest:                          ~/5.2.0/linux-modules-5.2.0-050200-generic_5.2.0-
050200.201907231526_amd64.deb
```

```
  - name: Install new Kernel headers 1

    become: true

    shell:      dpkg      -i      /home/sysadm/5.2.0/linux-headers-5.2.0-050200_5.2.0-
050200.201907231526_all.deb


  - name: Install new Kernel headers 2

    become: true

    shell:   dpkg   -i   /home/sysadm/5.2.0/linux-headers-5.2.0-050200-generic_5.2.0-
050200.201907231526_amd64.deb


  - name: Install new Kernel modules

    become: true

    shell:   dpkg   -i   /home/sysadm/5.2.0/linux-modules-5.2.0-050200-generic_5.2.0-
050200.201907231526_amd64.deb


  - name: Install new Kernel image

    become: true

    shell: dpkg -i /home/sysadm/5.2.0/linux-image-unsigned-5.2.0-050200-generic_5.2.0-
050200.201907231526_amd64.deb


  - name: Update grub

    become: true

    command: update-grub

    register: task_result
```

14.2 HSS Playbook - Test 4

```
- name: hss_config_playbook
  #gather_facts: falseS
  hosts: targets #10.5.10.172
  #remote_user: sysadm

  tasks:
  - name: Update and upgrade apt packages
    become: true
    apt:
     upgrade: yes
     update_cache: yes

  - name: Reboot a slow machine that might have lots of updates to apply
    become: true
    reboot:
     post_reboot_delay: 35
     reboot_timeout: 300

  - name: Change FQDN
    become: true
    lineinfile:
     path: /etc/hosts
     insertafter: "127.0.0.1       localhost"
     line: "127.0.1.1   hss.OpenAir5G.Alliance   hss"

  - name: git download
    git:
     repo: https://github.com/OPENAIRINTERFACE/openair-cn.git
     dest: ~/openair-cn
     clone: yes

  - name: Edit cassandra build file
```

```
    replace:
     path: /home/sysadm/openair-cn/scripts/build_cassandra
     regexp: "return  0"
     replace: "#return  0"

  - name: Edit cassandra build_helper file (corrects building ansible issue)
    replace:
     path: /home/sysadm/openair-cn/build/tools/build_helper
     regexp: "fullpath=`readlink -f $BASH_SOURCE`"
     replace: "fullpath=`/home/sysadm/openair-cn`"

  - name: install cassandra
    command: /home/sysadm/openair-cn/scripts/build_cassandra --check-installed-software
--force

  - name: install hss_rel14 dependencies
    shell:  cd  ~/openair-cn/scripts/;  ./build_hss_rel14  --check-installed-software  --
force

  - name: build hss_rel14
    shell:  ~/openair-cn/scripts/build_hss_rel14 --clean

  - name: Populate tables 1
    shell: cqlsh --file ~/openair-cn/src/hss_rel14/db/oai_db.cql 127.0.0.1

  - name: Populate tables 2
    shell: ~/openair-cn/scripts/data_provisioning_users --apn default --apn2 internet -
-key  6874736969202073796d4b2079650a73  --imsi-first  208920100001100  --msisdn-first
33638020000   --mme-identity   mme.OpenAir5G.Alliance   --no-of-users   20   --realm
OpenAir5G.Alliance --truncate True --verbose True --cassandra-cluster 127.0.0.1

  - name: Populate tables 3
    shell:   ~/openair-cn/scripts/data_provisioning_mme   --id   1   --mme-identity
mme.OpenAir5G.Alliance --realm OpenAir5G.Alliance --ue-reachability 1 --truncate True
--verbose True
```

```
- name: Create directory /usr/local/etc/oai if it does not exist
  become: true
  file:
   path: /usr/local/etc/oai
   state: directory

- name: Create directory /usr/local/etc/oai/freeDiameter if it does not exist
  become: true
  file:
   path: /usr/local/etc/oai/freeDiameter
   state: directory

- name: Copy config file hss_rel14.conf.conf
  become: true
  copy:
   src: /home/sysadm/openair-cn/etc/hss_rel14.conf
   dest: /usr/local/etc/oai/hss_rel14.conf
   remote_src: yes

- name: Copy config file hss_rel14.json.conf
  become: true
  copy:
   src: /home/sysadm/openair-cn/etc/hss_rel14.json
   dest: /usr/local/etc/oai/hss_rel14.json
   remote_src: yes

