PROYECTO FIN DE GRADO

TITLE: 5G Core Network: Design, Implementation and Testing of a Virtualized Deployment

AUTOR/A: Salvador Fernández García-Morales

TITULACIÓN: Grado en Ingeniería Telemática

DIRECTOR/A: Marta Muriel Elduayen

TUTOR/A: Ana Belén García Hernando

DEPARTAMENTO: Departamento de Ingeniería Telemática y Electrónica (DTE)

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE/A: Julia María García Luengo

TUTOR/A: Ana Belén García Hernando

SECRETARIO/A: Pedro Castillejo Parrilla

Fecha de lectura:

Calificación:

El Secretario/La Secretaria,

Greetings

Estas son las últimas palabras de una historia que conlleva 7 años de duro esfuerzo y dedicación. Pocas palabras son para describir el orgullo que siento por haber formado parte de este gran proyecto, el cual he tenido el placer de compartir junto con mis tutores (Ana Belén, Pedro y Marta) sin los cuales esto no podría haberse materializado. Gracias a los tres por el enorme esfuerzo que habéis hecho y por la calidad humana que tenéis, por despertar en mí un interés escondido con cada clase de RCM y gracias también por las risas y bromas que tuvimos durante nuestras reuniones.

Gracias también a toda mi familia, pues sin ellos no estaría hoy aquí escribiendo estas letras. En especial a mis padres y a mi hermana, por haberme aguantado en mis malos días y por el incondicional apoyo que me habéis dado siempre. Gracias también a los que me habéis enseñado la vida y me habéis acogido siempre, a mis abuelos y a mis yayos que, aunque alguno no podrá verme graduarme, esto va por todos vosotros.

Gracias también a todos mis amigos incondicionales por haberme apoyado y animado, incluso a los que no veo tanto por ser míster ocupado. Especial mención a Tommy, Isa, Cris, Alejandra, Peter y Jairo por haberme ayudado a resolver ciertos aspectos de este proyecto.

Me gustaría dedicar las últimas palabras de este proyecto a una persona muy especial y que gracias a su enorme apoyo incondicional y condición humana se merece todo lo bueno que le pase. A mi novia Arantxa que la quiero con locura, esto va por ti.

Resumen

A pesar de que la tecnología 6G está comenzando a ser especificada, todavía se está trabajando en su implementación mientras que el 5G está evolucionando a su forma Standalone que es el 5G puro compuesto por las unidades radio y núcleo (*core*), sin usar ningún elemento arquitectónico implementado en tecnologías previas como el 4G.

Además, cuanto más crece la tecnología mayor rendimiento en términos de velocidad es necesario para manejar los nuevos servicios sobre 5G. Estos servicios han sido desarrollados siguiendo las necesidades específicas para las diferentes aplicaciones lo cual requiere de capacidades de red específicas bajo demanda. Para poder abordar este problema de forma económica y eficiente, el uso de la virtualización es necesario.

La virtualización del núcleo de la red 5G supone muchas ventajas como por ejemplo la utilización de recursos, ahorro de costes, mejora de la escalabilidad de la red, recuperación ante fallos o desastres, y simplificación del manejo de la infraestructura de TI (Tecnologías de la Información). Por estas razones, el objeto de este proyecto estará enfocado en el desarrollo e implementación de un núcleo de red 5G virtualizado que siga las especificaciones del 3GPP.

Para facilitar este despliegue, abordaremos la instalación y configuración de una nube OpenStack que nos permitirá utilizar herramientas de SDN (Redes Definidas por Software) y desplegar máquinas virtuales de forma más sencilla. Además, como parte de este proyecto, también nos ocuparemos de la instalación y configuración de Free5GC y UERANSIM para simular redes reales del núcleo 5G y estaciones de radio dentro de la plataforma OpenStack.

Abstract

Even though the 6G technology is being nowadays specified, its real deployment is underway while 5G is evolving into its Standalone version form which is the pure 5G composed by radio and core units, without using any architectural elements implemented in previous technologies like 4G.

In addition, the more the technology grows the better speed performance is needed to hold the newest services over 5G. These services have been developed following the needs for different applications which require specific network capabilities on demand. To address this issue in an efficient and inexpensive way, the use of virtualization technologies is a must.

Virtualization of 5G Core Network provides huge advantages such as efficient resource utilization, cost savings, network scalability improvements, disaster recovery and simplified IT infrastructure management. For these reasons, the aim of this project will focus on the development and deployment of a virtualized 5G Core Network solution which follows the 3GPP (3rd Generation Partnership Project) specifications.

To facilitate this deployment, we will cover the installation and configuration of an OpenStack cloud which allows us to use SDN (Software-defined Networks) tools and to deploy virtual machines easier. Additionally, as part of this project, we will also address the installation and configuration of Free5GC and UERANSIM to simulate real 5G Core Networks and radio stations within the OpenStack platform.

List of Figures

List of acronyms

3GPP	3rd Generation Partnership Project
5G-NR	5G New Radio
5GC	5G Core
AF	Application Function
AMF	Access and Mobility Management Function
API	Application programming interface
AUC	Authentication Center
AUSF	Authentication Server Function
AVX	Advanced Vector Extensions
BSS	Base Station Subsystem
CHF	Charging Function
CLI	Command Line Interface
CPC	Continued Packet Connectivity
CPU	Central processing unit
DM	Direct Management
DN	Data Network
DNS	Domain Name System
DPI	Deep Packet Inspection
EB	Exabyte
EIR	Equipment Identity Register
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
FDD	Frequency-division duplexing
FTP	File Transfer Protocol
GGSN	Gateway GPRS Support Node

GMSK	Gaussian Minimum Shift Keying
gNB	gNodeB
GPRS	General Packet Radio Service
GSM	Global System for Mobile
GTP	GPRS Tunnelling Protocol
GUI	Graphical User Interface
GUTI	Global Unique Temporary Identifier
HLR	Home Location Register
HSDPA	High-Speed Downlink Packet Access
HSPA	High-Speed Packet Access
HSUPA	High-Speed Uplink Packet Access
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol version 4
KVM	Kernel-based Virtual Machine
LAN	Local Area Network
LTE	Long Term Evolution
LTS	Long-term support
LVM	Logical Volume Manager
LXC	Linux Containers
MANO	Management and Orchestration
MME	Mobility Management Entity
MSC	Mobile Switching Center

N3IWF	Non-3GPP Interworking Function
N3IWUE	Non-3GPP Interworking User Equipment
NAS	Non-Access Stratum
NAT	Network Address Translation
NEF	Network Exposure Function
NF	Network Function
NFV	Network Functions Virtualization
NG-RAN	Next Generation Radio Access Network
NGAP	Next Generation Application Protocol
NRF	NF Repository Function
NSS	Network Switching Subsystem
NSSF	Network Slice Selection Function
OP	Operator Code
OPC	Derived Operator Code
OS	Operating System
PC	Personal Computer
PCF	Policy and Charging Function
PDN-GW	Packet Data Network Gateway
PDU	Packet Data Unit
QCOW2	QEMU Copy On Write version 2
QEMU	Quick Emulator
QoS	Quality of Service
RAN	Radio Area Network
RDP	Remote Destock Protocol
RHEL	Red Hat Enterprise Linux
SA	Stand Alone
SCTP	Stream Control Transmission Protocol

SDG	Sustainable Development Goals
SDN	Software-defined Networks
SDR	Software-defined Radio
GW	Gateway
SGSN	Serving GPRS support node
SGW	Serving Gateway
SMF	Session Management Function
SMS	Short Message Service
SSC	Session and Service Continuity
SSH	Secure Shell Protocol
SSID	Service Set Identifier
TDMA	Time Division Multiple Access
TI	Tecnologías de la Información
TS	Technical Specification
UDM	Unified Data Management
UDR	Unified Data Repository
UE	User Equipment
UERANSIM	UE and RAN Simulator
ULCL	Uplink Classifier
UML	User Mode Linux
UMTS	Universal Mobile Telecommunications Service
UPF	User Plane Function
UPM	Technical University of Madrid
URL	Uniform Resource Locator
USB	Universal Serial Bus
UTRA	Universal Terrestrial radio Access
UTRAN	UMTS Terrestrial Radio Access Network

VLR	Visitor Location Register
VM	Virtual Machine
VMM	Virtual Machine Monitor
VNIC	Virtual Network Interface Card
VPN	Virtual Private Network

Contents

1. Introduction

The 5th Generation (5G) technology is the natural evolution process and development of mobile networks which is more focused on data networks and switching than its predecessors.

As a peek to the past, a summarized history of mobile networks will be covered, to explain the technological background of this project and why this technology is crucial to improve the mobile services offered to users.

1.1. Technological framework and motivation

1.1.1. Evolution of previous mobile systems: 1G to 4G

Mobile communications have been evolving over the years in concordance with the historical epoch situation, in which factors like economy, technology and social needs were totally dissimilar. For this reason, networks have been classified in different generations of mobile systems from the first voice services (1G) to the massive data management over 5G. This evolution is depicted in Figure 1.

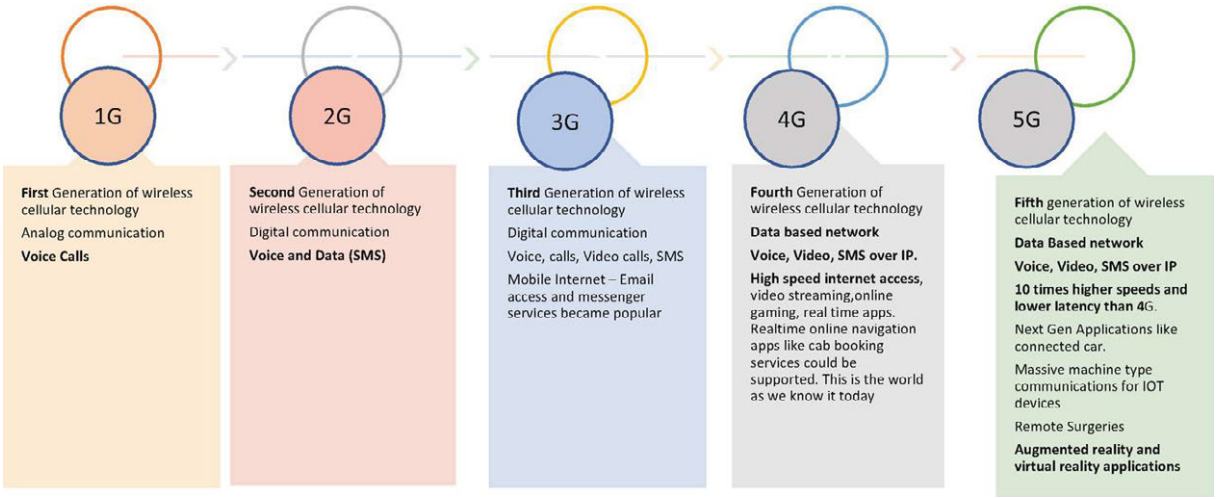

Figure 1. Generations of mobile systems from 1G to 5G by [1]

In the 1980's communications started to switch from wired technologies to wireless ones. The main commercial target were companies which have both an imperative need to communicate and enough funds to afford the high costs of voice services. In regards with the insights of these services, those used dedicated radio links to each communication which made those services extremely inefficient and poorly scalable.

In the 1990's ETSI (European Telecommunications Standards Institute) developed GSM (**Global System for Mobile**) which is the standard of the second generation (2G) of digital cellular networks. This network was designed to use dedicated radio links for voice only services but soon they evolved to support some additional services like SMS specified for the first time in 1996 by ETSI at [2].

The GSM network consists of two parts, one is the radio part known as BSS (Base Station Subsystem) which made use of GMSK (Gaussian Minimum Shift Keying) modulation with TDMA (Time Division Multiple Access) signaling over FDD (Frequency Division Duplex) carriers. The other part is the core network also called NSS (Network Switching Subsystem) that was composed of different elements such as the MSC (Mobile Switching Center) in charge of switch interconnection between clients of the net, the HLR (Home Location Register) which contains the info of mobile subscribers, the VLR (Visitor Location Register) to register when a subscriber changes location, AUC (Authentication Center) to define policies regarding authentication of mobile users and EIR (Equipment Identity Register) which is a database used to identify unequivocally a mobile device by the IMEI (International Mobile Equipment Identity). See in Figure 2 a graphical representation of the architecture of a GSM network.

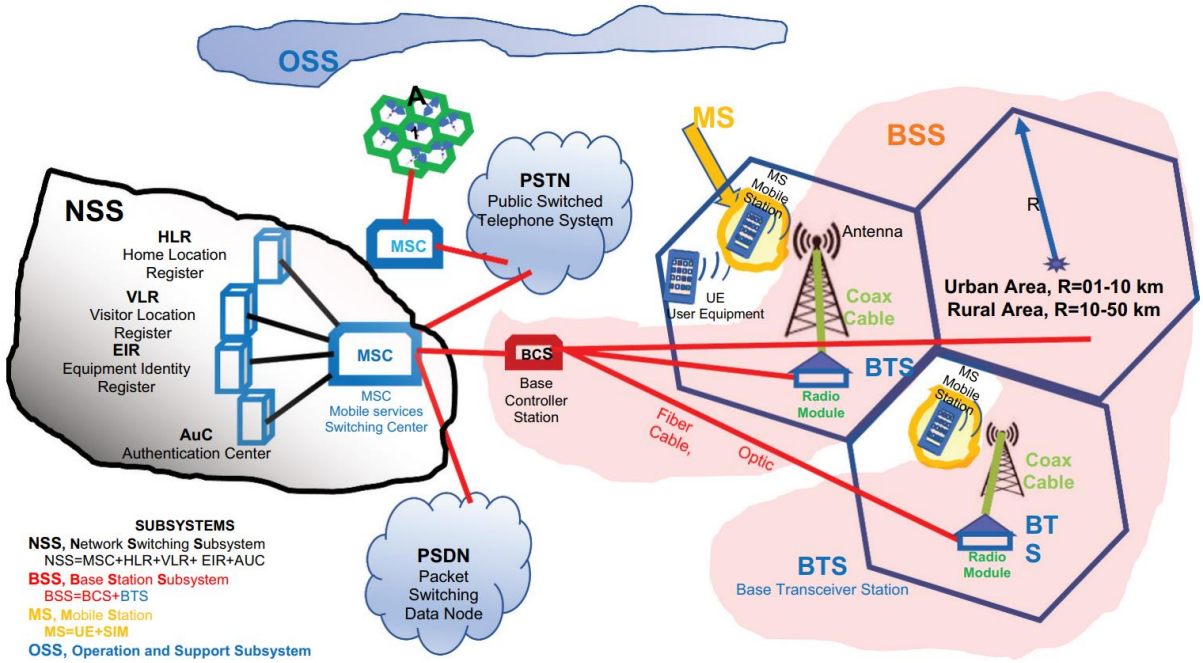

Figure 2. GSM architecture from [3]

The Internet era had already started up in the 1990's and with it, the 3GPP [4] which was founded in 1998 to produce technical specifications and reports for a 3G Mobile System based on evolved GSM core networks and the radio access technologies supported at that time, e.g. UTRA (Universal Terrestrial radio Access).

This goal was achieved by the launch of Release 99 [5] in December 1999 as marked in [6] which is a set of Technical Specifications for the first 3G Mobile Network System called GSM/UMTS (Universal Mobile Telecommunications Service). This system continuously evolved following the rules specified by its predecessors as for instance Release 4 for Bearer-Independent Core Network, Release 5 for HSDPA (High-Speed Downlink Packet Access), Release 6 for HSUPA (High-Speed Uplink Packet Access) and Release 7 for even faster HSPA (High-Speed Packet Access) and CPC (Continued Packet Connectivity)

As depicted in Figure 3 the core and radio systems are similar to GSM and in addition the UTRAN (UMTS Terrestrial Radio Access Network) was made to be compatible with classical

GSM networks. Due to its compatibility with GSM, 3GPP solutions became popular within the telecommunications industry. In fact, the increasing demand of internet services over IP result in the establishment of core components that supported packet networks, i.e. SGSN (Serving GPRS(General Packet Radio Service) support node) and GGSN (Gateway GPRS Support Node).

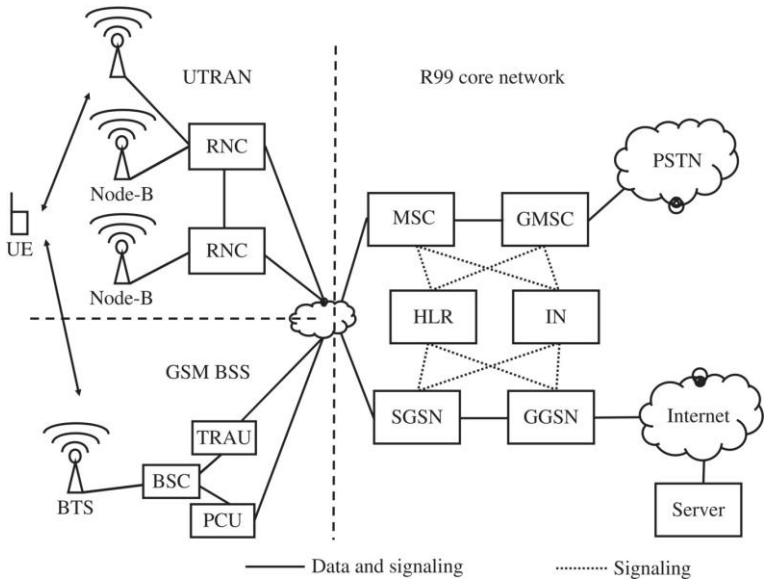

Figure 3. 3G Mobile network architecture specified in Release 99 by [7]

As per the mobility report published by Ericsson in November 2012 [8] it was predicted that by the end of 2012, total mobile subscriptions would be around 6.6 billion and by the end of 2018, were expected to reach 9.3 billion (related graph in Figure 4). These reports forced the creation of new specifications by the brand of 4G. Due to the high demand of subscribers over the data mobile networks and the increasing data consumption rates, 3G networks were unsustainable in the long term as they are not ready to manage such a high quantity of users and neither to support high-speed data rates.

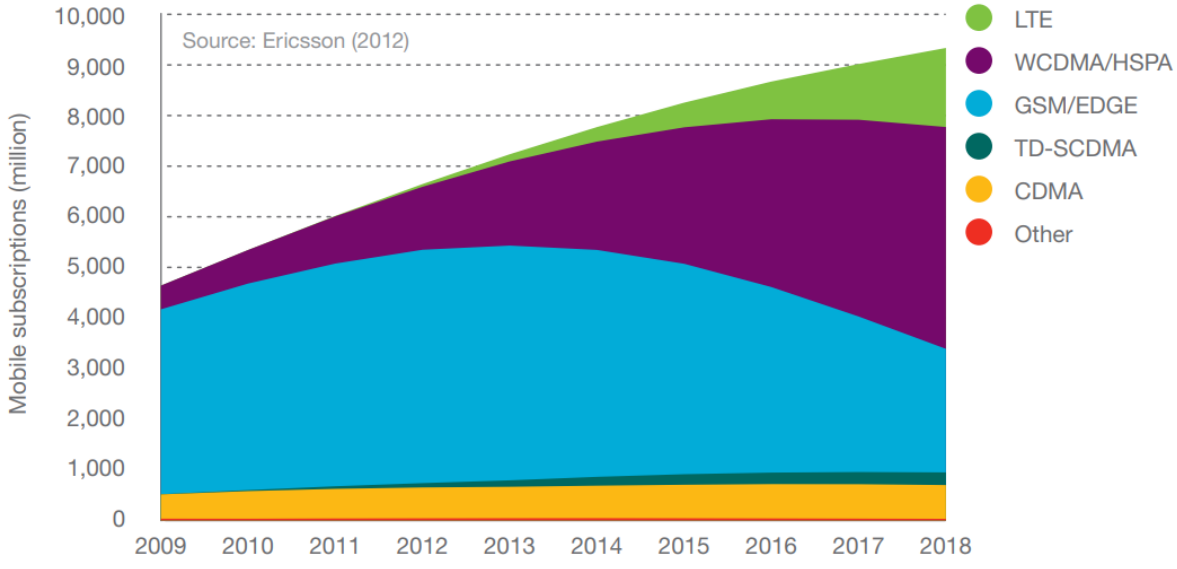

Figure 4. Mobile subscriptors by technology 2009-2018 by [8]

The 3GPP specifications for 4G are contained in Release 8 for LTE (Long Term Evolution), Release 9 for LTE enhancements, Release 10 for LTE-Advance, Release 11 for improved speed rates and Release 12 for addressing IoT (Internet of Things) special needs and continuing LTE-Advance improvement.

Regarding the core network, also known as EPC (Evolved Packet Core), there is a simplification in the general architecture as can be seen in Figure 5. There is a transition to a completely packet-switched network in which there are just a few components: the MME (Mobility Management Entity) for the control-plane and mobility purposes, the SGW or Serving-GW (Serving Gateway) which is in charge of routing traffic, signaling, some mobility functions and also uses GTP (GPRS Tunnelling Protocol) tunnel to forward packets to the external network, and the PDN-GW (Packet Data Network Gateway) that takes care of stablishing different data plane policies, e.g. NAT (Network Address Translation) or DPI (Deep Packet Inspection).