- name: Copy config file hss_rel14_fd.conf
  become: true
  copy:
   src: /home/sysadm/openair-cn/etc/hss_rel14_fd.conf
   dest: /usr/local/etc/oai/freeDiameter/hss_rel14_fd.conf
   remote_src: yes
  - name: Copy config file acl.conf
    become: true
    copy:
     src: /home/sysadm/openair-cn/etc/acl.conf
     dest: /usr/local/etc/oai/freeDiameter/acl.conf
     remote_src: yes
## Modificaciones hss
  - name: Modifying config file hss_rel14.conf line 24
    become: true
    replace:
     path: /usr/local/etc/oai/hss_rel14.conf
     regexp: "@cassandra_Server_IP@"
     replace: "127.0.0.1"

  - name: Modifying config file hss_rel14.json line 2
    become: true
    replace:
     path: /usr/local/etc/oai/hss_rel14.json
     regexp: "@PREFIX@/freeDiameter/hss_rel14_fd.conf"
     replace: "/usr/local/etc/oai/freeDiameter/hss_rel14_fd.conf"

  - name: Modifying config file hss_rel14.json line 3
    become: true
    replace:
     path: /usr/local/etc/oai/hss_rel14.json
     regexp: "@HSS_FQDN@"
     replace: "hss.OpenAir5G.Alliance"

  - name: Modifying config file hss_rel14.json line 4
    become: true
    replace:
     path: /usr/local/etc/oai/hss_rel14.json
     regexp: "@REALM@"
```

```
   replace: "OpenAir5G.Alliance"

 - name: Modifying config file hss_rel14.json line 10
   become: true
   replace:
    path: /usr/local/etc/oai/hss_rel14.json
    regexp: "@cassandra_Server_IP@"
    replace: "127.0.0.1"

 - name: Modifying config file hss_rel14.json line 16
   become: true
   replace:
    path: /usr/local/etc/oai/hss_rel14.json
    regexp: "@OP_KEY@"
    replace: ""

 - name: Modifying config file hss_rel14.json line 17
   become: true
   replace:
    path: /usr/local/etc/oai/hss_rel14.json
    regexp: "@ROAMING_ALLOWED@"
    replace: "true"

 - name: Modifying config file acl.conf line 19
   become: true
   replace:
    path: /usr/local/etc/oai/freeDiameter/acl.conf
    regexp: "ALLOW_OLD_TLS   [*].@REALM@  [*].localdomain [*].test3gpp.net"
    replace: "ALLOW_OLD_TLS  OpenAir5G.Alliance"

 - name: Modifying config file hss_rel1_fd.conf line 7
   become: true
   replace:
    path: /usr/local/etc/oai/freeDiameter/hss_rel14_fd.conf
    regexp: "@HSS_FQDN@"
    replace: "hss.OpenAir5G.Alliance"

- name: Modifying config file hss_rel1_fd.conf line 11
  become: true
  replace:
    path: /usr/local/etc/oai/freeDiameter/hss_rel14_fd.conf
    regexp: "@REALM@"
    replace: "OpenAir5G.Alliance"

- name: Modifying config file hss_rel1_fd.conf line 16
  become: true
  replace:
    path: /usr/local/etc/oai/freeDiameter/hss_rel14_fd.conf
    regexp: "@PREFIX@/freeDiameter/hss.cert.pem"
    replace: "/usr/local/etc/oai/freeDiameter/hss.cert.pem"

- name: Modifying config file hss_rel1_fd.conf line 16
  become: true
  replace:
    path: /usr/local/etc/oai/freeDiameter/hss_rel14_fd.conf
    regexp: "@PREFIX@/freeDiameter/hss.key.pem"
    replace: "/usr/local/etc/oai/freeDiameter/hss.key.pem"

- name: Modifying config file hss_rel1_fd.conf line 17
  become: true
  replace:
    path: /usr/local/etc/oai/freeDiameter/hss_rel14_fd.conf
    regexp: "@PREFIX@/freeDiameter/cacert.pem"
    replace: "/usr/local/etc/oai/freeDiameter/cacert.pem"

- name: Modifying config file hss_rel1_fd.conf line 75
```

```
    become: true

    replace:

     path: /usr/local/etc/oai/freeDiameter/hss_rel14_fd.conf

     regexp: "@PREFIX@/freeDiameter/acl.conf"

     replace: "/usr/local/etc/oai/freeDiameter/acl.conf"


  - name: Get FreeDiameter Certificates

    shell:  cd  /home/sysadm/openair-cn/scripts/;  sudo  ./check_mme_s6a_certificate
/usr/local/etc/oai/freeDiameter                                mme.OpenAir5G.Alliance;
../src/hss_rel14/bin/make_certs.sh hss OpenAir5G.Alliance /usr/local/etc/oai



  - name: load OP key  # Anadir comando comentado a BBDD

    shell: oai_hss -j /usr/local/etc/oai/hss_rel14.json --onlyloadkey #cqlsh -e "USE
vhss;    UPDATE    users_imsi    SET    OPc='504f20634f6320504f50206363500a4f'WHERE
imsi='208920100001101'; exit;"
```

14.3 MME Playbook - Test 4

```
- name: mme_config_playbook
  #gather_facts: falseS
  hosts: targets #10.5.10.13
  #remote_user: sysadm

  tasks:
  - name: Update and upgrade apt packages
    become: true
    apt:
     upgrade: yes
     update_cache: yes

  - name: Reboot a slow machine that might have lots of updates to apply
    become: true
    reboot:
     post_reboot_delay: 35
     reboot_timeout: 300

  - name: Change FQDN
    become: true
    lineinfile:
     path: /etc/hosts
     insertafter: "127.0.0.1       localhost"
     line: "127.0.1.1   mme.OpenAir5G.Alliance   mme"

  - name: git download
    git:
     repo: https://github.com/OPENAIRINTERFACE/openair-cn.git
     dest: ~/openair-cn
     clone: yes

  - name: install software dependencies mme
```

```
    shell: cd ~/openair-cn/scripts/; ./build_mme --check-installed-software --force

  - name: build mme
    command:  ~/openair-cn/scripts/build_mme --clean

  - name: Create directory /usr/local/etc/oai if it does not exist
    become: true
    file:
     path: /usr/local/etc/oai
     state: directory

  - name: Create directory /usr/local/etc/oai/freeDiameter if it does not exist
    become: true
    file:
     path: /usr/local/etc/oai/freeDiameter
     state: directory

  - name: Copy config file mme.conf
    become: true
    copy:
     src: /home/sysadm/openair-cn/etc/mme.conf
     dest: /usr/local/etc/oai/mme.conf
     remote_src: yes

# ## Modificaciones mme
  - name: Modifying config file mme.conf line 3
    become: true
    replace:
     path: /usr/local/etc/oai/mme.conf
     regexp: "@REALM@"
     replace: "OpenAir5G.Alliance"

  - name: Modifying config file mme.conf line 4
```

```
    become: true
    replace:
     path: /usr/local/etc/oai/mme.conf
     regexp: "@INSTANCE@"
     replace: "0"

  - name: Modifying config file mme.conf line 5
    become: true
    replace:
     path: /usr/local/etc/oai/mme.conf
     regexp: "@PID_DIRECTORY@"
     replace: "/var/run"

  - name: Modifying config file mme.conf line 35
    become: true
    replace:
     path: /usr/local/etc/oai/mme.conf
     regexp: "@PREFIX@/freeDiameter/mme_fd.conf"
     replace: "/usr/local/etc/oai/freeDiameter/mme_fd.conf"

  - name: Modifying config file mme.conf line 36
    become: true
    replace:
     path: /usr/local/etc/oai/mme.conf
     regexp: "@HSS_HOSTNAME@"
     replace: "hss"

  - name: Modifying config file mme.conf 51
    become: true
    replace:
     path: /usr/local/etc/oai/mme.conf
     regexp: '{MCC="@MCC@" ; MNC="@MNC@"; MME_GID="@MME_GID@" ; MME_CODE="@MME_CODE@"; 
}'
     replace: '{MCC="34" ; MNC="94"; MME_GID="4" ; MME_CODE="1"; }'
```

```
- name: Modifying config file mme.conf line 55
  become: true
  replace:
   path: /usr/local/etc/oai/mme.conf
   regexp: '{MCC="@MCC@" ; MNC="@MNC@";  TAC = "@TAC_0@"; },'
   replace: '{MCC="34" ; MNC="94";  TAC = "1"; }'

- name: Modifying config file mme.conf line 56
  become: true
  replace:
   path: /usr/local/etc/oai/mme.conf
   regexp: '{MCC="@MCC@" ; MNC="@MNC@";  TAC = "@TAC_1@"; },'
   replace: ""