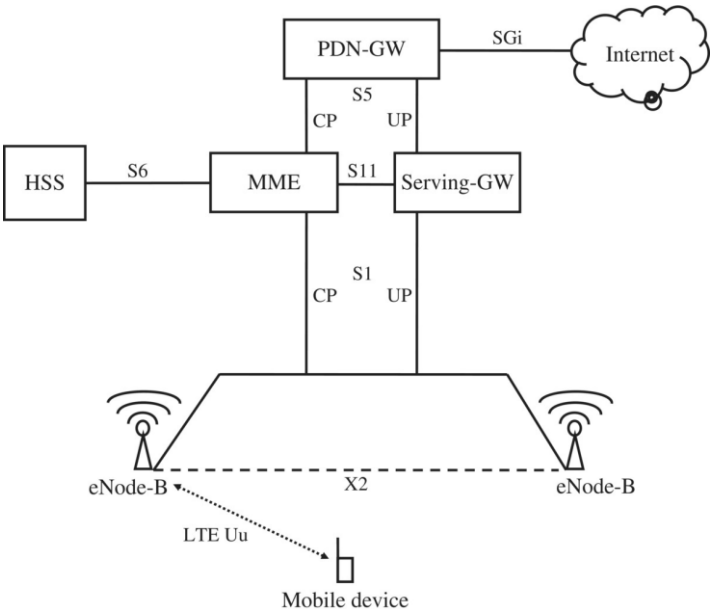

Figure 5. 4G LTE Network Architecture by [7]

1.1.2. 5G current context

Now that we have understood the historical context of this project, is the moment to move forward with the actual predictions. Ericsson last mobility report from June 2024 [9] estimates that mobile subscribers will be increased up to 9.3 billion by 2029 (related graph in Figure 6), this means that by that time more than 5 billion clients of the mobile networks will be 5G subscribers. Comparing the actual 1.5 billion 5G subscribers it means that 5G networks must support 4.5 billion subscribers more than now which makes it necessary to consider sustainability issues. In order to solve this problem, open virtualization technology for implementing 5G core is one of the technologies that can be helpful, and it is the primary motivation behind this project. By using it we could reduce the costs associated with virtualization to zero allowing new operators to start up with a lower initial investment which probably would enhance telecommunication market niche.

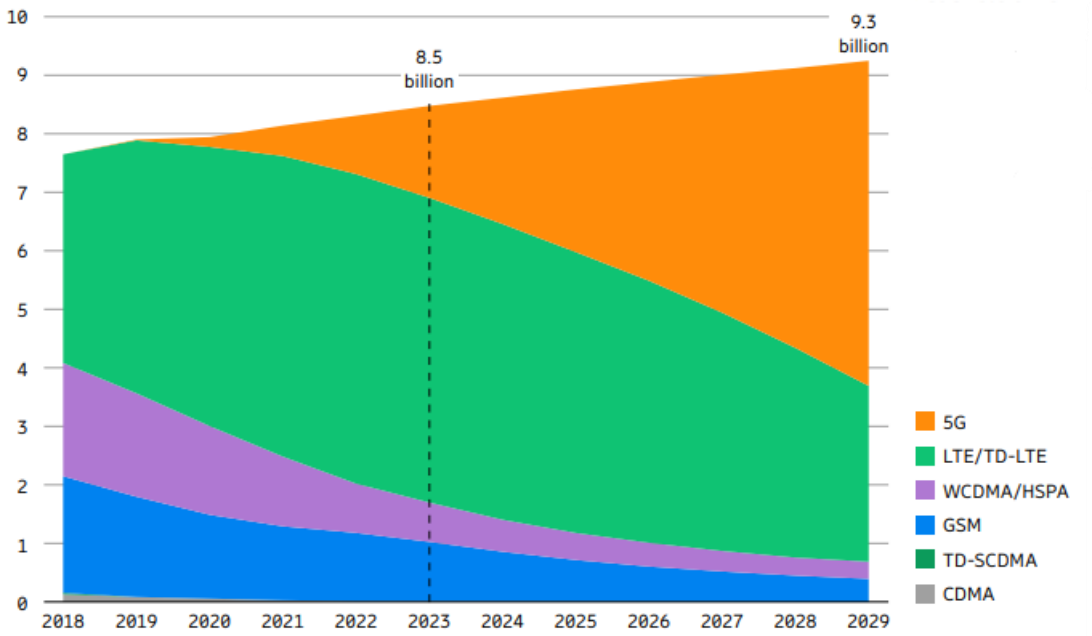

Figure 6. Mobile subscriptions by technology by [9]

Another challenge for 5G networks is to manage the increasing data density (see Figure 7). As per Ericsson report [9] claims, from first quarter 2023 to first quarter 2024 the mobile network data traffic has grown up to 25%. This is another reason that motivates this project, as it is necessary to scale up 5G core networks to adapt to data traffic demand fluctuation.

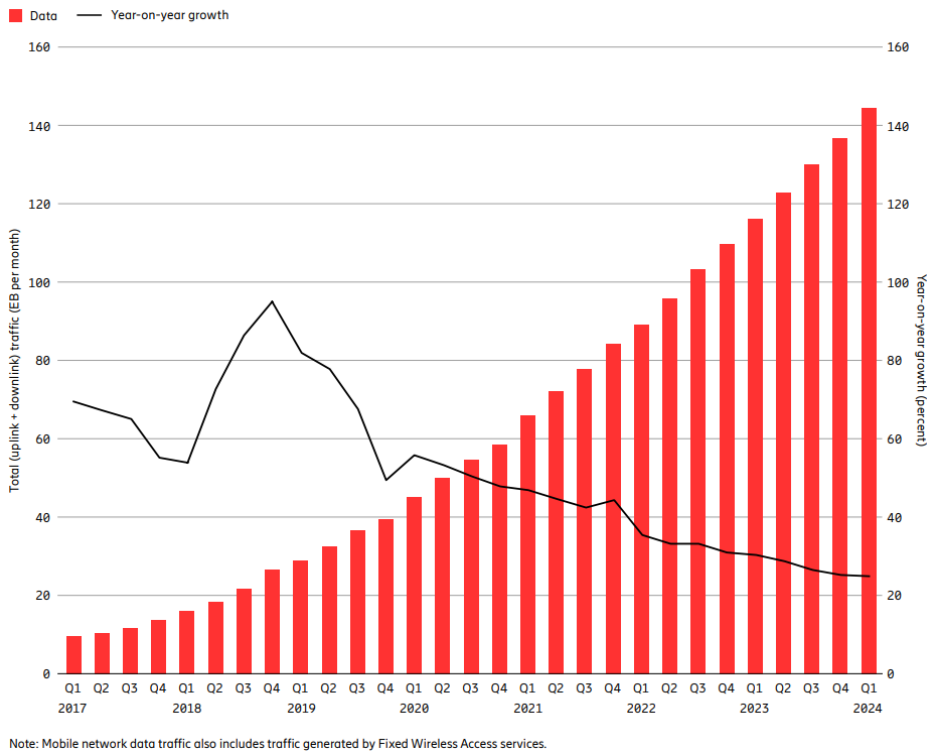

Figure 7. Global mobile network data traffic and year-on-year growth (EB per month) by [9]

In this project we will see how OpenStack cloud tools could address both issues making it easier to deploy 5G core networks as well as managing its resources adequately.

1.1.3. 5G Architecture

The 5G architecture is proposed by ETSI 3GPP in TS 123.501 [10] which is one of the Technical Specifications from Release 15. In this TS can be found the proposed general architecture for 5G which is composed of three main components (see Figure 8):

- UE (User Equipment): The Clients of the mobile network which can be a smartphone, laptop, tablet, or equivalent with 5G network compatibility.
- 5GC (5G Core): This normally belongs to the operator company that is in charge of giving the requested service to clients. This network is very complex and is structured as a set of cooperating NF (Network Functions) in order to simplify tasks.
- RAN (Radio Area Network): The network that is composed of the different gNB (gNodeB) which are the radio base stations in 5G.

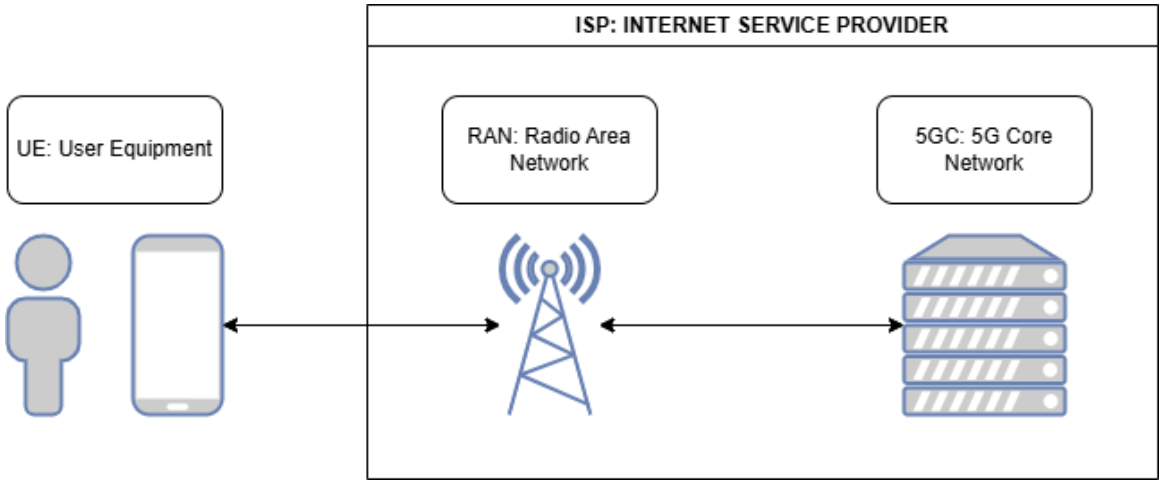

Figure 8. Simplified 5G architecture

In the 5G core architecture (see Figure 9), we can distinguish between NFs from control plane and data plane.

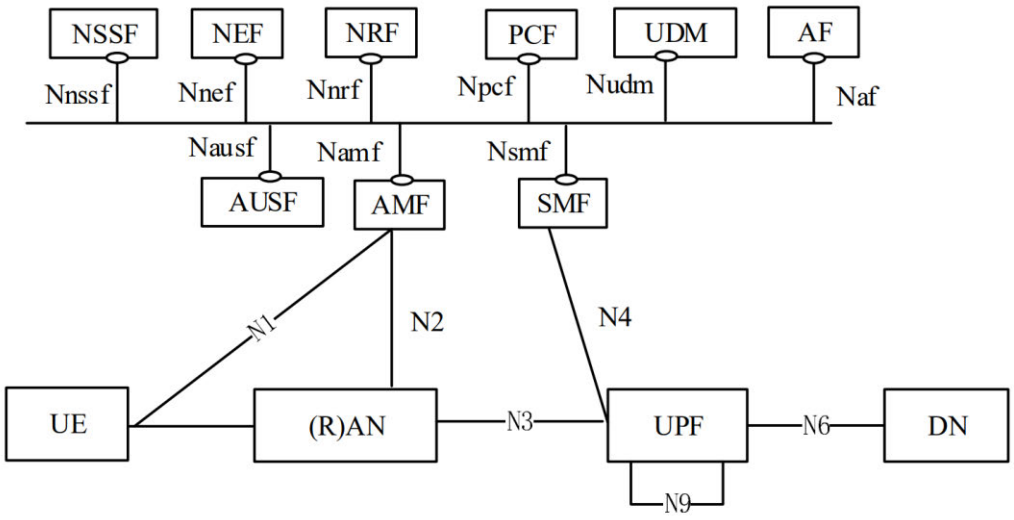

Figure 9. 5G architecture by [10]

In the data plane there are only a few NFs as for instance:

- **UPF** (User Plane Function): handles the user data from and to the data networks.
- **DN** (Data Network): this is not, strictly speaking, a 5G NF; it represents the internet or an external data network.
- **RAN** (Radio Area Network): as commented above is the radio access network, but in fact radio access is optional (other access technologies are allowed) and because of that "R" is between parentheses.
- **UE** (User Equipment)

In the control plane we have a lot more NFs:

- **AMF** (Access and Mobility Management Function): is the function that UEs and gNBs must contact to request and receive mobility and access services from the core.
- **SMF** (Session Management Function): oversees calls and sessions, as well as contacts the UPF accordingly.
- **AUSF** (Authentication Server Function): performs the authentication function of identifying UEs and storing authentication keys.
- **NRF** (NF Repository Function): serves as a repository for information about other network functions
- **UDM** (Unified Data Management): is responsible for processing network user data.
- **PCF** (Policy and Charging Function): is responsible for policy control (resource allocation, QoS (Quality of Service), service access policies) and charging (online and offline).
- **NSSF** (Network Slice Selection Function): provides the AMF with information about the location of network functions.
- **NEF** (Network Exposure Function): It allows external applications to access network functions and services
- **AF** (Application Function): provides specific application-related services like special routing policies for some applications.

1.2. Technical and academic objectives

From a technical standpoint, the objectives to be accomplished in this project are as follows:

- Establish an adaptable virtualization solution for a 5G network, with adaptability defined as the ability to modify the parameters of each machine individually.
- Establish a scalable solution, with scalability defined as the ability to add and remove virtual machines.
- Support virtual radio segment deployments.
- Provide a management system for the deployed network.
- Provide a machine backup system that allows for the redeployment or recovery of the network in case of data loss.

From an academic perspective, the student who develops this project acquires the following competencies and skills:

- Knowledge of different technologies:
  - Linux OS (Operating System)
  - Virtualization with KVM (Kernel-based Virtual Machine) and QEMU (Quick Emulator)
  - OpenStack cloud
  - Free5GC
  - UERANSIM
- Skills to design and manage resources in both physical and virtual servers, with resources meant as storage, memory, network and compute.
- Ability to design, deploy, configure and test virtualized networks to implement 5G core network.
- Competence in deploying cloud solutions and management of its resources.
- Ability to troubleshoot OS, application and networking errors.

1.3. Structure of this report

The rest of the document is organized as follows:

In chapter 2 of this document, it is offered information about the State of the art technologies used in this project, we will see an overview of the virtualization solutions used in this project as well as the 5GC and RAN technologies that will be implemented.

In chapter 3 we will define the design specifications and constraints which we will be considering for this solution.

To continue, in chapter 4, we will address the insights of this project by making a brief introduction and then describing the system architecture in more detail. After that, the deployment execution for all the components that shape this project will be performed.

In order to guarantee the system functionality, the results chapter will describe not only the necessary tests performed to check that the deployed components of the system are working as expected, but also comply with the technical specifications and restrictions established.

In the last part of this document, the detailed budget and impact of the project will be covered followed by the final conclusions and pending future works.

2. State of the art

2.1. Hypervisors

A hypervisor, also referred to as a VMM (Virtual Machine Monitor), is software designed to generate and operate VMs (Virtual Machines). The hypervisor enables a single host computer to accommodate multiple guest VMs by virtually allocating its resources, including memory and processing power. There are two types of hypervisors (depicted in Figure 10) depending on the way they work: Type 1 hypervisors (known as bare metal) and Type 2 hypervisors (known as hosted).

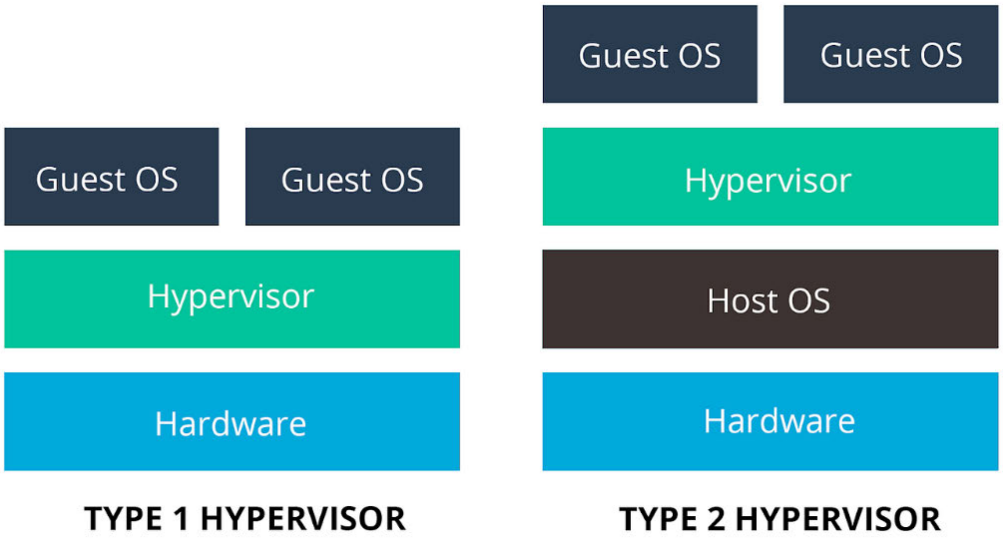

Figure 10. Hypervisors architecture from [11]

These hypervisors could be classified in three working modes depending on the technology behind the virtualization of physical resources and the way they handle system calls inside the VMs. These modes are described above:

1. **Full virtualization mode**: In Full Virtualization mode all the hardware is emulated as can be seen in Figure 12 This technique uses binary translation (see Figure 11) which introduce some delays as the VMs applications running in the guest OS must convert virtual resources into something that a program could run in physical hardware.

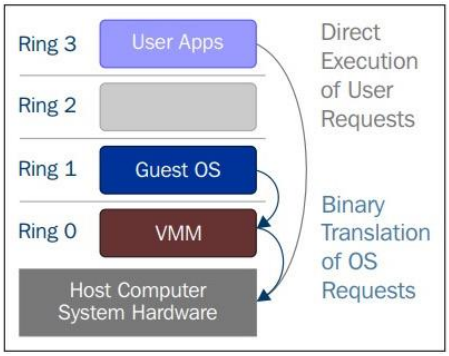

Figure 11. Binary translation from [11]

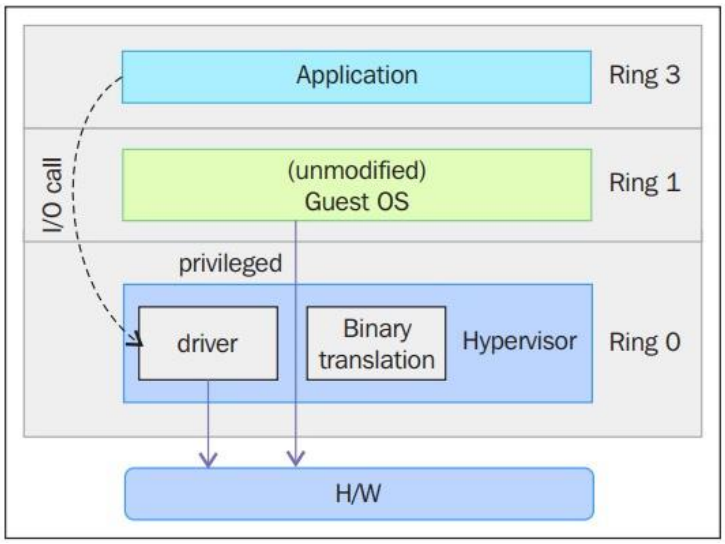

Figure 12. Full virtualization from [11]

2. **Paravirtualization mode**: In paravirtualization mode (see Figure 13) the guest OS needs to be altered to allow VMs instructions to access kernel layer (Ring 0). This is faster than Full Virtualization as the binary translation is not needed but, in any case, whether an app is accessing kernel layer, it may not be the best option.

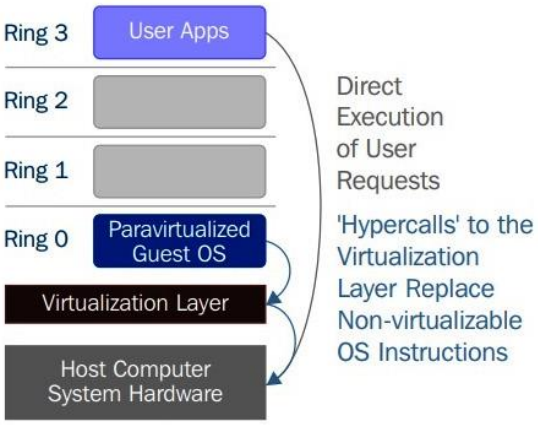

Figure 13. Paravirtualization from [11]

3. **Hardware-assisted mode**: Hardware-assisted mode virtualization is hard to implement in real systems, in Figure 14 the working scheme is located. Due to its complexity, AMD and Intel (the chipset companies) provide their own solutions for this type of virtualization, they are called Intel VT-x and AMD-V, respectively. Hardware-assisted virtualization not only introduces new instructions but also adds a new privileged access level, called ring -1, where the hypervisor can run.

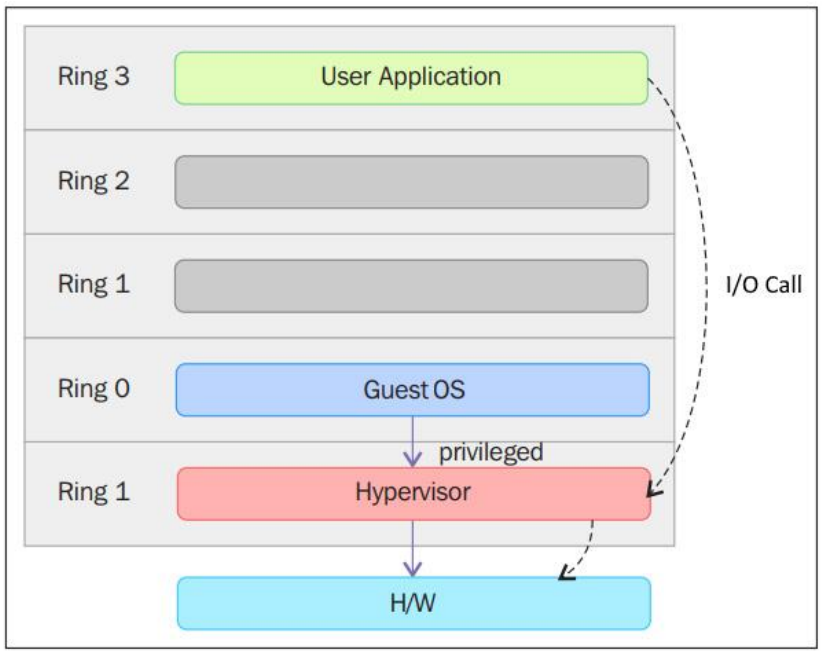

Figure 14. Hardware-assisted virtualization from [11]

The ASUS ESC4000-E10 server, which is the server that we are going to be using to host the virtualized functions of the 5G deployment, contains a processor that has enabled Intel VT-x by default, so that means that we will be using hardware-assisted virtualization as much as possible in this project.

2.1.1. Hypervisors type 1

Type 1 hypervisors, referred to as the bare metal hypervisor, run directly on the physical hardware of the host machine. The server Asus ESC4000-E10, which, as stated above, is the server that we will be using in this project, comes with KVM [12] preinstalled as default Type 1 Hypervisor.

In Table 1 there is a little comparison between the most used bare metal hypervisors serving as a reference to decide which is the best suitable for our project. As there is not so much difference between them the fastest way to start working in this project is with KVM so due to this it will be the hypervisor used in this project.

Table 1. Hypervisors type 1 comparison

HYPERVISOR TYPE 1	License type	Description

KVM	Open Source	KVM is a popular open-source bare-metal hypervisor that runs directly on the server hardware.
Proxmox	Open Source	Proxmox is another open-source bare-metal hypervisor that provides virtualization services.
Citrix (Xen)	Open Source	Xen is an open-source bare-metal hypervisor that allows for efficient and secure virtualization.
VMware ESXi	Commercial	VMware ESXi is a commercial bare-metal hypervisor known for its efficiency, security, and reliability.
Microsoft Hyper-V	Commercial	Microsoft Hyper-V is a free, standalone version of Microsoft's bare-metal hypervisor.

2.1.2. Hypervisors type 2

Type 2 hypervisors, also known as hosted, run as a normal application over the operating system. This leads to various performance issues such as for instance the speed of the VM. This is a serious problem when talking about 5G networks as it is preferred to avoid delays in the response time. In order to avoid issues regarding latency we will avoid this kind of hypervisors as much as possible. In case we need to, we will use QEMU [13] hypervisor as it supports hardware virtualization with KVM and is fully open source. The most used hosted hypervisors have been compared in Table 2 to give a global vision of the actual trade solutions.

Table 2. Hypervisors type 2 comparison

Hypervisor	Compatibility and Features	Performance and Use Cases	Licensing and Cost
Microsoft Virtual PC (Personal Computer)	Supports various Windows operating systems. Provides integration features for seamless interaction between host and guest OS.	Suitable for running legacy Windows applications and testing different versions of Windows.	Commercial use requires a license.
Oracle VirtualBox	Supports a wide range of host operating systems. Offers shared folders for VMs running on VMware and Oracle hypervisors.	Suitable for personal and educational use.	Free Open Source. Extension pack requires a license for commercial use.
VMware Workstation	Supports a wide range of operating systems, including Windows, Linux, and macOS. Offers powerful performance and 3D graphics support.	Suitable for building, testing, and demoing software.	VMware Workstation Pro requires a fee for business operations.
VMware Fusion	Designed specifically for macOS hosts. Offers full 3D graphics support.	Suitable for running Windows on Mac and for software development and testing.	Requires a licensing fee for macOS hosts.
QEMU	Can run on top of the operating system. Can utilize KVM for hardware-assisted virtualization.	Suitable for running virtual machines on Linux systems.	Open-source and free to use.
Oracle VM Server for x86	Provides a comprehensive and fully integrated stack of cloud applications and cloud platform services. Supports x86 architectures and a variety of workloads such as Linux, Windows, and Oracle Solaris. Offers efficient and optimized server virtualization.	Suitable for server virtualization and a variety of workloads including Linux, Windows, and Oracle Solaris.	Available through Oracle VM support agreements.

2.2. Virtual machine management: OpenStack

OpenStack is an open source cloud computing platform accepted by Linux Foundation for its massively scalability and configuring options. It supports more than ten different virtualization solutions including the following:

- KVM
- Xen (via libvirt)
- LXC (Linux containers)
- Microsoft Hyper-V
- VMware ESXi
- Citrix XenServer
- UML (User Mode Linux)
- PowerVM (IBM Power 5-9 platform)
- Virtuozzo
- z/VM (for IBM Z and IBM LinuxONE servers)

The architecture of OpenStack relies on multiple independent and open-source projects (depicted in **Error! Reference source not found.**) that by interacting with each other can provide a complete cloud environment solution. The following projects are part of the heart of OpenStack, as they are mandatory in all cloud environments released until now:

- **Keystone**: This service provides a single point of integration for managing user authentication.
- **Glance**: This service is in charge of managing image storage and metadata regarding these images.
- **Nova**: This service oversees all the computing resources available for VMs such as, for instance the Flavors (sizes of VMs) and Quotas (what resources a project and user has access to).
- **Neutron**: The network service of OpenStack, which provides a complete set of SDN tools to manage the cloud networks as for example fixing IP (Internet Protocol) addresses to VMs.
- **Horizon**: It is an optional dashboard service for OpenStack which provides a frontend to the other OpenStack services allowing users to launch VMs, configure networks, etc.
- **Cinder**: Another optional service for OpenStack that makes use of block storage devices for the VMs as well as adding additional space to a VM. The Block Storage service can be configured to use LVM or drivers from contributing companies to connect to their hardware.
- **Swift**: Is an optional Object Storage service that provides scalability and is optimized for high availability, durability, and concurrency across the data set.

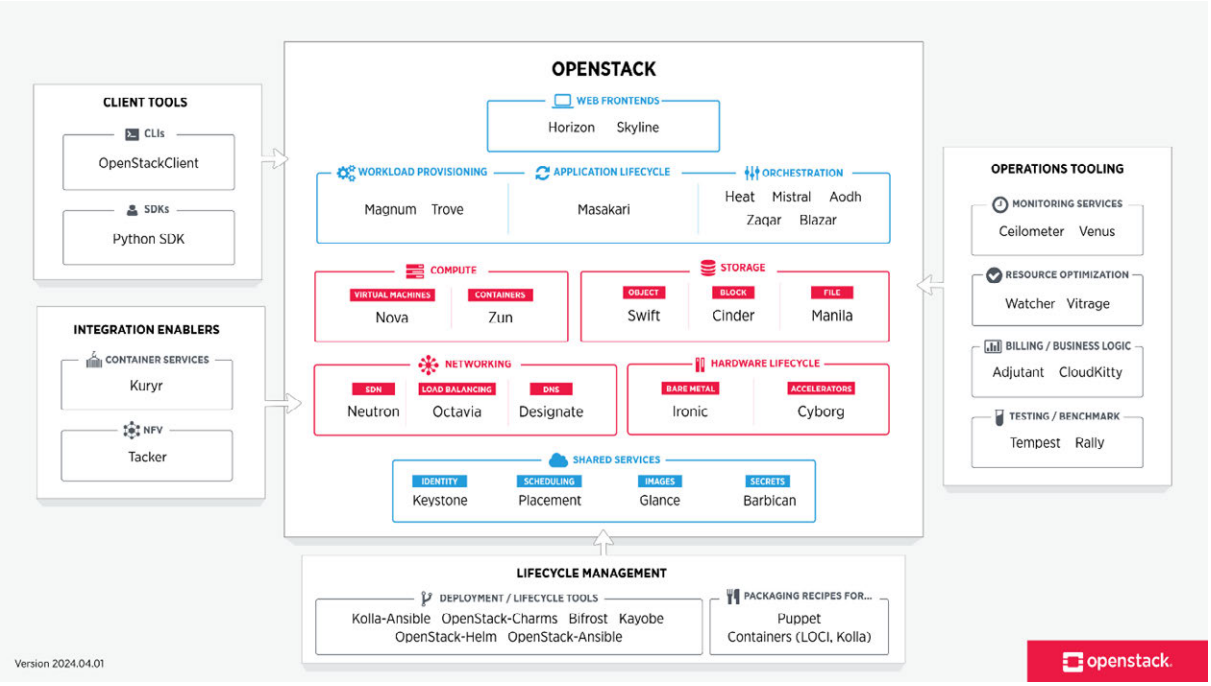

Figure 15.OpenStack projects and core functionalities by [14]

In this project, we will be using a tool named DevStack [15]. This tool is a set of scripts which allows a quick deployment of OpenStack and was designed to work in Ubuntu 22.04 LTS (Long-term support) (Jammy) OS, the one that will be using in ASUS ESC4000-E10 server. The services configured by DevStack, are the following: Identity (Keystone), Object Storage (Swift), Image Service (Glance), Block Storage (Cinder), Compute (Nova), Placement (Placement), Networking (Neutron) and Dashboard (horizon). The only one that is not mentioned above is Placement project which provides an HTTP (Hypertext Transfer Protocol) service for representing available resources in the cloud.

2.3. 5G solutions

There are lots of projects in 5G core networks, as for instance Project Magma [16], open5GS [17], Internship-5GCN [18], OAI-CN [19] among others. But the most widely used for academic and non-professional purposes are Free5GC and open5GS. In regard with the UEs and gNBs simulation there also open source projects like O-RAN [20] but in this project we will only overview the UERANSIM simulator.

2.3.1. Project Free5GC

The free5GC is an open-source project for 5G mobile core networks. The ultimate goal of this project is to implement the 5G core network (5GC) defined in 3GPP Release 15 (R15) and beyond.

The following reference contains a set of specifications and features supported by this tool [21]. As a summary, free5GC has a complete set of NFs (Network Functions) like:

- **AMF** (Access and Mobility Management Function): Registration Management, Connection Management, Reachability Management, Mobility Management, and Authentication.
- **SMF** (Session Management Function): Session Management, IP Assigning/Management.
- **UPF** (User Plane Function): Supports multiple UPF and ULCL (Uplink Classifier). SSC (Session and Service Continuity) mode 1. Packet Routing/Forwarding
- **CHF** (Charging Function): Online/Offline Charging supporting Flow-Based Charging on PDU (Packet Data Unit) Session.
- **AUSF** (Authentication Server Function): serves as a central entity that facilitates the authentication process between the UE and the network
- **NRF** (NF Repository Function): Generate authentication tokens for services.
- **UDM** (Unified Data Management): is responsible for managing information related to UE.
- **UDR** (Unified Data Repository): serves as a centralized data repository for subscription data, subscriber policy data, sessions, contexts, and application states.
- **PCF** (Policy and Charging Function): It contains data policies which must be requested in order to adapt PDU sessions to the services that have been acquired by a Subscriber.
- **NSSF** (Network Slice Selection Function): oversees management and orchestration of network slices which involves the creation of multiple virtual networks on a shared physical infrastructure.
- **N3IWF** (Non-3GPP Interworking Function): acts as a gateway for the 5GC network with support for N2 and N3 interfaces.
- **N3IWUE** (Non-3GPP Interworking User Equipment): creates GRE (Generic Routing Encapsulation) tunnels to transmit user data packets between the UE and the N3IWF. This NF is not active by default.

The Free5GC architecture by default is depicted in Figure 16.

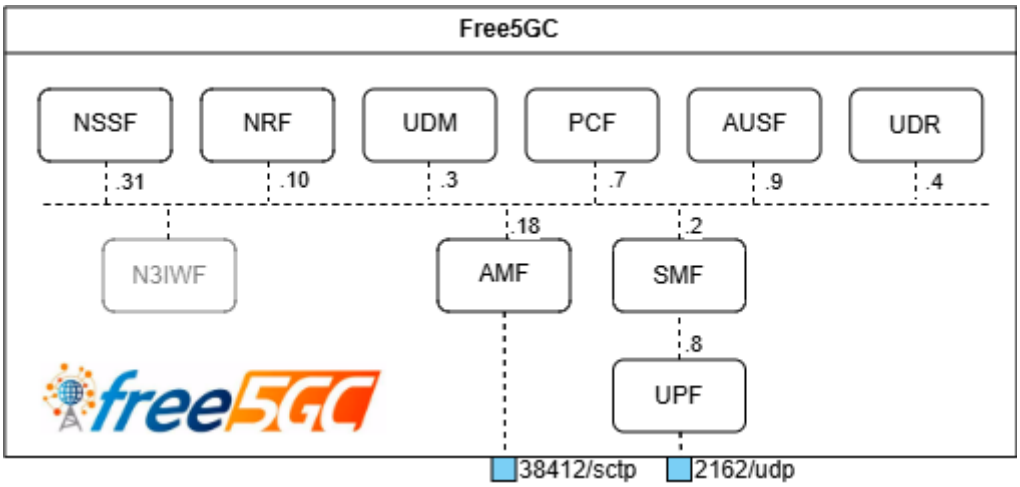

Figure 16. Free5GC default NF architecture scheme

2.3.2. Project UERANSIM

UERANSIM is an open source state-of-the-art 5G UE and RAN (gNB) simulator used for testing 5G Core Networks. It is compatible with different cores such as free5GC, open5GS and OAI-5GC. This project consists of a repository [22] in which there are a bunch of tools ready for testing and simulation.

In the control plane there are two interfaces: NAS (Non-Access Stratum) and NGAP (Next Generation Application Protocol).

- NAS: in control of UE with the following supported features:
  - Primary Authentication and Key Agreement
  - Security Mode Control
  - Identification
  - Generic UE Configuration Update
  - Initial and Periodic Registration
  - UE and Network initiated De-registration
  - UE initiated PDU session establishment
  - UE and Network initiated PDU session release
  - Service Request
  - Paging
- NGAP: in control of the gNBs with the following supported features:
  - PDU Session Resource Setup
  - PDU Session Resource Release
  - Initial Context Setup
  - UE Context Release (NG-RAN(Next Generation Radio Access Network) node initiated and AMF initiated)
  - UE Context Modification
  - Initial UE Message
  - Paging
  - Downlink NAS Transport
  - Uplink NAS Transport
  - NAS Non-Delivery Indication
  - Reroute NAS Request
  - NG Setup
  - Error Indication

In the user plane the RAN implements GTP protocol, but only IPv4 is supported currently.

UERANSIM is an open source state-of-the-art 5G UE and RAN (gNB) simulator used for testing 5G Core Networks. It is compatible with different cores such as free5GC, open5GS and OAI-5GC. This project consists of a repository [22] in which there are a bunch of tools ready for testing and simulation.

In the control plane there are two interfaces: NAS (Non-Access Stratum) and NGAP (Next Generation Application Protocol).

- NAS: in control of UE with the following supported features:
    - Primary Authentication and Key Agreement
    - Security Mode Control
    - Identification
    - Generic UE Configuration Update
    - Initial and Periodic Registration
    - UE and Network initiated De-registration
    - UE initiated PDU session establishment
    - UE and Network initiated PDU session release
    - Service Request
    - Paging
- NGAP: in control of the gNBs with the following supported features:
    - PDU Session Resource Setup
    - PDU Session Resource Release
    - Initial Context Setup
    - UE Context Release (NG-RAN(Next Generation Radio Access Network) node initiated and AMF initiated)
    - UE Context Modification
    - Initial UE Message
    - Paging
    - Downlink NAS Transport
    - Uplink NAS Transport
    - NAS Non-Delivery Indication
    - Reroute NAS Request
    - NG Setup
    - Error Indication

In the user plane the RAN implements GTP protocol, but only IPv4 is supported currently.
Specifications and design constraints

3. Specifications and design constraints

This chapter contains the specifications and design constraints regarding the deployment of the proposed solution.

3.1. Specifications

- An ASUS ESC4000-E10 server has to be used in this project.

UERANSIM is an open source state-of-the-art 5G UE and RAN (gNB) simulator used for testing 5G Core Networks. It is compatible with different cores such as free5GC, open5GS and OAI-5GC. This project consists of a repository [22] in which there are a bunch of tools ready for testing and simulation.

In the control plane there are two interfaces: NAS (Non-Access Stratum) and NGAP (Next Generation Application Protocol).

- NAS: in control of UE with the following supported features:
    - Primary Authentication and Key Agreement
    - Security Mode Control
    - Identification
    - Generic UE Configuration Update
    - Initial and Periodic Registration
    - UE and Network initiated De-registration
    - UE initiated PDU session establishment
    - UE and Network initiated PDU session release
    - Service Request
    - Paging
- NGAP: in control of the gNBs with the following supported features:
    - PDU Session Resource Setup
    - PDU Session Resource Release
    - Initial Context Setup
    - UE Context Release (NG-RAN(Next Generation Radio Access Network) node initiated and AMF initiated)
    - UE Context Modification
    - Initial UE Message
    - Paging
    - Downlink NAS Transport
    - Uplink NAS Transport
    - NAS Non-Delivery Indication
    - Reroute NAS Request
    - NG Setup
    - Error Indication

In the user plane the RAN implements GTP protocol, but only IPv4 is supported currently.
Specifications and design constraints

- Network design must be based on open-source tools which follow free software guidelines.
- The internal network of the laboratory where the server is located must have access to ASUS server in order to use and configure the 5G network remotely from internal clients (those attached to the laboratory network) or external clients through the internet.
- The virtualization solution must be implemented in ASUS ESC4000-E10 server

UERANSIM is an open source state-of-the-art 5G UE and RAN (gNB) simulator used for testing 5G Core Networks. It is compatible with different cores such as free5GC, open5GS and OAI-5GC. This project consists of a repository [22] in which there are a bunch of tools ready for testing and simulation.

In the control plane there are two interfaces: NAS (Non-Access Stratum) and NGAP (Next Generation Application Protocol).

- NAS: in control of UE with the following supported features:
    - Primary Authentication and Key Agreement
    - Security Mode Control
    - Identification
    - Generic UE Configuration Update
    - Initial and Periodic Registration
    - UE and Network initiated De-registration
    - UE initiated PDU session establishment
    - UE and Network initiated PDU session release
    - Service Request
    - Paging
- NGAP: in control of the gNBs with the following supported features:
    - PDU Session Resource Setup
    - PDU Session Resource Release
    - Initial Context Setup
    - UE Context Release (NG-RAN(Next Generation Radio Access Network) node initiated and AMF initiated)
    - UE Context Modification
    - Initial UE Message
    - Paging
    - Downlink NAS Transport
    - Uplink NAS Transport
    - NAS Non-Delivery Indication
    - Reroute NAS Request
    - NG Setup
    - Error Indication

In the user plane the RAN implements GTP protocol, but only IPv4 is supported currently.
Specifications and design constraints

- The virtualization solution must allow creation, modification and deletion of VMs as well as to resize VMs.
- To deploy and configure VMs a web or other type of GUI (Graphical User Interface) must be provided.
- The solution implementation must use free software.
- Virtualized 5G core network, UEs and gNBs must be deployed in different VMs to simulate a real deployment environment.

UERANSIM is an open source state-of-the-art 5G UE and RAN (gNB) simulator used for testing 5G Core Networks. It is compatible with different cores such as free5GC, open5GS and OAI-5GC. This project consists of a repository [22] in which there are a bunch of tools ready for testing and simulation.

In the control plane there are two interfaces: NAS (Non-Access Stratum) and NGAP (Next Generation Application Protocol).

- NAS: in control of UE with the following supported features:
    - Primary Authentication and Key Agreement
    - Security Mode Control
    - Identification
    - Generic UE Configuration Update
    - Initial and Periodic Registration
    - UE and Network initiated De-registration
    - UE initiated PDU session establishment
    - UE and Network initiated PDU session release
    - Service Request
    - Paging
- NGAP: in control of the gNBs with the following supported features:
    - PDU Session Resource Setup
    - PDU Session Resource Release
    - Initial Context Setup
    - UE Context Release (NG-RAN(Next Generation Radio Access Network) node initiated and AMF initiated)
    - UE Context Modification
    - Initial UE Message
    - Paging
    - Downlink NAS Transport
    - Uplink NAS Transport
    - NAS Non-Delivery Indication
    - Reroute NAS Request
    - NG Setup
    - Error Indication

In the user plane the RAN implements GTP protocol, but only IPv4 is supported currently.
Specifications and design constraints

- The virtualized core network must follow 3GPP Release 15 specifications.

3.2. Design constraints

- The design of 5G core must be in accordance with what is specified in Release 15 specification.
- 5GC UEs and gNBs must be in separate Virtual Machines.
- The non-virtualized network design must be adapted to UPM security policies.

UERANSIM is an open source state-of-the-art 5G UE and RAN (gNB) simulator used for testing 5G Core Networks. It is compatible with different cores such as free5GC, open5GS and OAI-5GC. This project consists of a repository [22] in which there are a bunch of tools ready for testing and simulation.

In the control plane there are two interfaces: NAS (Non-Access Stratum) and NGAP (Next Generation Application Protocol).