- name: Modifying config file mme.conf line 57
  become: true
  replace:
   path: /usr/local/etc/oai/mme.conf
   regexp: '{MCC="@MCC@" ; MNC="@MNC@";  TAC = "@TAC_2@"; }'
   replace: ""

- name: Modifying config file mme.conf line 77
  become: true
  replace:
   path: /usr/local/etc/oai/mme.conf
   regexp: "@MME_INTERFACE_NAME_FOR_S1_MME@"
   replace: "ens3:m1c"

- name: Modifying config file mme.conf line 78
  become: true
  replace:
   path: /usr/local/etc/oai/mme.conf
   regexp: "@MME_IPV4_ADDRESS_FOR_S1_MME@"
   replace: "172.56.56.2/16"

- name: Modifying config file mme.conf line 81
  become: true
  replace:
   path: /usr/local/etc/oai/mme.conf
   regexp: "@MME_INTERFACE_NAME_FOR_S11@"
   replace: "ens5"

- name: Modifying config file mme.conf line 82
  become: true
  replace:
   path: /usr/local/etc/oai/mme.conf
   regexp: "@MME_IPV4_ADDRESS_FOR_S11@"
   replace: "10.0.2.2/29"

- name: Modifying config file mme.conf line 87
  become: true
  replace:
   path: /usr/local/etc/oai/mme.conf
   regexp: "@MME_INTERFACE_NAME_FOR_S10@"
   replace: "lo"

- name: Modifying config file mme.conf line 88
  become: true
  replace:
   path: /usr/local/etc/oai/mme.conf
   regexp: "@MME_IPV4_ADDRESS_FOR_S10@"
   replace: "127.0.0.100/8"

- name: Modifying config file mme.conf line 97
  become: true
```

```
    replace:
     path: /usr/local/etc/oai/mme.conf
     regexp: "@OUTPUT@"
     replace: "CONSOLE"     # "/home/sysadm/mme.log"

  - name: Modifying config file mme.conf line 118
    become: true
    replace:
     path: /usr/local/etc/oai/mme.conf
     regexp:                                '{ID="tac-lb@TAC-LB_SGW_TEST_0@.tac-hb@TAC-
HB_SGW_TEST_0@.tac.epc.mnc001.mcc001.3gppnetwork.org"                          ;
SGW_IPV4_ADDRESS_FOR_S11="@SGW_IPV4_ADDRESS_FOR_S11_TEST_0@";},'
     replace:     '{ID="tac-lb01.tac-hb00.tac.epc.mnc092.mcc208.3gppnetwork.org"     ;
SGW_IPV4_ADDRESS_FOR_S11="127.0.0.30/8";}'

  - name: Modifying config file mme.conf line 119
    become: true
    replace:
     path: /usr/local/etc/oai/mme.conf
     regexp:                                     '{ID="tac-lb@TAC-LB_SGW_0@.tac-hb@TAC-
HB_SGW_0@.tac.epc.mnc@MNC3_SGW_0@.mcc@MCC_SGW_0@.3gppnetwork.org"                    ;
SGW_IPV4_ADDRESS_FOR_S11="@SGW_IPV4_ADDRESS_FOR_S11_0@";},'
     replace: ""

  - name: Modifying config file mme.conf line 120
    become: true
    replace:
     path: /usr/local/etc/oai/mme.conf
     regexp:                                     '{ID="tac-lb@TAC-LB_MME_0@.tac-hb@TAC-
HB_MME_0@.tac.epc.mnc@MNC3_MME_0@.mcc@MCC_MME_0@.3gppnetwork.org"                    ;
PEER_MME_IPV4_ADDRESS_FOR_S10="@PEER_MME_IPV4_ADDRESS_FOR_S10_0@";},'
     replace: ""


  - name: Modifying config file mme.conf line 121
    become: true
    replace:

     path: /usr/local/etc/oai/mme.conf

     regexp:                                    '{ID="tac-lb@TAC-LB_MME_1@.tac-hb@TAC-
HB_MME_1@.tac.epc.mnc@MNC3_MME_1@.mcc@MCC_MME_1@.3gppnetwork.org"                   ;
PEER_MME_IPV4_ADDRESS_FOR_S10="@PEER_MME_IPV4_ADDRESS_FOR_S10_1@";}'

     replace: ""


  - name: Creating an emty file mme_fd.conf

    become: true

    copy:

     content: ""

     dest: /usr/local/etc/oai/freeDiameter/mme_fd.conf

     force: no



  - name: Editing /usr/local/etc/oai/freeDiameter/mme_fd.conf

    become: true

    blockinfile:

     marker_begin: ""

     marker_end: ""

     marker: ""

     path: /usr/local/etc/oai/freeDiameter/mme_fd.conf

     insertafter: ""

     block: |

       # -------- Local ---------


       # Uncomment if the framework cannot resolv it.

       Identity = "mme.OpenAir5G.Alliance";

       Realm = "OpenAir5G.Alliance";


       # TLS configuration (see previous section)

       TLS_Cred = "/usr/local/etc/oai/freeDiameter/mme.cert.pem",

                  "/usr/local/etc/oai/freeDiameter/mme.key.pem";

       TLS_CA   = "/usr/local/etc/oai/freeDiameter/mme.cacert.pem";
```

```
# Disable use of TCP protocol (only listen and connect in SCTP)
# Default : TCP enabled
No_SCTP;

# This option is ignored if freeDiameter is compiled with DISABLE_SCTP option.
# Prefer TCP instead of SCTP for establishing new connections.
# This setting may be overwritten per peer in peer configuration blocs.
# Default : SCTP is attempted first.
Prefer_TCP;



No_IPv6;

# Overwrite the number of SCTP streams. This value should be kept low,
# especially if you are using TLS over SCTP, because it consumes a lot of
# resources in that case. See tickets 19 and 27 for some additional details on
# this.
# Limit the number of SCTP streams
SCTP_streams = 3;



# By default, freeDiameter acts as a Diameter Relay Agent by forwarding all
# messages it cannot handle locally. This parameter disables this behavior.
NoRelay;


# Use RFC3588 method for TLS protection, where TLS is negociated after CER/CEA exchange is completed
# on the unsecure connection. The alternative is RFC6733 mechanism, where TLS protects also the
# CER/CEA exchange on a dedicated secure port.
# This parameter only affects outgoing connections.
# The setting can be also defined per-peer (see Peers configuration section).
# Default: use RFC6733 method with separate port for TLS.
```

```
AppServThreads = 4;

# Specify the addresses on which to bind the listening server. This must be
# specified if the framework is unable to auto-detect these addresses, or if the
# auto-detected values are incorrect. Note that the list of addresses is sent
# in CER or CEA message, so one should pay attention to this parameter if some
# adresses should be kept hidden.
ListenOn = "10.0.1.2";

Port = 3870;
SecPort = 5870;

# -------- Extensions ---------

# Uncomment (and create rtd.conf) to specify routing table for this peer.
LoadExtension = "/usr/local/lib/freeDiameter/dict_3gpp2_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_draftload_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_etsi283034_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_rfc4004_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_rfc4006bis_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_rfc4072_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_rfc4590_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_rfc5447_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_rfc5580_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_rfc5777_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_rfc5778_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_rfc6734_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_rfc6942_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_rfc7155_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_rfc7683_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_rfc7944_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29061_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29128_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29154_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29173_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29212_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29214_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29215_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29217_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29229_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29272_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29273_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29329_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29336_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29337_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29338_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29343_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29344_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29345_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29368_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts29468_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_ts32299_avps.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_S6as6d.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_S6t.fdx";
LoadExtension = "/usr/local/lib/freeDiameter/dict_S6c.fdx";
#-------- Peers ---------

# The framework will actively attempt to establish and maintain a connection
# with the peers listed here.
# For only accepting incoming connections, see the acl_wl.fx extension.

# ConnectPeer
# Declare a remote peer to which this peer must maintain a connection.
# In addition, this allows specifying non-default parameters for this peer only
# (for example disable SCTP with this peer, or use RFC3588-flavour TLS).
```

```
        # Note that by default, if a peer is not listed as a ConnectPeer entry, an
        # incoming connection from this peer will be rejected. If you want to accept
        # incoming connections from other peers, see the acl_wl.fdx? extension which
        # allows exactly this.