- NAS: in control of UE with the following supported features:
    - Primary Authentication and Key Agreement
    - Security Mode Control
    - Identification
    - Generic UE Configuration Update
    - Initial and Periodic Registration
    - UE and Network initiated De-registration
    - UE initiated PDU session establishment
    - UE and Network initiated PDU session release
    - Service Request
    - Paging
- NGAP: in control of the gNBs with the following supported features:
    - PDU Session Resource Setup
    - PDU Session Resource Release
    - Initial Context Setup
    - UE Context Release (NG-RAN(Next Generation Radio Access Network) node initiated and AMF initiated)
    - UE Context Modification
    - Initial UE Message
    - Paging
    - Downlink NAS Transport
    - Uplink NAS Transport
    - NAS Non-Delivery Indication
    - Reroute NAS Request
    - NG Setup
    - Error Indication

In the user plane the RAN implements GTP protocol, but only IPv4 is supported currently.
Specifications and design constraints

- ASUS ESC4000-E10 must use non-proprietary OS (i.e, Windows, MacOS)

4. Description of proposed solution

4.1. Introduction

In this chapter, we will describe the proposed solution with the architectural schemes that have been used and then we will see in detail how the solution has been deployed and configured, referring to both the physical, the OpenStack and the 5G networks respectively.

4.2. General description

4.2.1. System introduction and limitations

In order to start the design process of the different components of our project, we will overview the different design constraints regarding this solution.

- DevStack supports the two latest LTS releases of Ubuntu, Rocky Linux 9 and openEuler but Ubuntu 22.04 (Jammy) is recommended as the preferred supported OS.
- Free5GC requires the following modules/packets:
    - Go 1.21.8
    - Linux kernel (5.0.0-23-generic or 5.4.x) for using UPF
    - Any version of MongoDB (versions equal or upper than 5.0 requires CPU (Central processing unit) with AVX (Advanced Vector Extensions) support)
    - gcc
    - g++
    - cmake
    - autoconf
    - libtool
    - pkg-config
    - libmnl-dev
    - libyaml-dev
- OS memory of 2G or more is recommended for using WebConsole (the Free5GC web GUI)
- UERANSIM requires Ubuntu 16.04 or later and the following packages:
    - CMake 3.17 or later
    - gcc 9.0.0 or later
    - g++ 9.0.0 or later
- CHF NF is only supported in free5GC release v3.3.1.
- ASUS ESC4000-E10 requires OS installation and is limited by its technical specifications contained in [23].

In this solution, it has been agreed to follow the next design rules in order to accomplish the specifications described in chapter 3:

- ASUS ESC4000-E10 will be using a non-propietary Ubuntu OS.

- OpenStack cloud will be deployed in ASUS ESC4000 E10 server via DevStack.
- To implement 5G SA (Stand Alone) network architecture, Free5GC and UERANSIM tools will be used.
- Free5GC will implement the 5GC, and UERANSIM will simulate the RAN (UE and gNBs).
- UEs must be registered in the network by using WebConsole.
- Unless OpenStack documentation makes use of some CLIs like Neutron or Glance API, to avoid issues, the OpenStack CLI must be used with the only purpose but troubleshooting, as it is preferred to use Horizon web GUI.

4.2.2. Architecture of physical network

The proposed solution as shown in Figure 17 is composed of a server (the ASUS ESC4000-E10) and a physical client that can be accessed via Windows Remote Desktop app [24] through UPMvpn (Technical University of Madrid VPN (Virtual Private Network)) client. ASUS server contains three physical network interfaces, one for Direct Management (DM) and the rest for general purposes (LINK LOCAL 1 and LINK LOCAL 2). The LOCAL 1 physical interface is connected to 10.48.0.0/15 subnet which is the one that provides internet access to the server via LAB. GATEWAY.

In the PHYSICAL CLIENT there are only two physical interfaces, the ETHERNET 2 serves to access ASUS dashboard control center (consult page 69) and ETHERNET 1 serves to connect via ssh to ASUS server and to provide internet access to the machine.

In a general aspect, this configuration provides secure enough remote access, allowing to work with ASUS server and PHYSICAL CLIENT from any place with the only requirement of a good internet connection and the credentials for VPN and Remote Desktop.

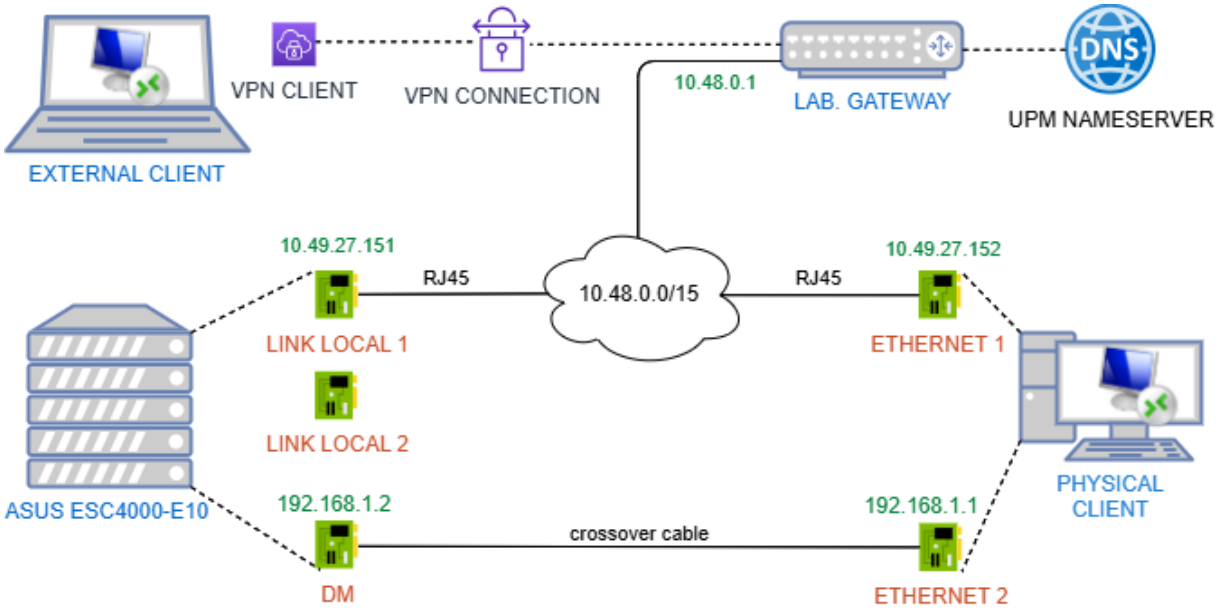

Figure 17. General physical network architecture scheme

The PHYSICAL CLIENT comes with Windows 10 by default, in this machine we will be using MobaXTerm [25] app as default SSH (Secure Shell Protocol) client, and also Microsoft Edge internet browser.

The ASUS ESC4000-E10 server had no preinstalled OS. The decision to configure it with Ubuntu Server 22.04.4 LTS (Jammy Jellyfish) was made in regard to the terms of compatibility of DevStack tool, that is the most tested version among Ubuntu distros. In the initial configuration (see Annex 2) OpenSSH version 3.0.2 has been installed as well as the OpenStack cloud platform following the scheme in Figure 18

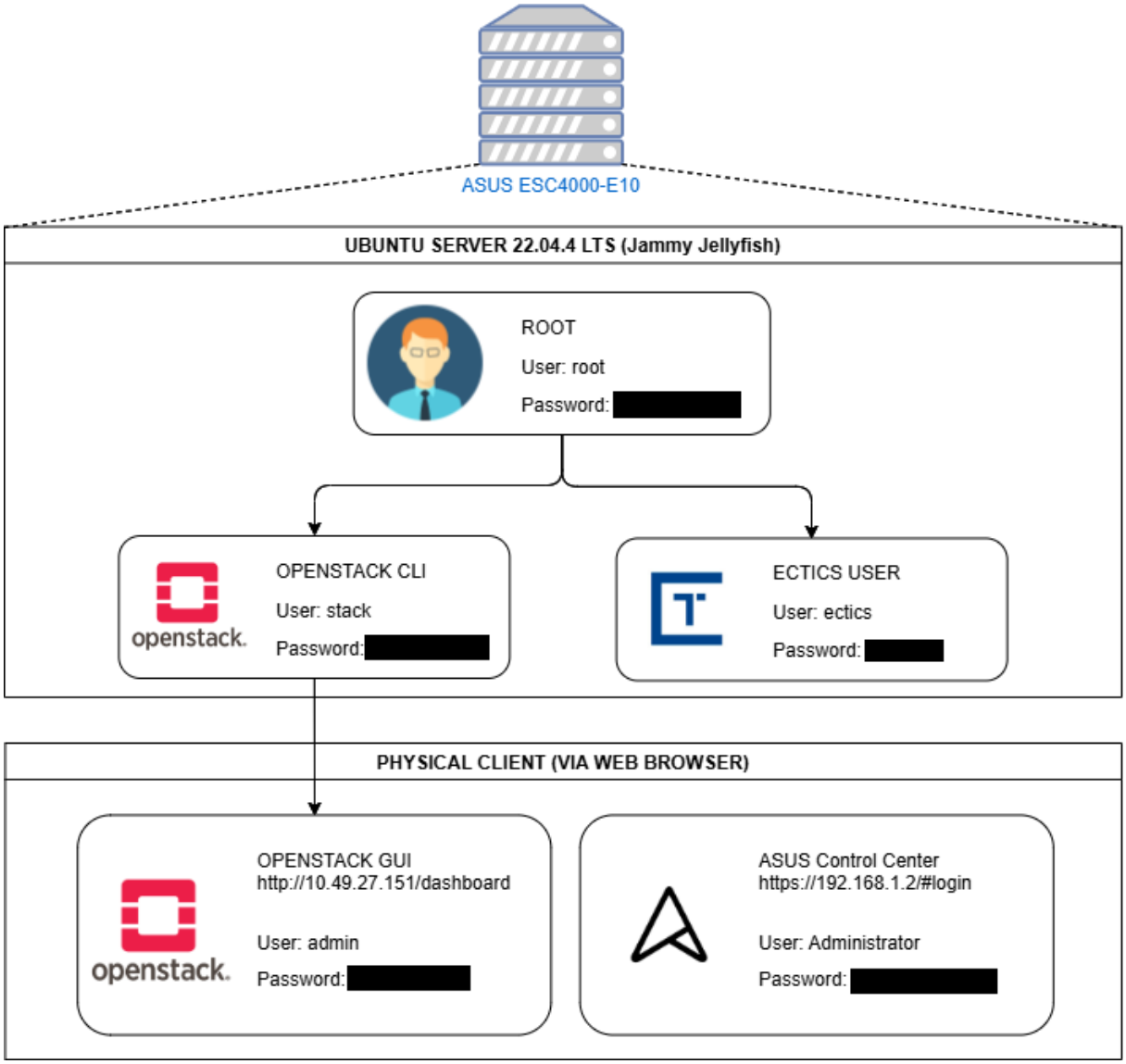

Figure 18. Users and passwords diagram

4.2.3. Architecture of OpenStack

OpenStack, a powerful tool used by companies, has a lot of configuration options in regard with security access, and project management. Because of that the proposed solution uses

these tools to provide an independent solution among other future projects that can be implemented.

In general, OpenStack defines the following tools for project management:

- **Domains**: they act as high-level containers for projects, users and groups. They can be used to centrally manage all Keystone-based identity components. Within domains, server, storage and other resources can be logically grouped into multiple projects which can themselves be grouped under a master account-like container. In addition, multiple users can be managed within an account domain and assigned roles that vary for each project.
- **Projects**: a project is a group of zero or more users. In Compute, a project owns virtual machines. In Object Storage, a project owns containers. Users can be associated with more than one project. Each project and user pairing can have a role associated with it.
- **Group**: entity which provides the project manager with the ability to assign one or more users as one independent system. This entity could be assigned to projects so that all the users of that group can be assigned to one project without the need to assign them individually.
- **User**: Entity which represents a person, a service or another that can enter the system with some individual credentials.
- **Role**: The roles are special rules that can be assigned to users or groups to give or subtract specific permissions. As default, admins and nonadmins are set.

In this project, we will configure these tools using Figure 19 as a reference.

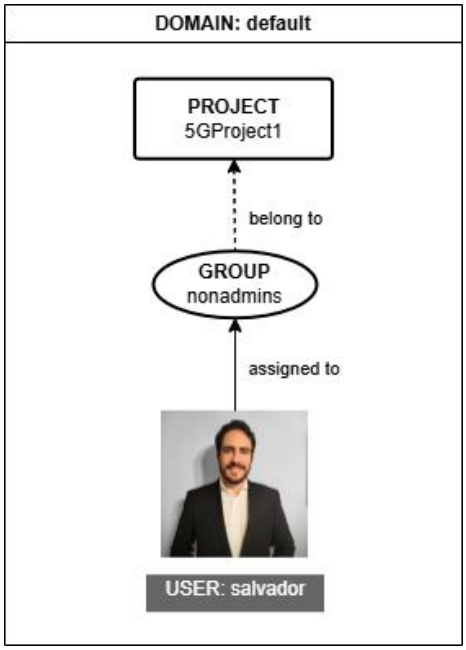

Figure 19. Openstack configuration: Project architecture

OpenStack provides SDN tools to allow users to configure virtual network elements for the created instances. Instances are just VMs running inside the host machine, in this case the

ASUS ESC4000-E10. In this solution we will be using three networks and one router as shown in Figure 20:

- **Public**: This network provides a centralized connection point with the external network through an external gateway. This external gateway in our case is the br-ex interface which must be configured with IP 172.24.4.1.
- **Private**: All the instances have virtual network interfaces connected to this network (10.0.0.0/26), these interfaces have IPs that are not reachable by default. For this reason, there is NAT (Network Address Translation) which allows us to give secure access to instances by assigning a Floating IP which is a Public Address associated with the instance private IP.
- **Shared**: Is another private network which is not used in our deployment but is created by Devstack initially.
- **R1**: In charge of routing the private network dataflow to the external networks to ASUS br-ex interface.

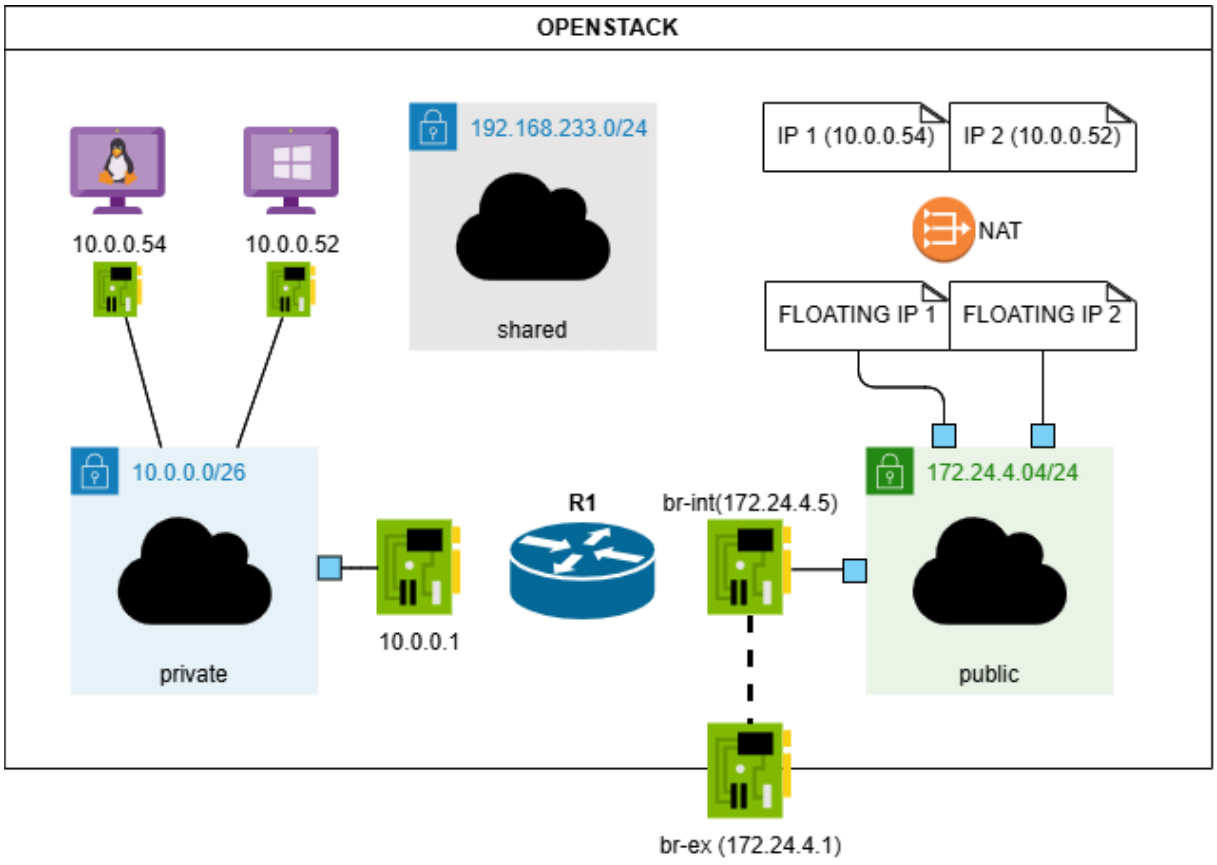

Figure 20. OpenStack configuration: Network architecture

Once the network is configured, it is necessary to prepare the following resources for the OpenStack instances to work properly (see Figure 21).

- Images: File which contains a virtual disk with bootable OS installed on it. This project will use ready to use Ubuntu Cloud images in .img format which are compatible with amd64 processor architecture.

- Flavor: Is a set of resources such as compute, memory, and storage capacity that can be assigned to an instance.
- Network: Instances can be assigned to one or more networks depending on availability.
- SSH-keys: to access an instance from the external network is necessary to generate SSH key pairs. This could grant secure access for that instance.
- Security Group: is a set of networking rules that accept or deny some traffic, as for instance, the SSH, ICMP and HTTP protocols. It is important to set those rules correctly to provide DNS (Domain Name System), ICMP (Internet Control Message Protocol) and SSH connectivity to the instances.
- Boot: Is referred to boot source which can be an Image or a volume. We need to specify one to launch instances.
- Floating IPs: We need to assign one if we need to connect the instance with the external network.
- Volume: is a logical item that is associated with real hardware, this mechanism provides persistence to instances. Whether snapshots are associated with that volume, the volume cannot be removed. There are limitations in this regard as each project has different quotas, which are a set of resources that can be assigned to a project.

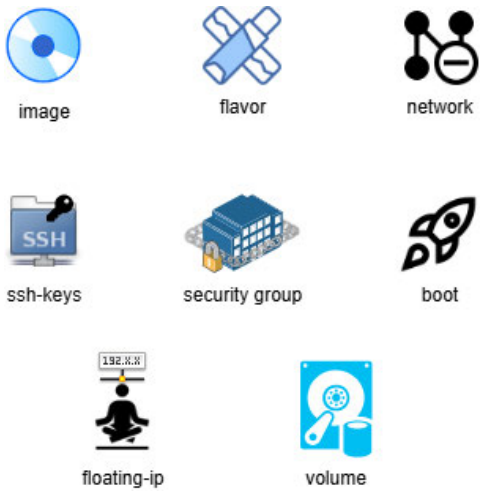

Figure 21. OpenStack configuration: instance requirements

4.2.4. Architecture of 5G System

In the OpenStack cloud, we will implement the following VMs to configure the 5G SA network solution following the architecture shown in Figure 22:

- **Free5GC**: This machine will contain the 5GC with all the elements running except for N3IWF as it is not necessary to run this NF in our case.
- **UE1**: Network client that will be connected to gNB1.
- **UE2**: Network client that will be connected to gNB1.
- **UE3**: Network client that will be connected to gNB2.
- **UE4**: Network client that will be connected to gNB2.

All the UEs have associated the corresponding IMSI (International mobile subscriber identity) to identify them as 5G net subscribers. This correspondence must follow the stablished in Table 3.

Table 3. UE - IMSI mapping

	IMSI
UE1	208930000000001
UE2	208930000000002
UE3	208930000000003
UE4	208930000000004

- **gNB1**: This machine will establish connection with AMF and UPF and will establish a tunnel connection with the UE1 and UE2 only.
- **gNB2**: This machine will have the same connection as gNB1 with AMF and UPF, but they will cover the connection only with UE3 and UE4 (also via a tunnel).