        ConnectPeer=  "hss.OpenAir5G.Alliance"  {  ConnectTo  =  "10.0.1.3";  No_SCTP  ;
No_IPv6; Prefer_TCP; No_TLS; port = 3868;  realm = "OpenAir5G.Alliance";};



  - name: Get FreeDiameter Certificates
    become: true
    shell:     cd     /home/sysadm/openair-cn/scripts;     ./check_mme_s6a_certificate
/usr/local/etc/oai/freeDiameter                               mme.OpenAir5G.Alliance;
../src/hss_rel14/bin/make_certs.sh hss OpenAir5G.Alliance /usr/local/etc/oai

  - name: Create MME S1-C interface
    become: true
    command: ifconfig ens3:m1c 172.56.56.2 up
```

14.4 SPGW Playbook - Test 4

```
- name: spgw_playbook
  #gather_facts: falseS
  hosts: targets #10.5.10.14
  #remote_user: sysadm

  tasks:
  - name: Update and upgrade apt packages
    become: true
    apt:
     upgrade: yes
     update_cache: yes

  - name: Reboot a slow machine that might have lots of updates to apply
    become: true
    reboot:
     post_reboot_delay: 35
     reboot_timeout: 300

  - name: git download
    git:
     repo: https://github.com/OPENAIRINTERFACE/openair-cn-cups.git
     dest: /home/sysadm/openair-cn-cups
     clone: yes
     #version: v1.0.1

  - name: Install spgw_u dependencies
    shell: /home/sysadm/openair-cn-cups/build/scripts/build_spgwu -I -f

  - name: Build spgw_u
    shell: /home/sysadm/openair-cn-cups/build/scripts/build_spgwu -c -V -b Debug -j

  - name: Install spgw_c dependencies
```

```
    shell: /home/sysadm/openair-cn-cups/build/scripts/build_spgwc -I -f

  - name: Build spgw_c
    shell: /home/sysadm/openair-cn-cups/build/scripts/build_spgwc -c -V -b Debug -j

  - name: Copy config file spgw_u.conf
    become: true
    copy:
     src: /home/sysadm/openair-cn-cups/etc/spgw_u.conf
     dest: /usr/local/etc/oai/spgw_u.conf
     remote_src: yes

  - name: Copy config file spgw_c.conf
    become: true
    copy:
     src: /home/sysadm/openair-cn-cups/etc/spgw_c.conf
     dest: /usr/local/etc/oai/spgw_c.conf
     remote_src: yes

## SPGW_U configuration
  - name: Modifying config file spgw_u.conf line 23
    become: true
    replace:
     path: /usr/local/etc/oai/spgw_u.conf
     regexp: "@INSTANCE@"
     replace: "0"

  - name: Modifying config file spgw_u.conf line 24
    become: true
    replace:
     path: /usr/local/etc/oai/spgw_u.conf
     regexp: "@PID_DIRECTORY@"
     replace: "/var/run"
```

```
- name: Modifying config file spgw_u.conf line 59
  become: true
  replace:
   path: /usr/local/etc/oai/spgw_u.conf
   regexp: "@SGW_INTERFACE_NAME_FOR_S1U_S12_S4_UP@"
   replace: "ens3:s1u"

- name: Modifying config file spgw_u.conf line 60
  become: true
  replace:
   path: /usr/local/etc/oai/spgw_u.conf
   before: "SX :"
   regexp: "read"
   replace: "172.57.57.2/16"

- name: Modifying config file spgw_u.conf line 72
  become: true
  replace:
   path: /usr/local/etc/oai/spgw_u.conf
   regexp: "@SGW_INTERFACE_NAME_FOR_SX@"
   replace: "ens3:sxu"

- name: Modifying config file spgw_u.conf line 73
  become: true
  replace:
   path: /usr/local/etc/oai/spgw_u.conf
   before: "SGI :"
   regexp: "read"
   replace: "172.55.55.102/16"

- name: Modifying config file spgw_u.conf line 85
  become: true
```

```
    replace:
     path: /usr/local/etc/oai/spgw_u.conf
     regexp: "@SGW_INTERFACE_NAME_FOR_SGI@"
     replace: "ens3"

  - name: Modifying config file spgw_u.conf line 86
    become: true
    replace:
     path: /usr/local/etc/oai/spgw_u.conf
     before: "PDN_NETWORK_LIST  ="
     regexp: "read"
     replace: "10.5.10.14/16"

## SPGW_C configuration
  - name: Modifying config filespgw_c.conf line 23
    become: true
    replace:
     path: /usr/local/etc/oai/spgw_c.conf
     before: "P-GW ="
     regexp: "@INSTANCE@"
     replace: "0"

  - name: Modifying config file filespgw_c.conf line 24
    become: true
    replace:
     path: /usr/local/etc/oai/spgw_c.conf
     before: "P-GW ="
     regexp: "@PID_DIRECTORY@"
     replace: "/var/run"

  - name: Modifying config file filespgw_c.conf line 71
    become: true
    replace:
```

```
    path: /usr/local/etc/oai/spgw_c.conf
    regexp: "@SGW_INTERFACE_NAME_FOR_S11@"
    replace: "ens4"

- name: Modifying config file filespgw_c.conf line 72
  become: true
  replace:
    path: /usr/local/etc/oai/spgw_c.conf
    after: "S11_CP :"
    before: "        S5_S8_CP :"
    regexp: "read"
    replace: "10.0.2.3/29"

- name: Modifying config file filespgw_c.conf line 84
  become: true
  replace:
    path: /usr/local/etc/oai/spgw_c.conf
    regexp: "@SGW_INTERFACE_NAME_FOR_S5_S8_CP@"
    replace: "ens3:s5c"

- name: Modifying config file filespgw_c.conf line 85
  become: true
  replace:
    path: /usr/local/etc/oai/spgw_c.conf
    after: "        S5_S8_CP :"
    before: "P-GW ="
    regexp: "read"
    replace: "172.58.58.102/16"

- name: Modifying config filespgw_c.conf line 99
  become: true
  replace:
    path: /usr/local/etc/oai/spgw_c.conf
    after: "P-GW ="
    regexp: "@INSTANCE@"
    replace: "0"

- name: Modifying config file filespgw_c.conf line 100
  become: true
  replace:
    path: /usr/local/etc/oai/spgw_c.conf
    after: "P-GW ="
    regexp: "@PID_DIRECTORY@"
    replace: "/var/run"

- name: Modifying config file filespgw_c.conf line 147
  become: true
  replace:
    path: /usr/local/etc/oai/spgw_c.conf
    regexp: "@PGW_INTERFACE_NAME_FOR_S5_S8_CP@"
    replace: "ens3:p5c"

- name: Modifying config file filespgw_c.conf line 148
  become: true
  replace:
    path: /usr/local/etc/oai/spgw_c.conf
    before: "SX :"
    regexp: "read"
    replace: "172.58.58.101/16"

- name: Modifying config file filespgw_c.conf line 160
  become: true
  replace:
    path: /usr/local/etc/oai/spgw_c.conf
    regexp: "@PGW_INTERFACE_NAME_FOR_SX@"
    replace: "ens3:sxc"
```

```
- name: Modifying config file filespgw_c.conf line 161
  become: true
  replace:
   path: /usr/local/etc/oai/spgw_c.conf
   before: "IP_ADDRESS_POOL :"
   regexp: "read"
   replace: "172.55.55.101/16"

- name: Modifying config file filespgw_c.conf line 203
  become: true
  replace:
   path: /usr/local/etc/oai/spgw_c.conf
   regexp: "DEFAULT_DNS_IPV4_ADDRESS@"
   replace: "8.8.8.8"

- name: Modifying config file filespgw_c.conf line 204
  become: true
  replace:
   path: /usr/local/etc/oai/spgw_c.conf
   regexp: "@DEFAULT_DNS_SEC_IPV4_ADDRESS@"
   replace: "4.4.4.4"

- name: Create SPGW-U SXab interface
  become: true
  command: ifconfig ens3:sxu 172.55.55.102 up

- name: Create SPGW-C SXab interface
  become: true
  command: ifconfig ens3:sxc 172.55.55.101 up

- name: Create SGW-C S5S8 interface
  become: true
```

```
    command: ifconfig ens3:s5c 172.58.58.102 up

  - name: Create PGW-C S5S8 interface
    become: true
    command: ifconfig ens3:p5c 172.58.58.101 up

  - name: Create SPGW-U S1-u interface
    become: true
    command: ifconfig ens3:s1u 172.57.57.2 up
```

14.5 eNB Playbook -Test 5

```
- name: eNB_config_playbook
  hosts: targets
  #remote_user: sysadm

  tasks:
  - name: Update and upgrade apt packages
    become: true
    apt:
     upgrade: yes
     update_cache: yes

  - name: Add UHD repository and update packages
    become: true
    apt_repository:
     repo: ppa:ettusresearch/uhd
     state: present
     update_cache: yes