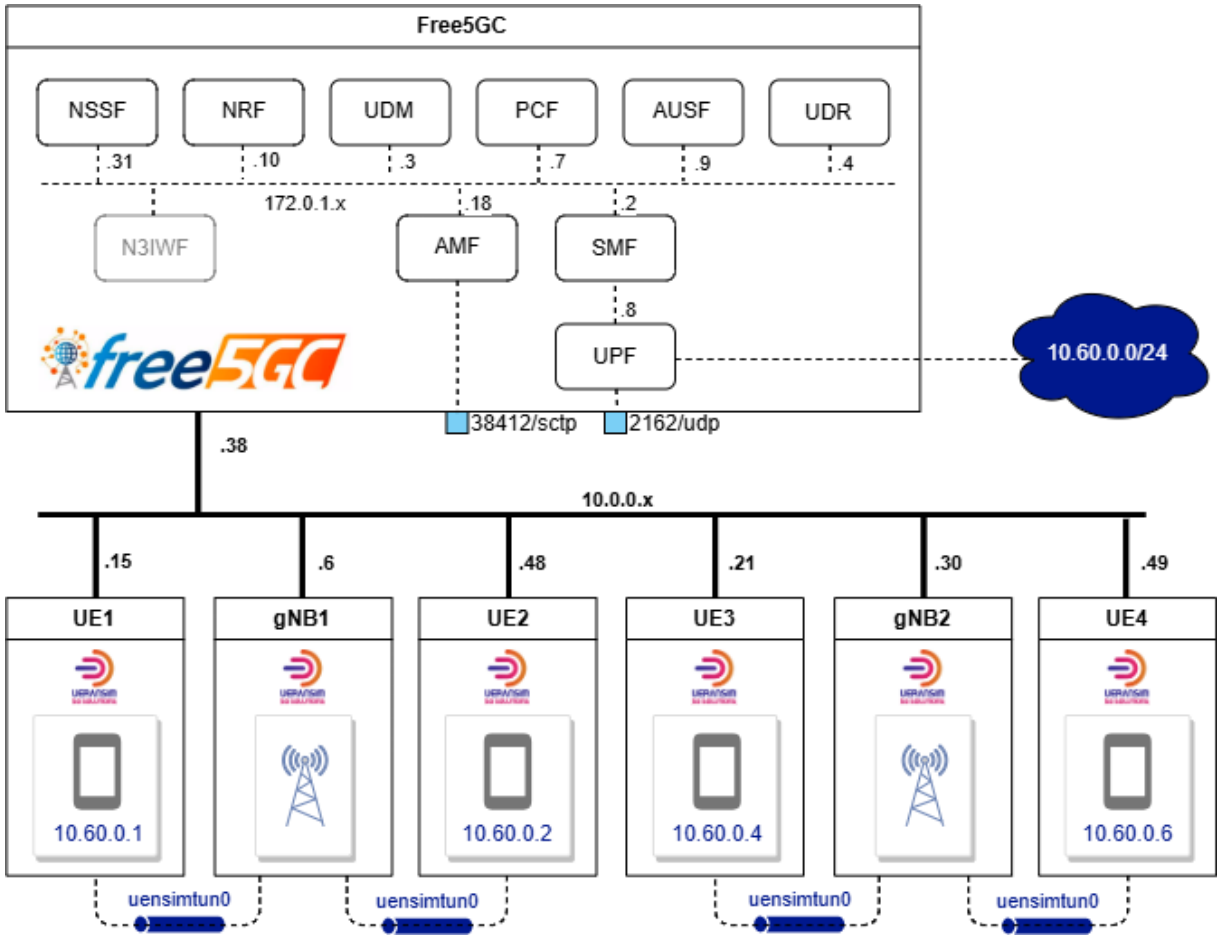

Figure 22. 5G solution: VM architecture

This architecture will implement a whole functional 5G System solution which allows to modify individually parameters inside every component of the network (including NFs, UEs and gNBs) and offering a complete configurable infrastructure. Not to mention that virtual machines can

be created, modified, deleted and resized within the OpenStack GUI with just a few clicks, offering quick scalability and modularity. In addition, this solution allows us to make Snapshots for every VM in order to restore it easily if something goes wrong, providing a simple backup system for the entire 5G System.

4.3. Deployment of physical network

Once the physical connections have been established between all nodes of the network as shown in Figure 17, it is necessary to configure the ASUS ESC4000-E10 OS (explained in Annex 2) and the corresponding network settings. This is made through the modification of the netplan configuration file by using the following command:

$ sudo nano /etc/netplan/00-installer-config.yaml

The file should be configured as the one shown in Figure 23.

```
stack@ectics-server:~$ sudo cat /etc/netplan/00-installer-config.yaml
# This is the network config written by 'subiquity'
network:
  ethernets:
    enp1s0f0:
      addresses:
      - 10.49.27.151/15
      nameservers:
        addresses:
        - UPM NAME SERVER IP
        search: []
      routes:
      - to: default
        via: 10.48.0.1
    enp1s0f1:
      dhcp4: true
    br-ex:
      addresses:
      - 172.24.4.1/24
      nameservers:
        addresses:
        - UPM NAME SERVER IP
        search: []
  version: 2
```

Figure 23. Netplan configuration file in ASUS server

After setting the configuration file, it is necessary to apply the changes by using the command:

$ sudo netplan apply

We have reached the end of ASUS ESC4000-E10 configuration. Moving to the PHYSICAL CLIENT, it is needed to set the UPM NAME SERVER proxy (to provide the machine with internet connection) and the RDP (Remote Destock Protocol) connection to provide external secure access, but this is out of the scope of this project.

4.4. Deployment of OpenStack

4.4.1. DevStack Openstack installation

Now it is necessary to follow DevStack documentation (see [15]) in order to set up the prerequisites of OpenStack. The first step is to add the user stack, to do this type the command:

$ sudo useradd -s /bin/bash -d /opt/stack -m stack

It is important that the home directory for the stack user has executable permission for all, as in Ubuntu 21.04 or later the "750" permissions given by default could cause issues during deployment.

```
$ sudo chmod +x /opt/stack
```

As this user will be making many changes to the system it should be assured that sudo privileges are set:

```
$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
$ sudo -u stack -i
```

The next step is to download the DevStack project from git repository by using:

```
$ git clone https://opendev.org/openstack/devstack
```

We need to create or retrieve a file named local.conf inside DevStack folder as it is required to set some sort of configuration for the OpenStack deployment. To do this, type the following commands and write the needed configuration as shown in Figure 24:

```
$ cd devstack
$ cp samples/local.conf local.conf
$ sudo nano local.conf
```

```
# 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]]

# Minimal Contents
# ----------------

# While ``stack.sh`` is happy to run without ``localrc``, devlife is better when
# there are a few minimal variables set:

# If the ``*_PASSWORD`` variables are not set here you will be prompted to enter
# values for them by ``stack.sh``and they will be added to ``local.conf``.
ADMIN_PASSWORD=
DATABASE_PASSWORD=
RABBIT_PASSWORD=
SERVICE_PASSWORD=$ADMIN_PASSWORD

# ``HOST_IP`` and ``HOST_IPV6`` should be set manually for best results if
# the NIC configuration of the host is unusual, i.e. ``eth1`` has the default
# route but ``eth0`` is the public interface.  They are auto-detected in
# ``stack.sh`` but often is indeterminate on later runs due to the IP moving
# from an Ethernet interface to a bridge on the host. Setting it here also
# makes it available for ``openrc`` to include when setting ``OS_AUTH_URL``.
# Neither is set by default.
HOST_IP=10.49.27.151
#HOST_IPV6=2001:db8::7


# Logging
# -------

# By default ``stack.sh`` output only goes to the terminal where it runs.  It can
# be configured to additionally log to a file by setting ``LOGFILE`` to the full
# path of the destination log file.  A timestamp will be appended to the given name.
LOGFILE=$DEST/logs/stack.sh.log
```

Figure 24. Main Configuration of DevStack local.conf

To start the deployment of OpenStack there is a bash script that can be executed by the command:

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

This execution may take between 15 and 30 minutes to complete depending on the internet connection and server available resources. Once the deployment has finished the machine standard output should be something similar to the one in Figure 25.

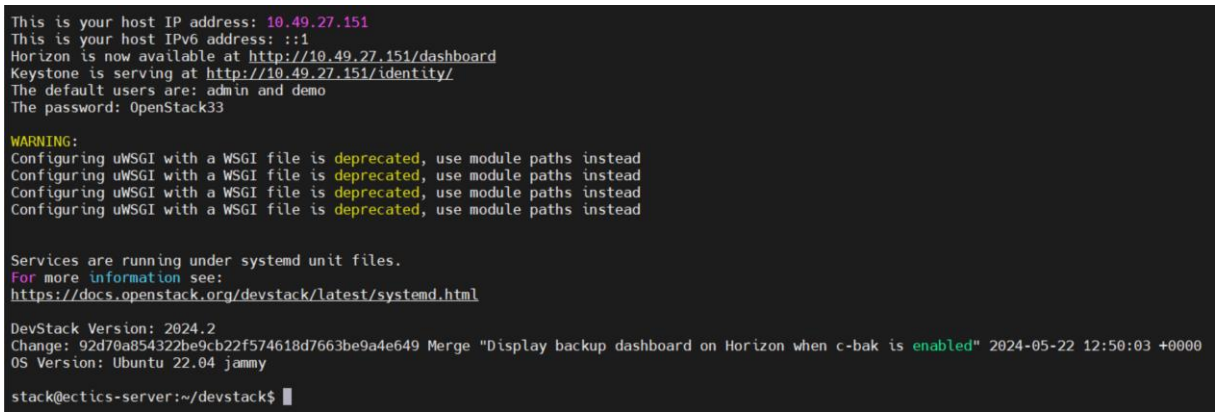

Figure 25. Successful run of stack.sh script in ASUS server

4.4.2. OpenStack deployment: Users and project configurations

With all that has been done, it is possible to enter the Horizon web GUI dashboard from the PHYSICAL CLIENT as shown in Annex 3 (see Figure 86). Once the credentials have been introduced, we can proceed with the initial configuration of the user deployment shown in Figure 19.

In the panel over the left find the Identity > Domains button and click it. In this window (see Figure 26) nothing is going to be set but notice that you could not create domains from the GUI interface. To configure domains, it is necessary to use OpenStack CLI (consult OpenStack CLI documentation [26] and [27]).

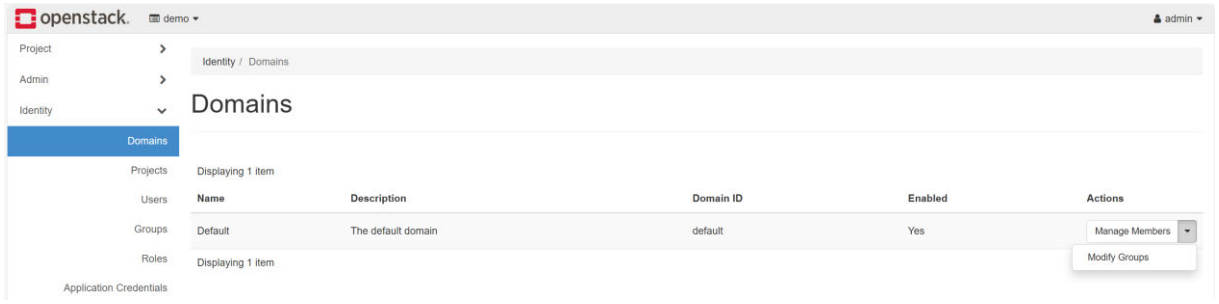

Figure 26. OpenStack deployment: Domains

First it is needed to define the roles of our project, this can be done by clicking Identity > Roles button. This will show the following window (Figure 27) in which the roles are defined. We will be using the default configuration.

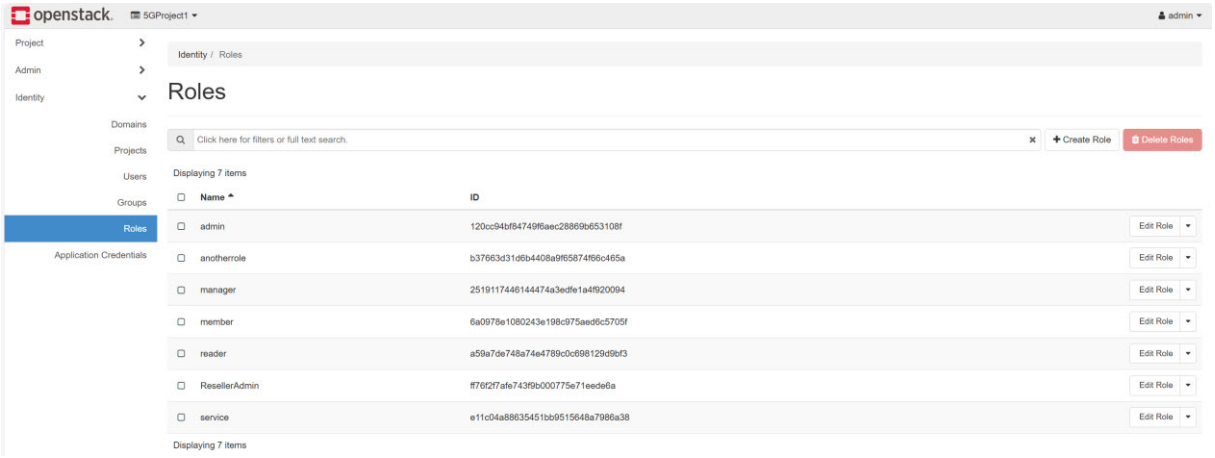

Figure 27. OpenStack deployment: Roles

Roles are nothing without a user that could perform that function. Because of that it is necessary to create a user, for that, go to Identity > User and look for the Create User button and by just typing the name, email, role and password and clicking on create user it is enough. An example for the user Salvador is shown in Figure 28. It is important to remark that there are some users already created by DevStack such as the admin user.

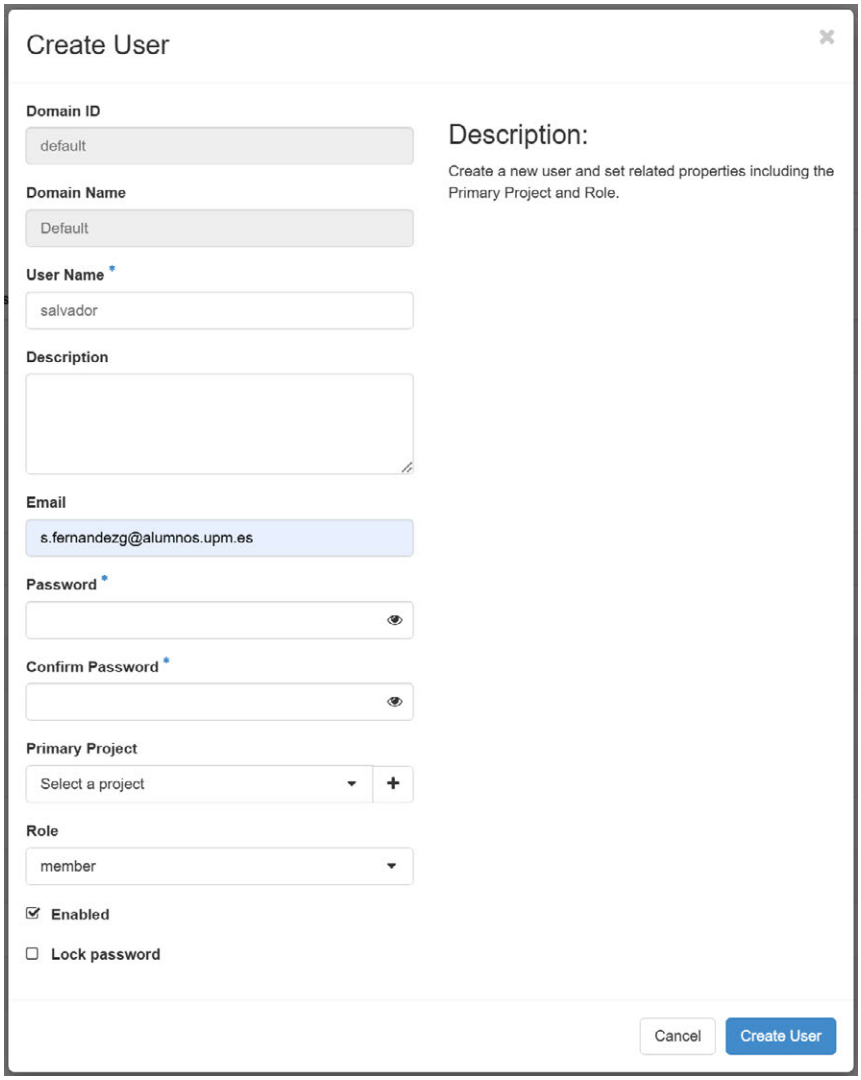

Figure 28. OpenStack deployment: Create User

To configure groups, the Identity > Groups button allows to manage the groups defined in OpenStack (see Figure 29).

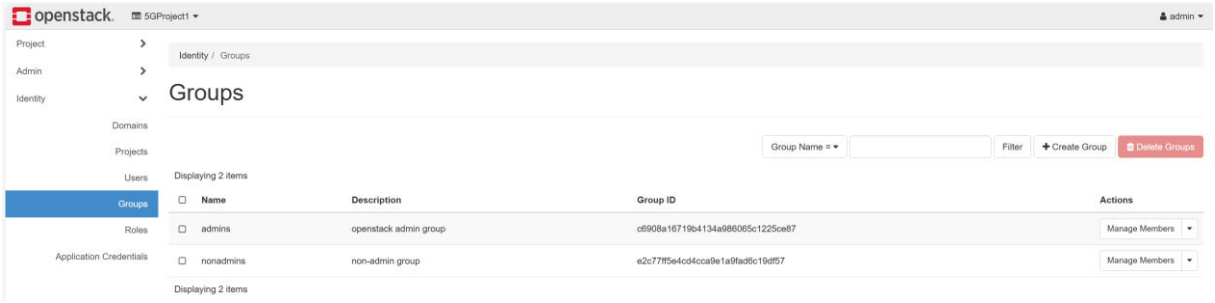

Figure 29. OpenStack deployment: Groups

To assign users to groups is necessary to click on Manage Members button (in Figure 29) which redirects you to the Group Management window (which is depicted in Figure 30), and here it is possible to manage the users of the group from a pool of already created users. As can be seen in Figure 30, salvador user has been assigned for nonadmins group.

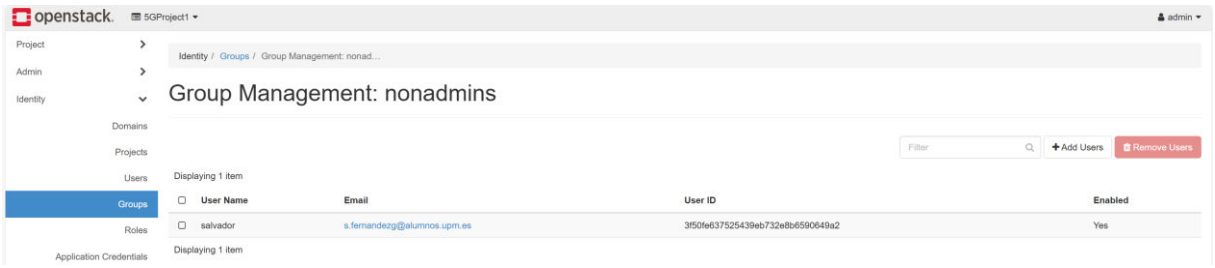

Figure 30.OpenStack deployment: Assign users to groups

Now that we have defined a user, groups and roles inside a domain, we will go ahead with the setting of a new project. Look for the Identity > Projects button and there, click the Create Project button. This will open the window from Figure 31.

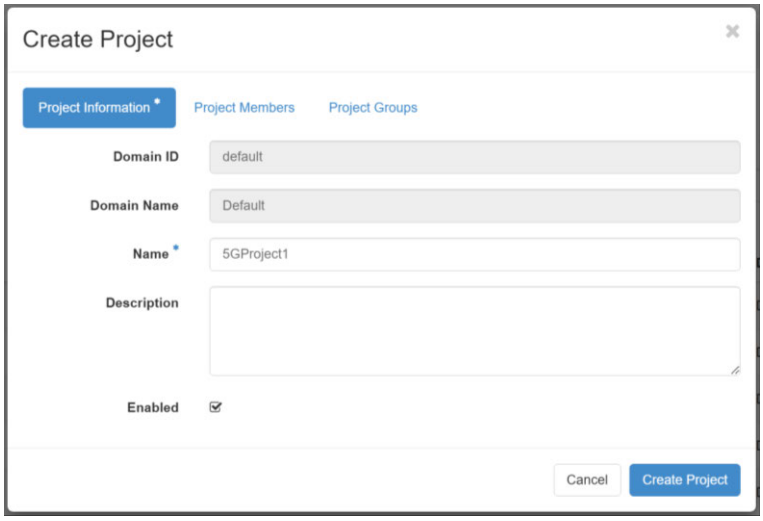

Figure 31. OpenStack deployment: Create a project (Project Information)

The creation process could be performed just by typing the name of the project, but it makes no sense if no one works on this project. Because of that OpenStack allows to edit the project in order to set the appropriate Project Members (Figure 32) and Groups (Figure 33).