  - name: Installed USRP drivers and required packages
    become: true
    apt: name={{ item }} state=installed
    with_items:
    - libuhd-dev
    - libuhd003
    - uhd-host
    - build-essential
    - cmake
    tags:
     - packages

  - name: Reboot
    become: true
    reboot:
     post_reboot_delay: 35
     reboot_timeout: 300

  - name: git download
    git:
     repo: https://gitlab.eurecom.fr/oai/openairinterface5g.git
     dest: ~/openairinterface5g
     clone: yes
     version: develop

  - name: Install eNB dependencies and Build eNB for USRP
    shell:       cd       /home/sysadm/openairinterface5g;       source       oaienv;
/home/sysadm/openairinterface5g/cmake_targets/build_oai -I --eNB -x -w USRP;
    # -I: installs required packages.
    # --eNB: installs eNB, i.e., lte-softmodem.
    # -x: adds a software oscilloscope feature to the produced binaries.
    # -w: adds the hardware support.

## eNB configuration
  - name: Modifying config file enb.band7.tm1.50PRB.usrpb210.conf line 3
    replace:
     path:                /home/sysadm/openairinterface5g/targets/PROJECTS/GENERIC-LTE-
EPC/CONF/enb.band7.tm1.50PRB.usrpb210.conf
     regexp: 'Asn1_verbosity = "none";'
     replace: 'Asn1_verbosity = "info";'

  - name: Modifying config file enb.band7.tm1.50PRB.usrpb210.conf line 18
    replace:
     path:                /home/sysadm/openairinterface5g/targets/PROJECTS/GENERIC-LTE-
EPC/CONF/enb.band7.tm1.50PRB.usrpb210.conf
     before: "tr_s_preference"
     regexp: 'mcc = 208; mnc = 93; mnc_length = 2;'
     replace: 'mcc = 34; mnc = 94; mnc_length = 2;'
  - name: Modifying config file enb.band7.tm1.50PRB.usrpb210.conf line 175

    replace:

     path:                /home/sysadm/openairinterface5g/targets/PROJECTS/GENERIC-LTE-
EPC/CONF/enb.band7.tm1.50PRB.usrpb210.conf

     regexp: 'ipv4       = "10.64.93.19";'

     replace: 'ipv4       = "10.5.10.15";'


  - name: Modifying config file enb.band7.tm1.50PRB.usrpb210.conf line 191

    replace:

     path:                /home/sysadm/openairinterface5g/targets/PROJECTS/GENERIC-LTE-
EPC/CONF/enb.band7.tm1.50PRB.usrpb210.conf

     regexp: 'ENB_INTERFACE_NAME_FOR_S1_MME            = "eno1";'

     replace: 'ENB_INTERFACE_NAME_FOR_S1_MME            = "eno1:e1c";'


  - name: Modifying config file enb.band7.tm1.50PRB.usrpb210.conf line 192

    replace:

     path:                /home/sysadm/openairinterface5g/targets/PROJECTS/GENERIC-LTE-
EPC/CONF/enb.band7.tm1.50PRB.usrpb210.conf

     after: "ENB_INTERFACE_NAME_FOR_S1_MME"

     before: "ENB_IPV4_ADDRESS_FOR_S1U"

     regexp: '"10.64.45.62/23";'

     replace: '"172.56.56.3/16";'


  - name: Modifying config file enb.band7.tm1.50PRB.usrpb210.conf line 193

    replace:

     path:                /home/sysadm/openairinterface5g/targets/PROJECTS/GENERIC-LTE-
EPC/CONF/enb.band7.tm1.50PRB.usrpb210.conf

     regexp: 'ENB_INTERFACE_NAME_FOR_S1U               = "eno1";'

     replace: 'ENB_INTERFACE_NAME_FOR_S1_MME            = "eno1:e1u";'


  - name: Modifying config file enb.band7.tm1.50PRB.usrpb210.conf line 194

    replace:

     path:                /home/sysadm/openairinterface5g/targets/PROJECTS/GENERIC-LTE-
EPC/CONF/enb.band7.tm1.50PRB.usrpb210.conf

     after: "ENB_INTERFACE_NAME_FOR_S1_MME"

     regexp: '"10.64.45.62/23";'
```

```
   replace: '"172.57.57.3/16";'

- name: Create eNB S1-C interface
  become: true
  command: ifconfig eno1:e1c 172.56.56.3 up

- name: Create eNB S1-u interface
  become: true
  command: ifconfig eno1:e1u 172.57.57.3 up
```