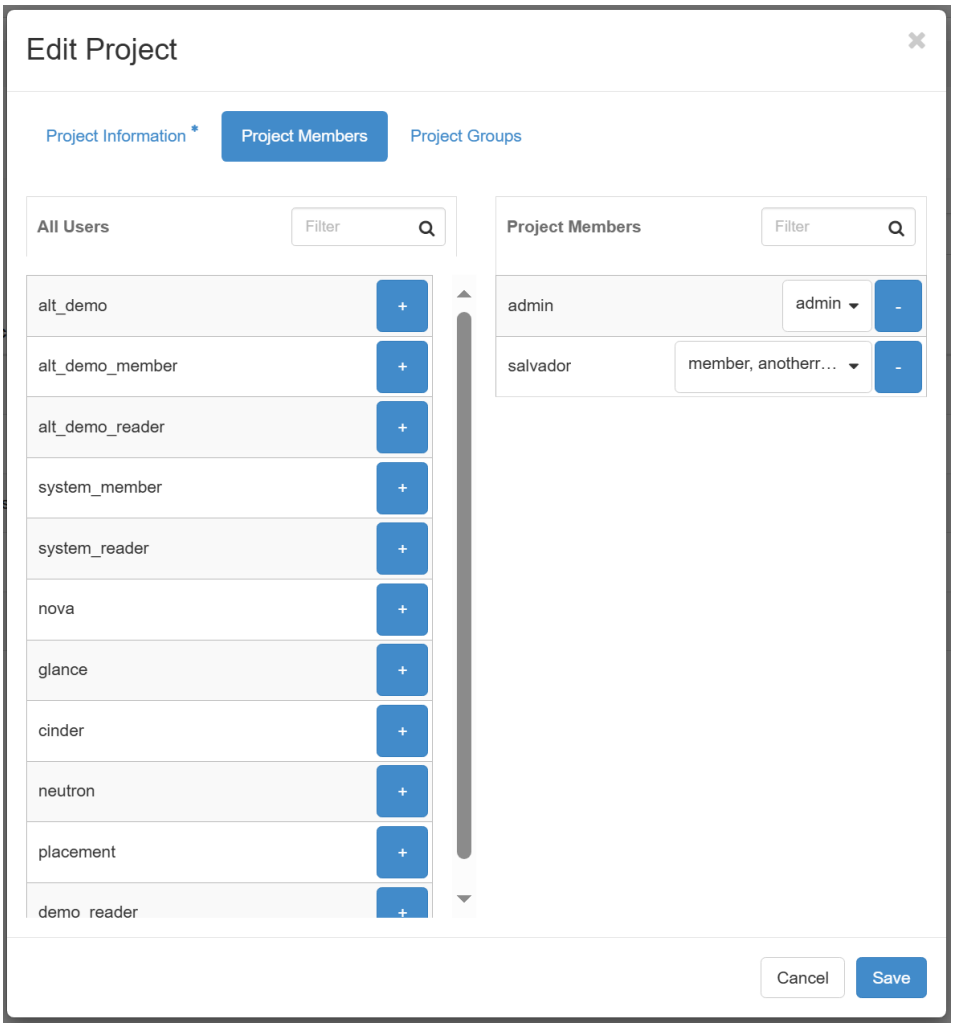

Figure 32. OpenStack deployment: Create a project (Project Members)

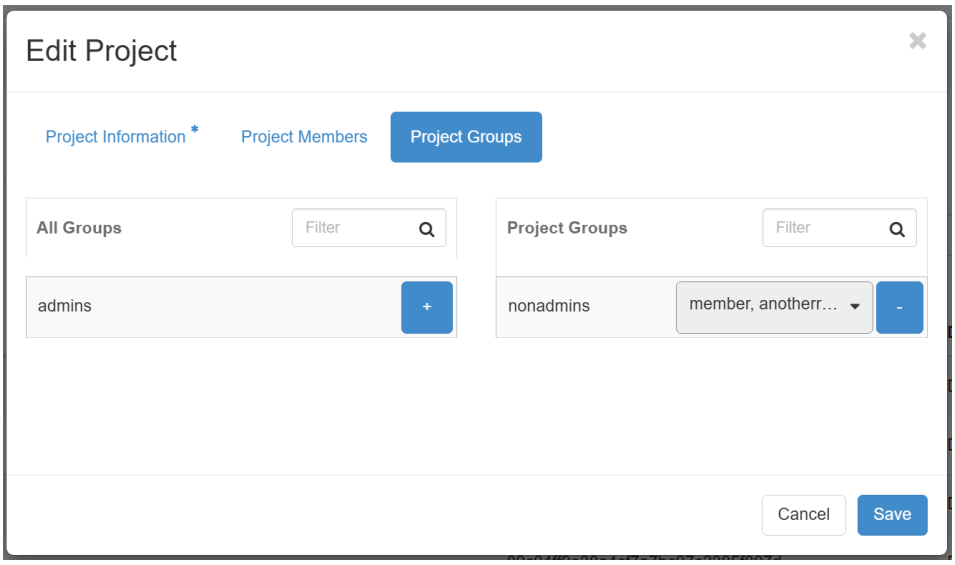

Figure 33. OpenStack deployment: Create a project (Project Groups)

4.4.3. OpenStack deployment: Network configuration

In order to ease the deployment of OpenStack, we will use the default network configuration for this project, as it is adequate to the architecture proposed in Figure 20. We can see the default configuration in Figure 34.

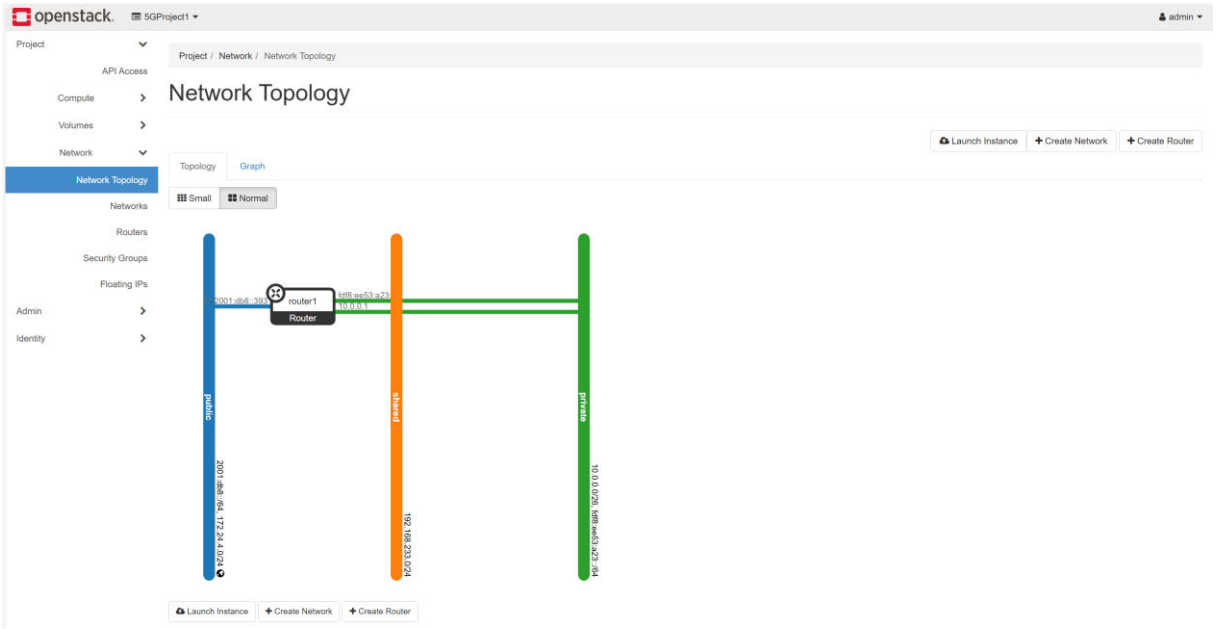

Figure 34. OpenStack deployment: Network Topology

To change any configuration in this regard, login with admin user and click on Admin > Network > Networks button, this can also be done as a project user from the Project > Network > Networks button. This procedure will open the window depicted in Figure 35.

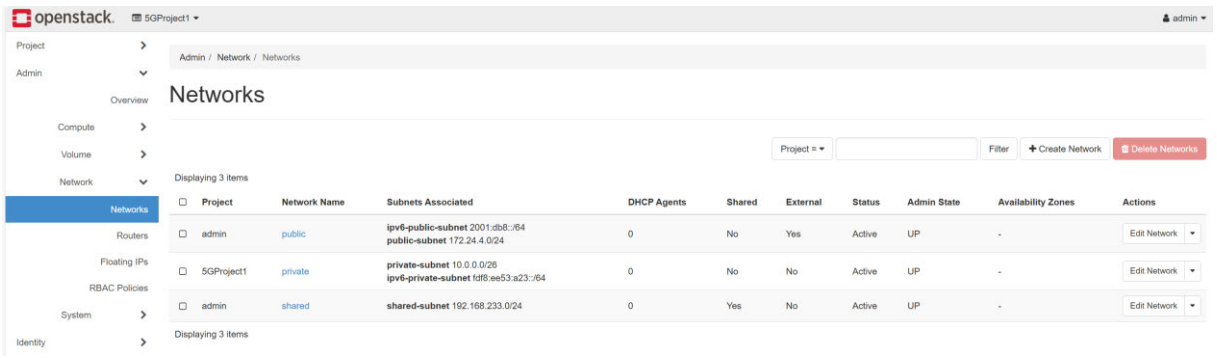

Figure 35. OpenStack deployment: Networks

To provide internet DNS servers to the instances it is necessary to click the private button and click the subnet tab (this is shown in Figure 36).

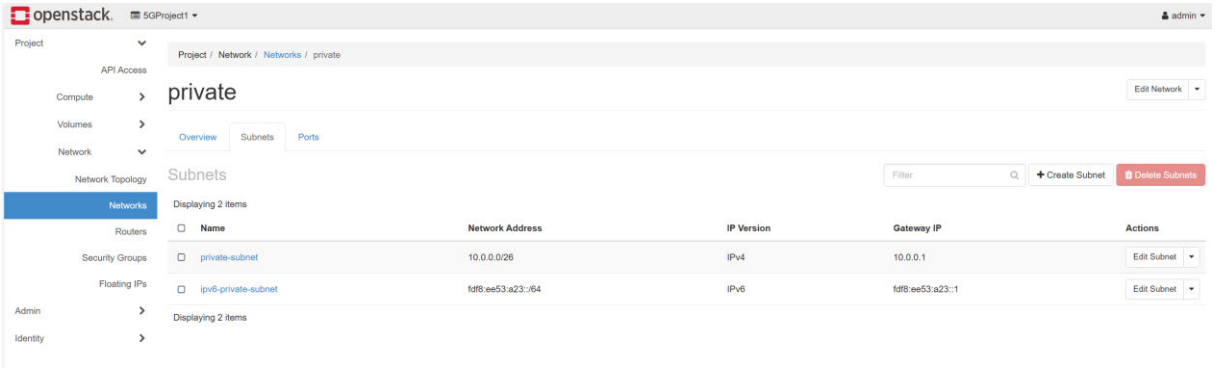

Figure 36. OpenStack deployment: Private network (subnets)

To configure the subnet, it is necessary to click Edit Subnet and in the tab Subnet details add DNS Name Servers as shown in Figure 37. In this case, google name servers have been set.

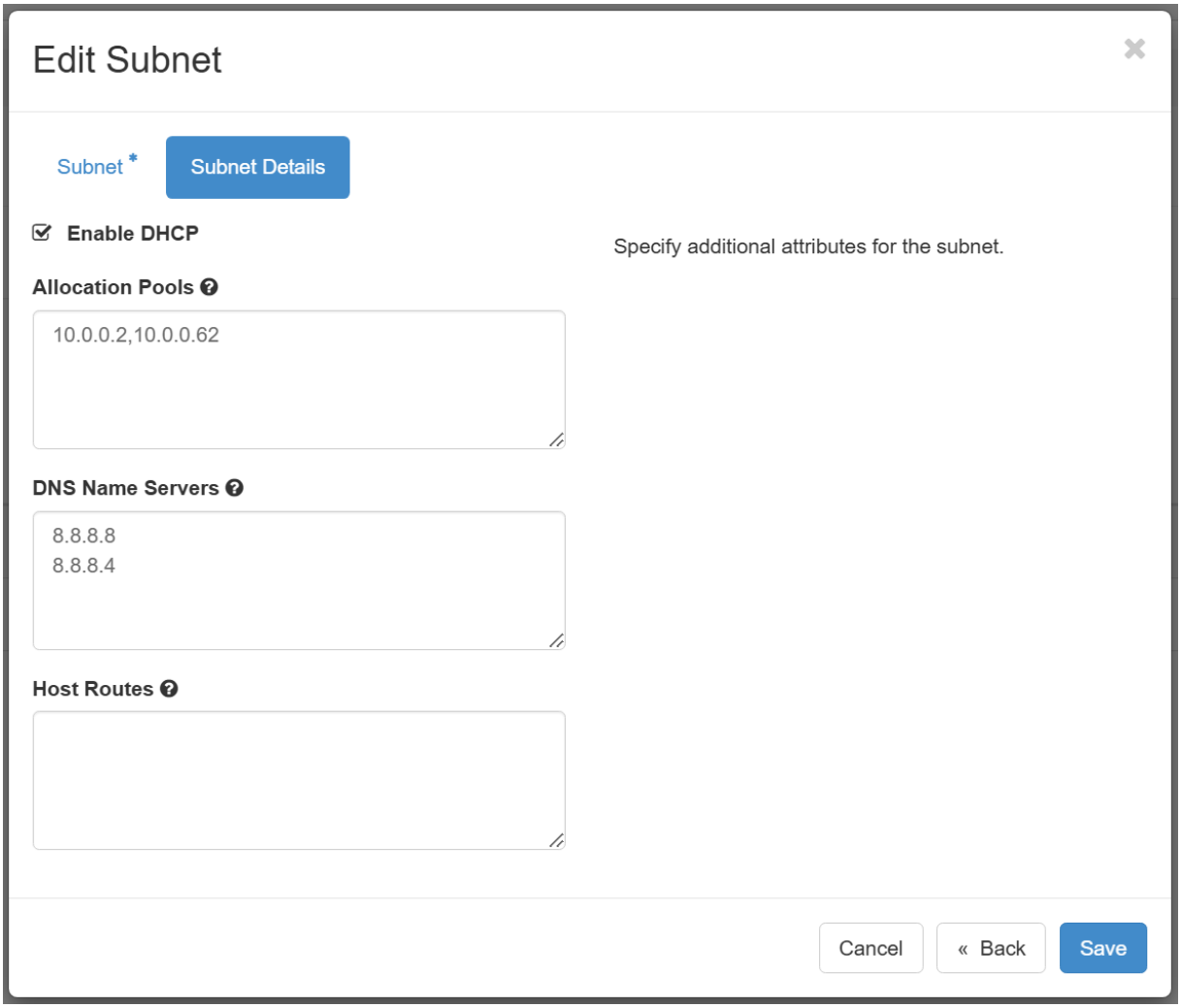

Figure 37. OpenStack deployment: Edit subnet (private-subnet)

The router configuration can also be modified as for instance the name from router1 to R1. This is done in Admin > Networks > Routers panel (this option shows all routers from all projects) or Project > Network > Routers (see Figure 38).

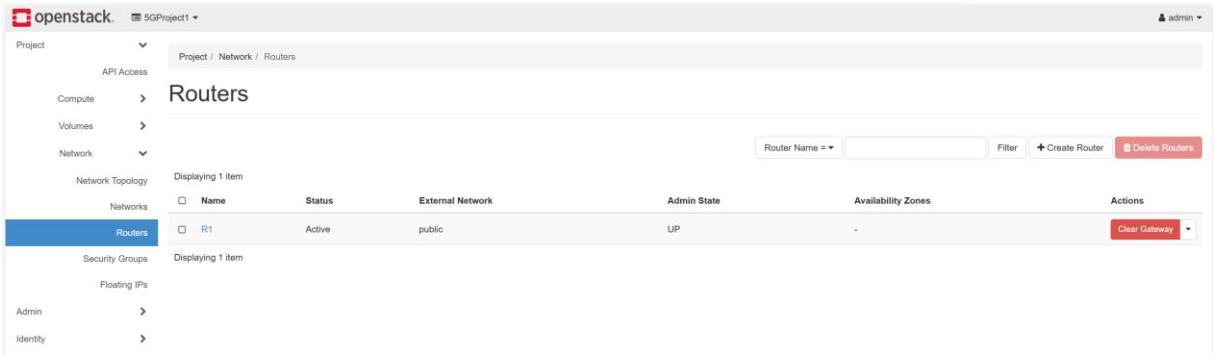

Figure 38. OpenStack deployment: Routers

To complete the network deployment, it is required to establish security groups. We will be using the default security group for this project as can be seen in Figure 39

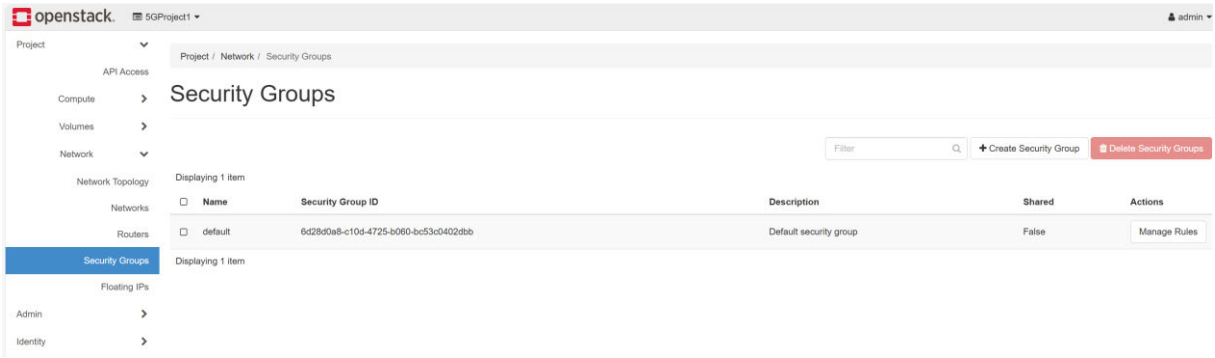

Figure 39. OpenStack deployment: Security Groups

By default, OpenStack VMs allow all the traffic from instances to the external network but the traffic from external networks to the instances is not allowed. Due to that the ICMP, SSH, DNS and HTTP(S) must be set (refer to Figure 40) to allow the correct working functionality of the instances.

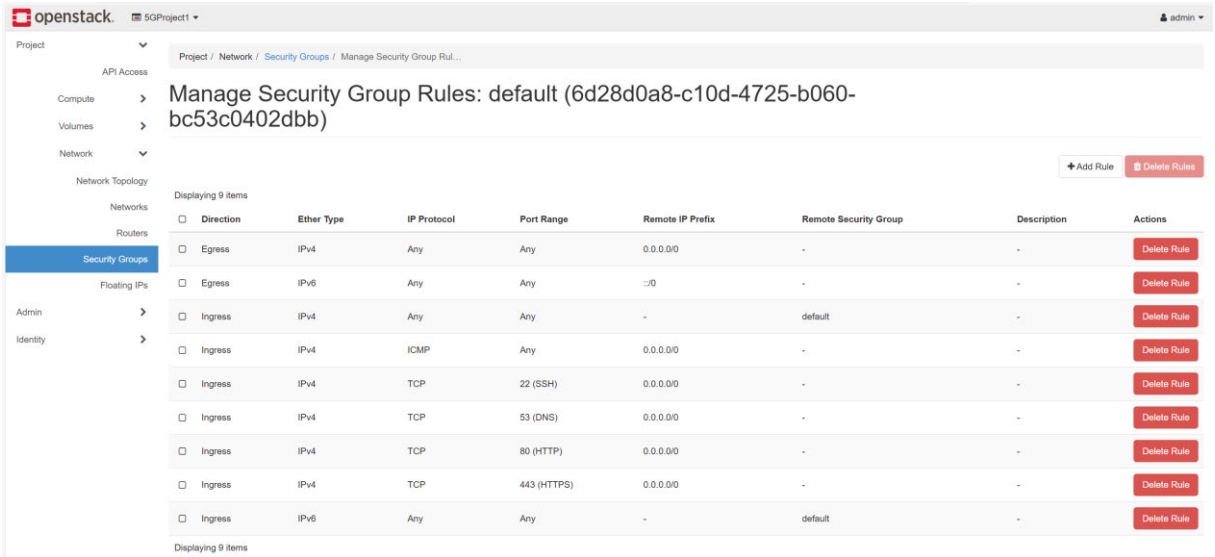

Figure 40. OpenStack deployment: Manage Security Group Rules

With this the OpenStack deployment is finished. The next step is to run the instances (VMs) which is explained deeply in the Annex 3 of this document.

4.5. Deployment of 5G System

This chapter will address the deployment of the 5G System architecture described in Figure 22 in which there are three elements to be configured: free5GC, UERANSIM gNBs, UERANSIM UEs.

4.5.1. Image preparation

To install Free5GC and UERANSIM in the corresponding VMs we will need to use UBUNTU 20.04 LTS (Focal Fossa) cloud image. To create the image, login the OpenStack Horizon web with user salvador and find the Project > Compute > Image section and press the Create Image button. Insert the img file and the format which is QCOW2 (QEMU Copy On Write version 2) in this case (refer to Figure 41).

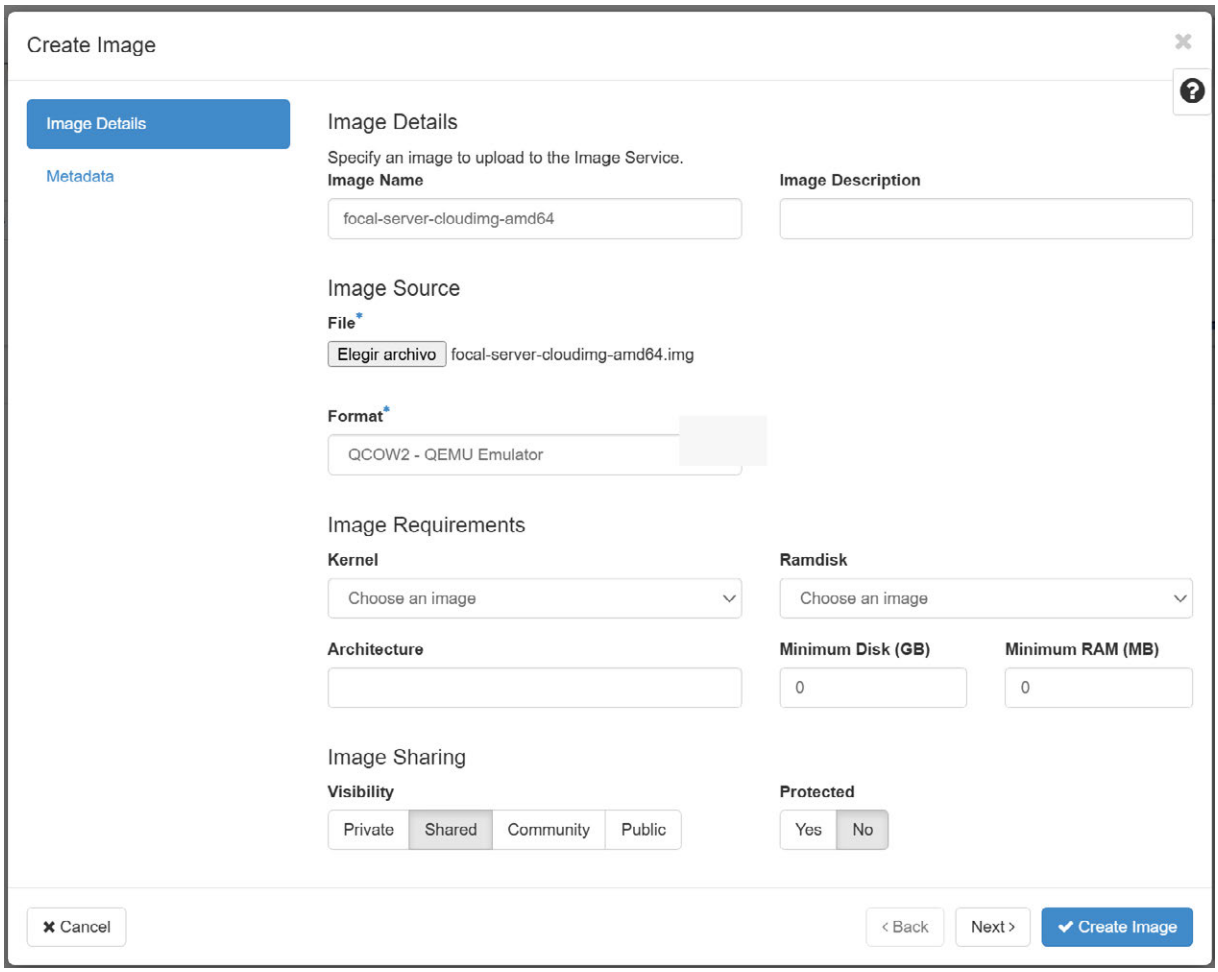

Figure 41. 5G System: Image Creation

4.5.2. Free5GC: Deployment and initial checks

Once the image is created, we will launch the instance by setting the following parameters:

- Instance Name: Free5GC
- Delete Volume on Instance Delete set to YES
- Focal image allocated

- Flavor: m1.xlarge
- Networks: private allocated
- Security Groups: default
- Key Pair: free5gc (to create a key pair refer to Figure 93)

Once the instance has been launched it must appear in Project > Compute > Instances panel with active status. If it does not work, erasing volumes that are unused, modifying quotas for the project or consulting DevStack or OpenStack documentation should help.

To provide external reachability to the instance, a Floating IP must be assigned to the VM (refer to Figure 96). To test if everything is going right, it is convenient to try reaching the floating IP in ASUS ESC4000-E10 machine (see Figure 42).

```
ectics@ectics-server:~$ ping 172.24.4.25
PING 172.24.4.25 (172.24.4.25) 56(84) bytes of data.
64 bytes from 172.24.4.25: icmp_seq=1 ttl=63 time=3.63 ms
64 bytes from 172.24.4.25: icmp_seq=2 ttl=63 time=0.693 ms
64 bytes from 172.24.4.25: icmp_seq=3 ttl=63 time=0.078 ms
64 bytes from 172.24.4.25: icmp_seq=4 ttl=63 time=0.067 ms
^C
--- 172.24.4.25 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3037ms
rtt min/avg/max/mdev = 0.067/1.116/3.627/1.471 ms
```

Figure 42. 5G System: Ping test to Free5GC instance

To SSH the instance, refer to Figure 98 and subsequent. Then, changing the password of the VM is recommended to access from the built-in OpenStack console system. To do that type the following command:

$ sudo passwd ubuntu

To prepare the VMs it is also necessary to check the internet connection as it is made in Figure 43.

```
ubuntu@free5gc:~$ ping google.com
PING google.com (142.250.200.78) 56(84) bytes of data.
64 bytes from mad07s24-in-f14.1e100.net (142.250.200.78): icmp_seq=1 ttl=106 time=3.49 ms
64 bytes from mad07s24-in-f14.1e100.net (142.250.200.78): icmp_seq=2 ttl=106 time=3.06 ms
64 bytes from mad07s24-in-f14.1e100.net (142.250.200.78): icmp_seq=3 ttl=106 time=2.78 ms
64 bytes from mad07s24-in-f14.1e100.net (142.250.200.78): icmp_seq=4 ttl=106 time=2.78 ms
^C
--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 2.777/3.026/3.489/0.290 ms
```

Figure 43. 5G System: Ping test to check internet connection inside Free5GC instance

If there is no IP connectivity, it is necessary to check that the private network subnet configuration is well set with DNS nameservers configured. To avoid errors it is important to update and upgrade Ubuntu by typing the following commands

$ Sudo apt update
$ Sudo apt upgrade

4.5.3. Free5GC VM: Prerequisites

Free5gc documentation establishes some prerequisites, for instance, setting the VM hostname as free5gc. In our case, this is not necessary as it was set automatically by OpenStack. It is also required to change the hosts file located in /etc folder in order to locate the free5gc NF host (by adding line 2 from Figure 44). The following command could be used to modify that file:

```
$ sudo nano /etc/hosts
```

```
127.0.0.1 localhost
127.0.1.1 free5gc

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Figure 44. Free5GC VM: hosts file configuration

After checking the content and saving this file, it is necessary to check kernel compatibility as it is needed for UPF to establish the corresponding GTP tunnels correctly. In this case, focal server has 5.4.x by default (check out Figure 45) which is compatible with free5GC.

```
ubuntu@free5gc:~$ uname -r
5.4.0-186-generic
```

Figure 45. Free5GC VM: Kernel version check

The next prerequisite is to install Go version 1.21.8. This could be done by using the following commands:

```
$ wget https://dl.google.com/go/go1.21.8.linux-amd64.tar.gz
$ sudo tar -C /usr/local -zxvf go1.21.8.linux-amd64.tar.gz
$ mkdir -p ~/go/{bin,pkg,src}
$ # The following assume that your shell is bash:
$ echo 'export GOPATH=$HOME/go' >> ~/.bashrc
$ echo 'export GOROOT=/usr/local/go' >> ~/.bashrc
$ echo 'export PATH=$PATH:$GOPATH/bin:$GOROOT/bin' >> ~/.bashrc
$ echo 'export GO111MODULE=auto' >> ~/.bashrc
$ source ~/.bashrc
```

After this, it is advisable to do a simple check to see if the installation has been successful (refer to Figure 46).

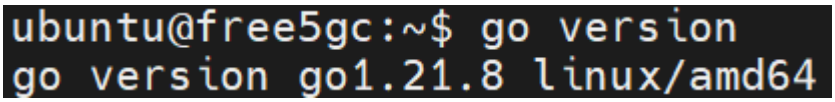

Figure 46. Free5GC VM: Go version check

Following the documentation, the next step is to install and run MongoDB server. As our system is not compatible with AVX it is needed to install MongoDB version 4.4 or lower. Enter the following commands to install. Figure 47**$ sudo apt -y update**
$ sudo apt -y install wget git

To check if the service is running, follow the procedure in Figure 47.

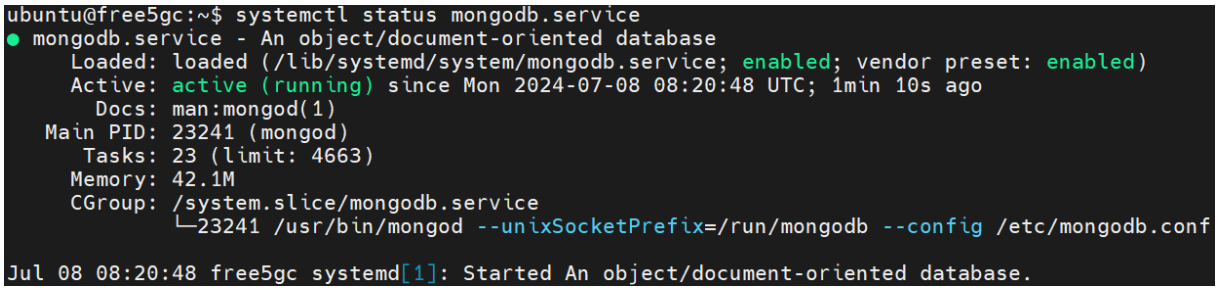

Figure 47. Free5GC VM: MongoDB successful status check

Now that mongo is running, it is time to install the supporting packages for the user-plane. This is made by the following commands:

$ sudo apt -y update
$ sudo apt -y install git gcc g++ cmake autoconf libtool pkg-config libmnl-dev libyaml-dev

4.5.4. Free5GC VM: Installation

This is the moment to install Free5GC using the following commands

$ cd ~
$ git clone --recursive -b v3.4.1 -j `nproc`
https://github.com/free5gc/free5gc.git
$ cd free5gc

At this moment, Free5GC is installed in the VM and the first step to set up this machine to work with this tool, is running the reload_host_config.sh script inside the folder named free5gc that appeared once the github repo has been cloned. This script configures the network for the interface that will be used to reach the 5GC. To see the available interfaces use ip command (more detail in Figure 48).

```
ubuntu@free5gc:~/free5gc$ 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: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1442 qdisc fq_codel state UP group default qlen 1000
    link/ether fa:16:3e:a4:5e:3b brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.38/26 brd 10.0.0.63 scope global dynamic ens3
       valid_lft 31368sec preferred_lft 31368sec
    inet6 fdf8:ee53:a23:0:f816:3eff:fea4:5e3b/64 scope global mngtmpaddr noprefixroute
       valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fea4:5e3b/64 scope link
       valid_lft forever preferred_lft forever
```

Figure 48. Free5GC VM: Availability check of the interfaces

In Figure 48 it can be seen that the name of the only interface available is ens3 and to set this interface for Free5GC, the following commands must be used:

```
$ ./reload_host_config.sh ens3
```

Now that the network settings have been configured, the NFs will be built by using the following command:

```
$ make
```

Once the command finishes, the control plane elements are ready to test, but first it is needed to prepare the user plane function with the UPF function. To allow the UPF to compile, first run the following commands:

```
$ git clone -b v0.8.6 https://github.com/free5gc/gtp5g.git
$ cd gtp5g
$ make
$ sudo make install
```

This will allow the UPF to compile successfully when running the next commands:

```
$ cd ~/free5gc
$ make upf
```

To finish the configuration of this VM, we will install the WebConsole GUI by installing nodejs:

```
$ curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
$ sudo apt update
$ sudo apt install -y nodejs
$ sudo corepack enable
```

And then the WebConsole has to be built inside the free5gc folder:

```
$ make webconsole
```

Installing GNOME is highly recommended in order to provide a graphical interface for managing webconsole and capture packets from the interface. To do so, we will be using this command:

$ sudo apt install ubuntu-desktop

At this point, Free5GC VM is fully deployed, configured and ready for testing. Free5GC provides some scripts that allow us to validate not only that the installation has been successful but also that the 5GC is working as expected. The results of these tests can be found in Annex 4. Once the 5GC has been validated we are able to run the 5GC inside the VM and, to do so, the following command must be used:

`$ ./run.sh`

4.5.5. Free5GC VM: Configuration

In Free5GC VM, the configuration files are located in ~/free5gc/config folder. To follow the guidelines in the 5G System architecture depicted in Figure 22 it is required to modify the following three configuration files:

- ~/free5gc/config/amfcfg.yaml
- ~/free5gc/config/smfcfg.yaml
- ~/free5gc/config/upfcfg.yaml

In the AMF configuration file (settings Figure 49) it is required to modify ngapIPList which is the AMF IP that will use gNBs to connect its N2 interface (see Figure 9).

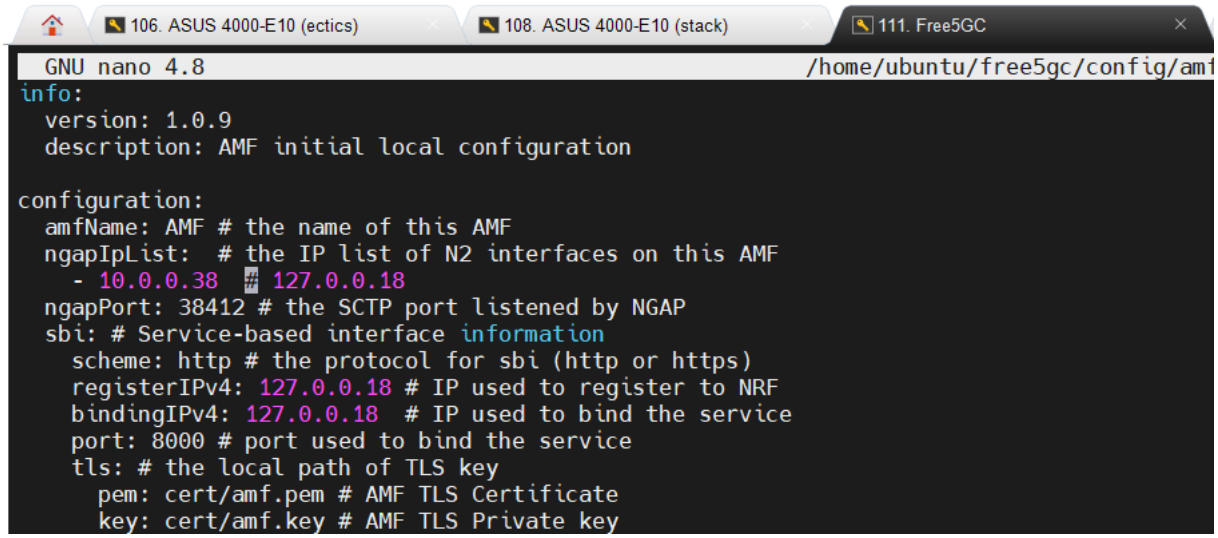

Figure 49. Settings for AMF configuration file

SMF establishes data sessions and in consequence, the configuration file must know the IP of the UPF N3/N9 interface to whom it is attached to. In Figure 50 as can be seen, it has been set the free5gc IP (10.0.0.38) which must be the same used as the defined in Figure 51.

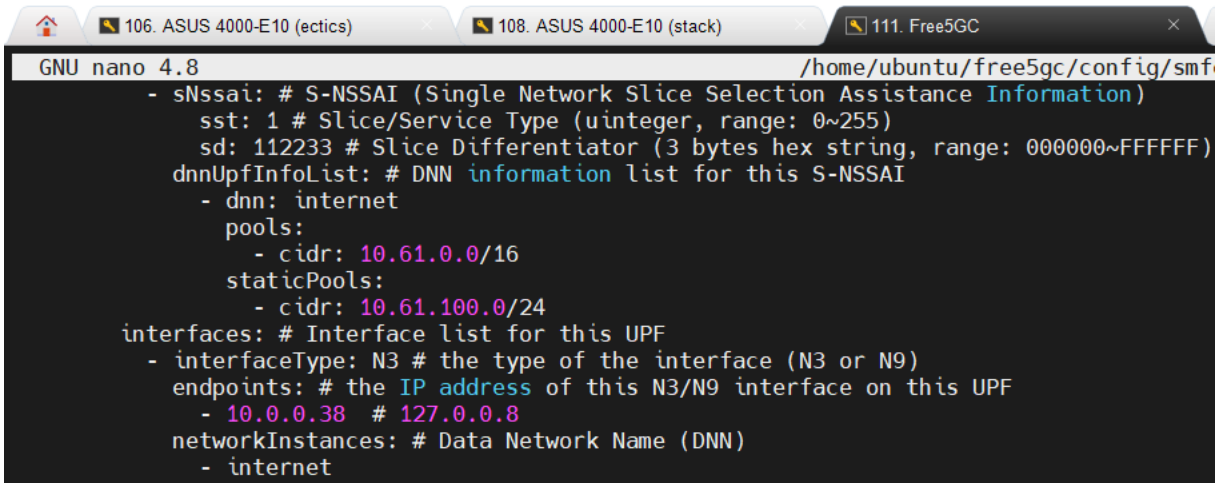

Figure 50. Settings for SMF configuration file

The UPF configuration file must define the IP of N3/N9 interface which connects to gNBs as defined in Figure 51.

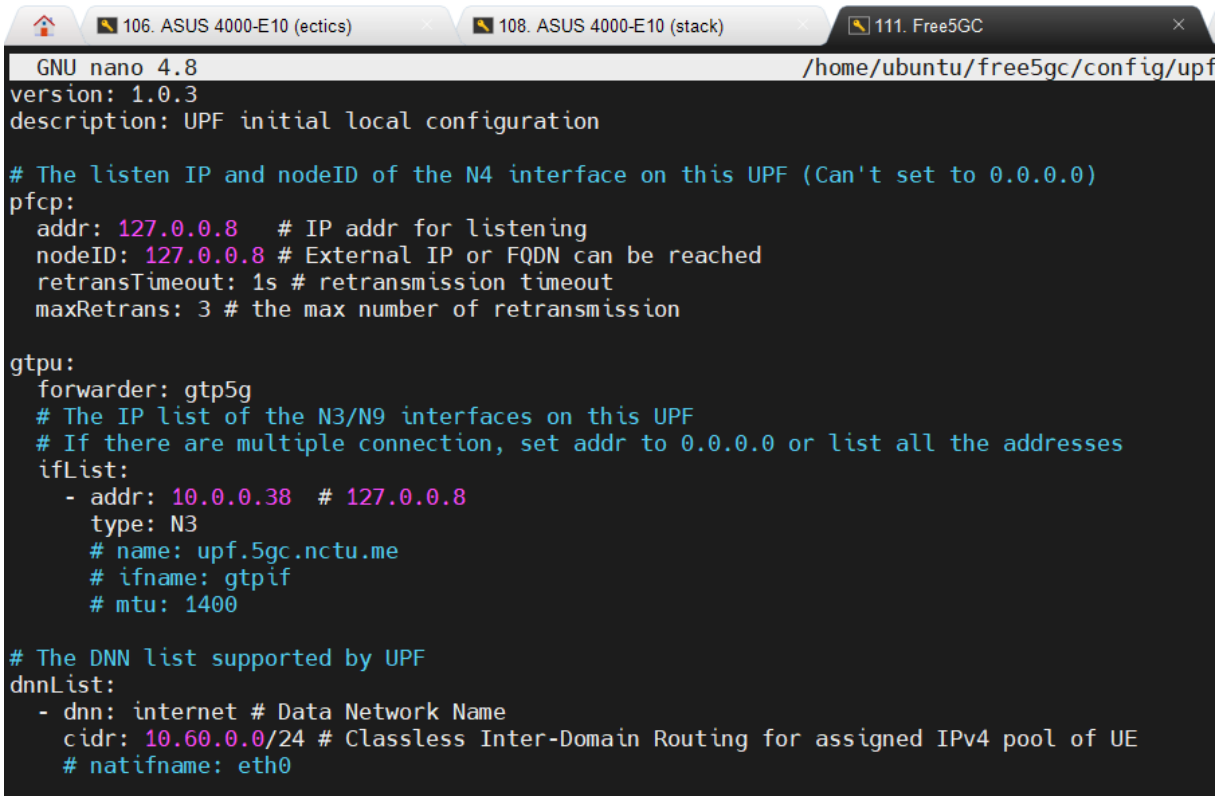

Figure 51. Settings for UPF configuration file

4.5.6. UERANSIM: Deployment and installation

To configure UERANSIM first is needed to launch an instance with the following configuration:

- Instance Name: UERANSIM
- Delete Volume on Instance Delete set to YES
- Focal image allocated

- Flavor: ds1G
- Networks: private allocated
- Security Groups: default
- Key Pair: free5gc

If the instance is successfully running, the connection via SSH should be possible. From an SSH session it is needed to install the UERANSIM tool and all the required packets. To install UERANSIM, the following commands must be used:

```
$ cd ~
$ git clone https://github.com/aligungr/UERANSIM
$ cd UERANSIM
$ git checkout 3a96298
$ sudo apt update
$ sudo apt upgrade
```

In order to build c files, it is necessary to install the packages and libraries shown in the following commands:

```
$ sudo apt install make
$ sudo apt install g++
$ sudo apt install libsctp-dev lksctp-tools
$ sudo apt install iproute2
$ sudo snap install cmake --classic
```

To build UERANSIM packages, the following commands will be used:

```
$ cd ~/UERANSIM
$ make
```

After the execution, it must be seen a message similar to the one shown in Figure 52.

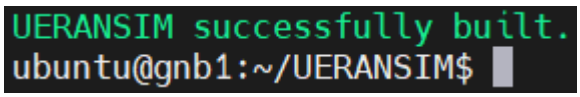

Figure 52. UERANSIM VM: Build message successful

4.5.7. UERANSIM VMs: gNB configuration

The requirement for 5G System gNBs to work is to have accessibility to the AMF which is located in free5GC VM (with IP 10.0.0.38). However, it is also demanded to set '*linkIp*', '*ngapIp*' and '*gtpIp*' with the ip allocated by OpenStack to gNB instance. The result is two configuration files, one for gNB1 instance (Figure 53) and another for gNB2 instance (Figure 54). These files can be found in ~/UERANSIM/config/free5gc-gnb.yaml file.

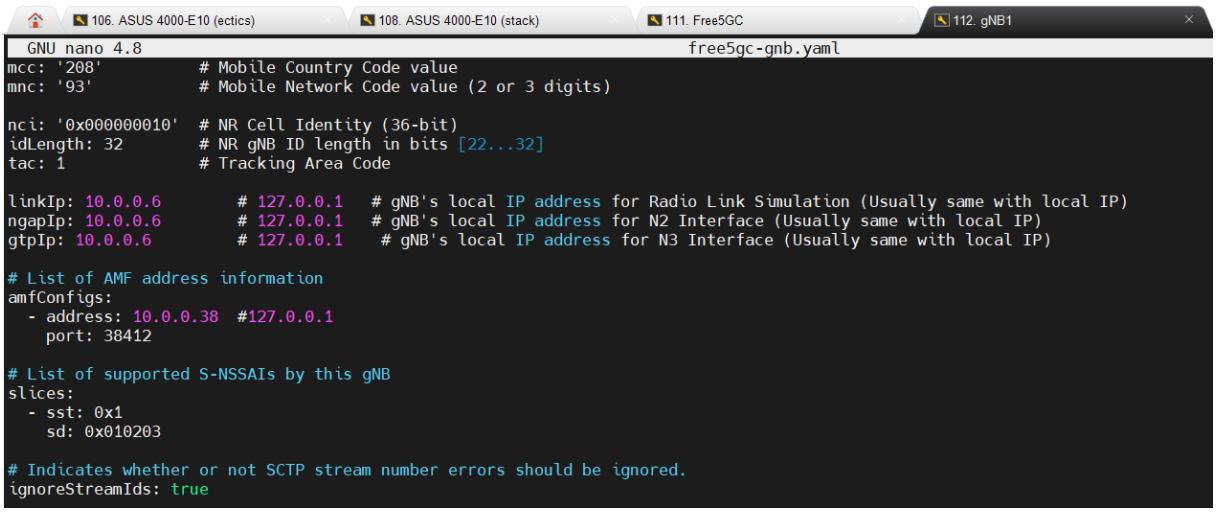

Figure 53. gNB1 configuration file: free5gc-gnb.yaml

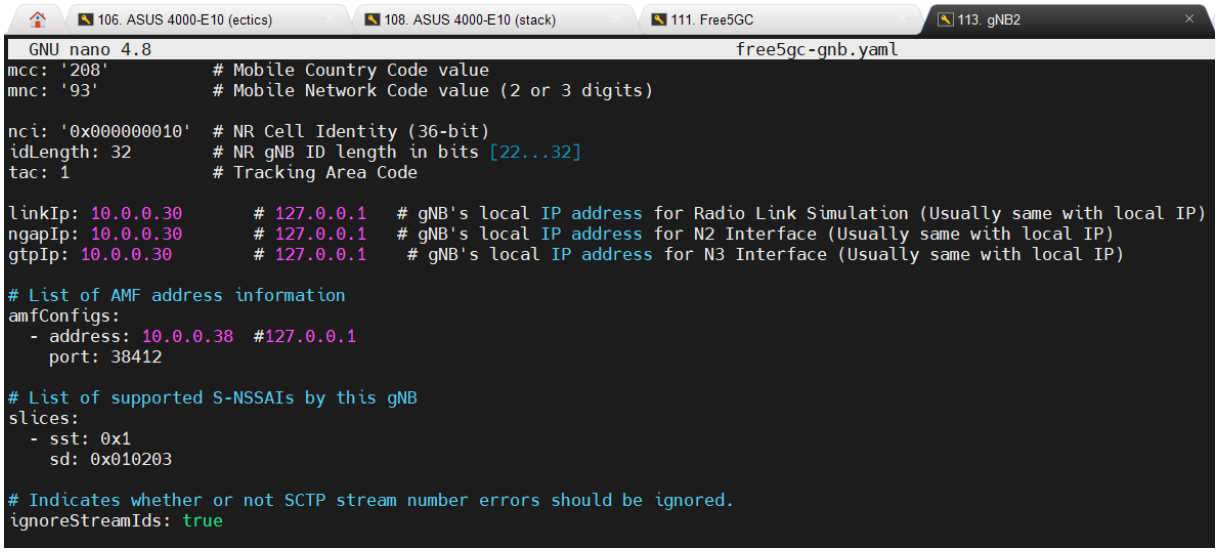

Figure 54. gNB2 configuration file: free5gc-gnb.yaml

4.5.8. UERANSIM VMs: UE configuration

The configuration file of UEs can be found at ~/UERANSIM/config/free5gc-ue.yaml and must be set according to the configuration set in WebConsole subscriber panel. In our case the only parameters that have been changed are the IMSI number (Figure 55) which identifies the UE (following the allocation of Table 3), the operator type (Figure 56) to OP (Operator Code) and the IP of the simulated gNBs to which the UE is going to attach to (Figure 57 and Figure 58). Note that OPC (Derived Operator Code) is more complex and has been not tested in our deployment

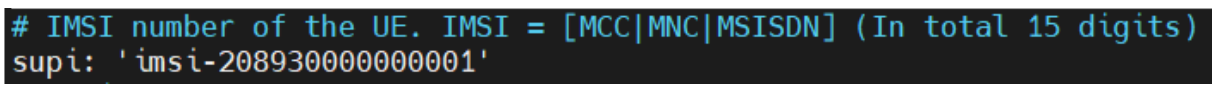

Figure 55. UE1 IMSI parameter in free5gc-ue.yaml

```
# This value specifies the OP type and it can be either 'OP' or 'OPC'
opType: 'OP'
```

Figure 56. opType parameter in free5gc-ue.yaml

```
# List of gNB IP addresses for Radio Link Simulation
gnbSearchList:
  - 10.0.0.6  # 127.0.0.1
```

Figure 57. UE1 and UE2 free5gc-ue.yaml configuration file (gnbSearchList parameter)

```
# List of gNB IP addresses for Radio Link Simulation
gnbSearchList:
  - 10.0.0.30  # 127.0.0.1
```

Figure 58. UE3 and UE4 free5gc-ue.yaml configuration file (gnbSearchList parameter)

4.5.9. Backup System

This solution relies on OpenStack backup solution system which consists of Snapshots. A Snapshot is a mechanism that allows to create new images from running instances. The way it works is simple, it saves the main disk of an instance into an image so that later on, it can be booted as a new instance containing the same data and configuration. To create a Snapshot go to the instance list panel in Horizon (see Figure 96) and click the Create Snapshot button which will show the window in Figure 59.

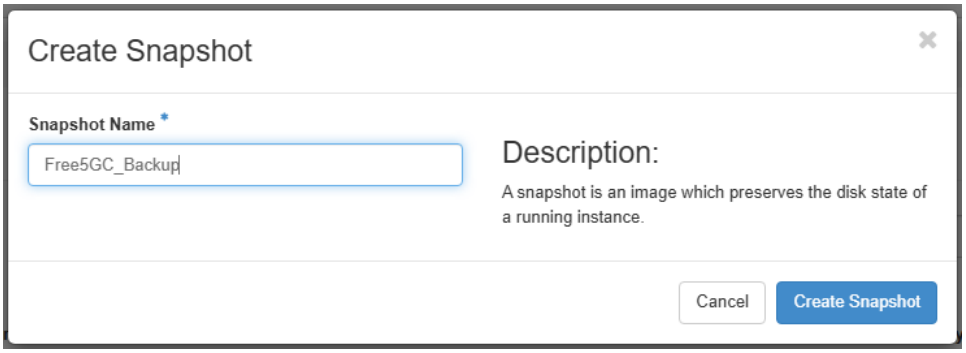

Figure 59. 5G System: Creation of Free5GC VM backup (Create Snapshot)

5. Results

In this chapter, we will be performing some tests in order to check how valid this solution is and whether it guarantees not only the specifications established above in section 3.1 but also the guidelines described in section 4.2 .

5.1. Physical configuration test

To test the physical architecture described in Figure 17 we will perform two tests:

- Test 1: we will connect to an external network which is out of the UPM network, and we will try to connect through RDP to demonstrate that the network is not reachable without connecting to UPM VPN.
- Test 2: We will continue in the same environment set in Test 1 but activating the VPN service in order to demonstrate if the network can establish RDP connection with PHYSICAL CLIENT.

In test 1 (Figure 60) we connected to the Wi-Fi [28] with SSID (Service Set Identifier) equal to '*Pixelcito*' (this Wi-Fi network was created using the Wi-Fi hotspot option on a Pixel 8 phone) whose network is identified by 192.168.156.0/24. In the command prompt (Figure 60) can be seen the '*Wireless LAN adapter Wi-Fi*' interface in which the computer is attached to that network. This interface has the IPv4 address 192.168.156.93 that does not belong to the UPM network domain. As can be seen we are not able to connect DESKTOP-AOG6A0Q that is the hostname of our implemented PHYSICAL CLIENT set in Figure 17.

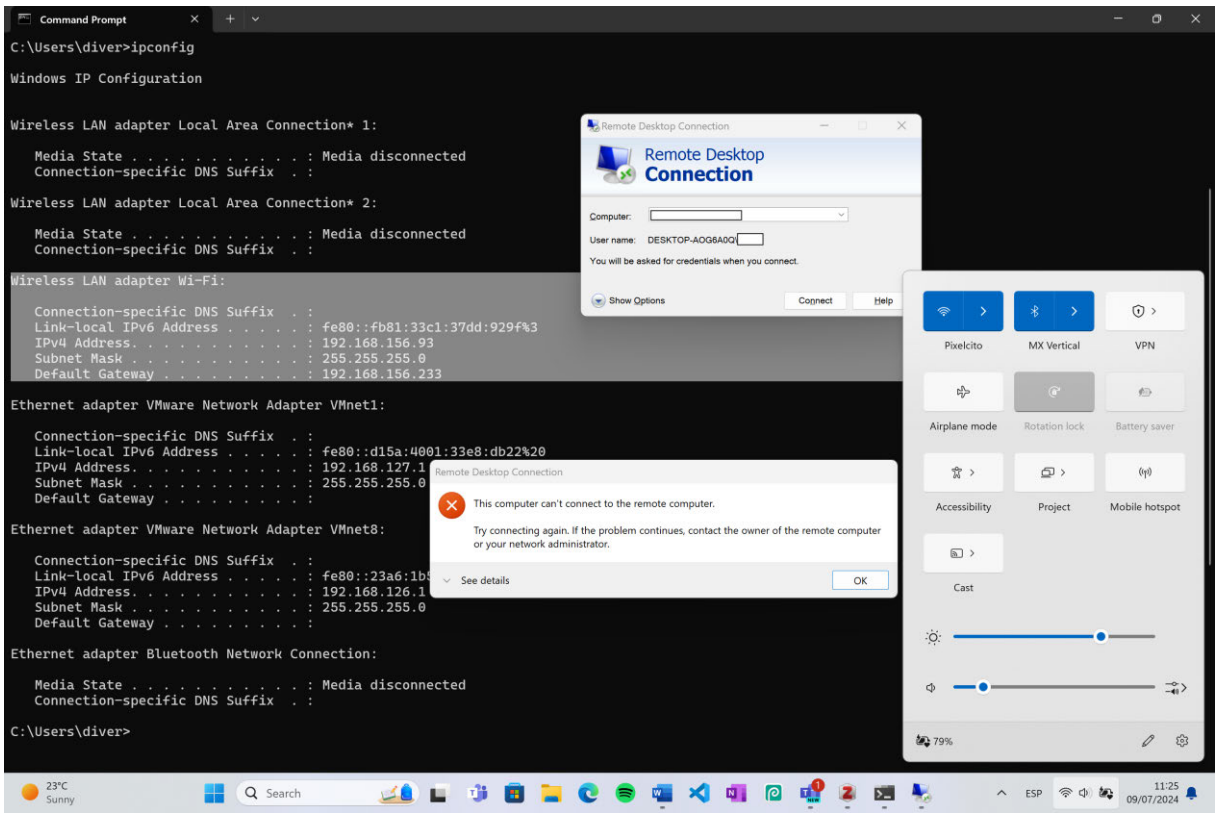

Figure 60. Test 1: Demonstration of bad connectivity without VPN service

Moving to the second test where UPMvpn is activated, the network connection is established with UPM internal network (as can be seen in Figure 61) allowing us to enter the credentials.

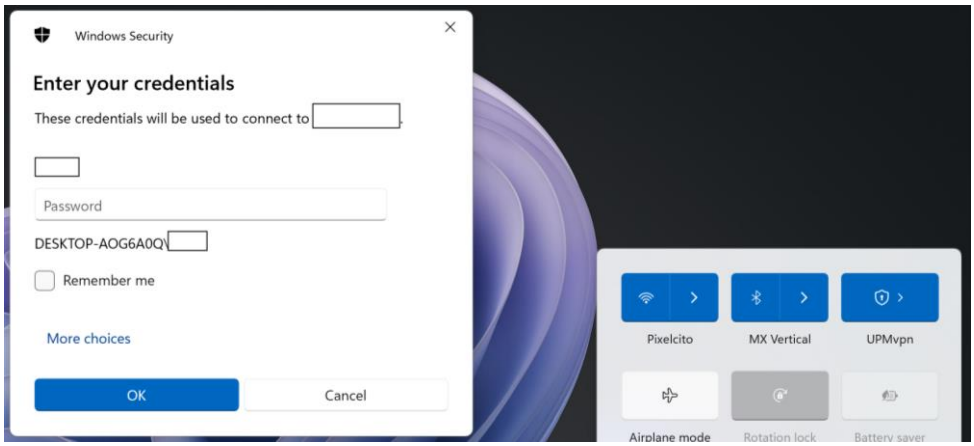

Figure 61. Test 2: Login request

Once the credentials are correctly entered, the RDP connection is established allowing us to SSH the ASUS ESC4000-E10 server as can be seen in Figure 62.

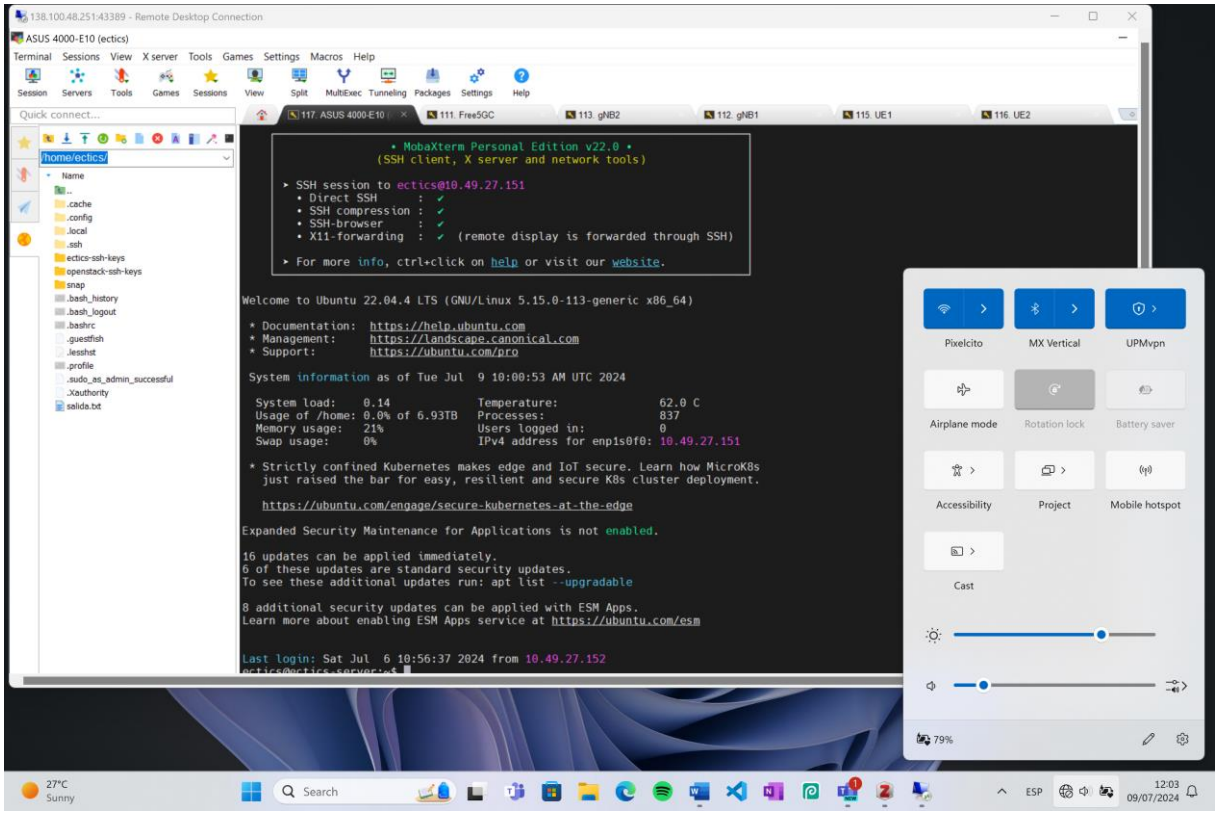

Figure 62. Test 2: Remote connection established with VPN enabled

5.2. Virtualization solution check

5.2.1. Creation of VMs

The creation of VMs is as simple as login into OpenStack Horizon web GUI and going to the Instances panel (Figure 63).

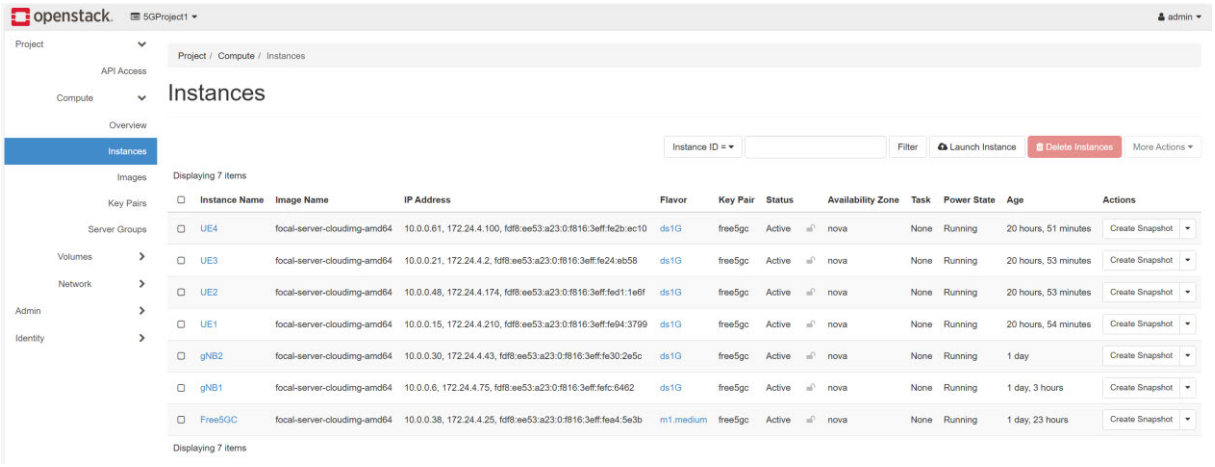

Figure 63. VM creation test: Instances before creation

To create the virtual machine, we have to create an instance (refer to Annex 3), as for example another UE named '*UE5*'.

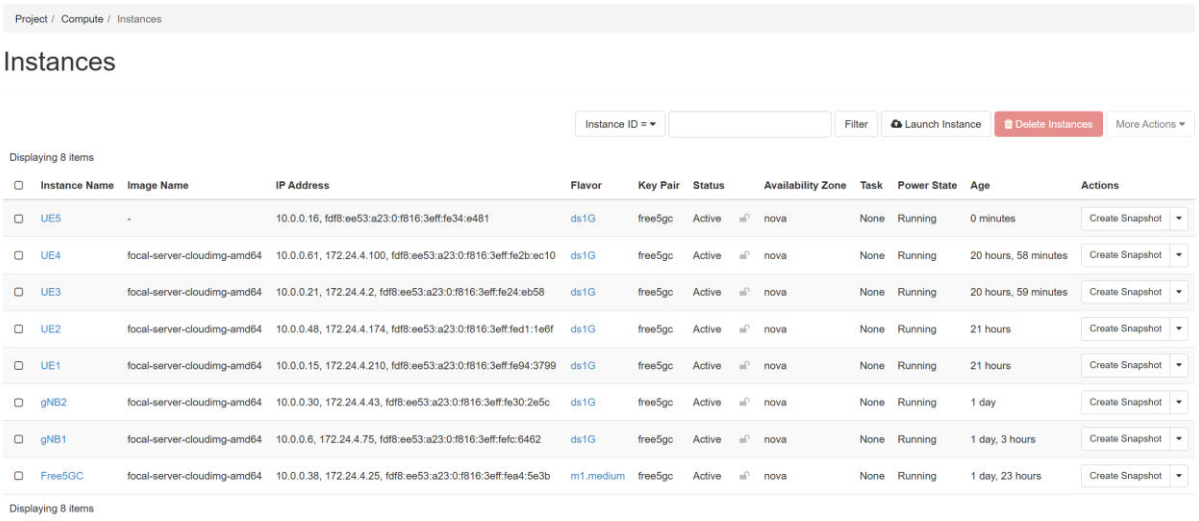

Figure 64. VM creation test: Instances after creation

As can be seen in Figure 64, the UE5 instance appears in the list of instances with the status of Active, which means that the VM has been created successfully.

5.2.2. Modification of VMs

This test is made to prove that the VM already created, has the ability to be modified. In the OpenStack instances panel can be observed that there is an action panel in which there are some options available (see Figure 65).

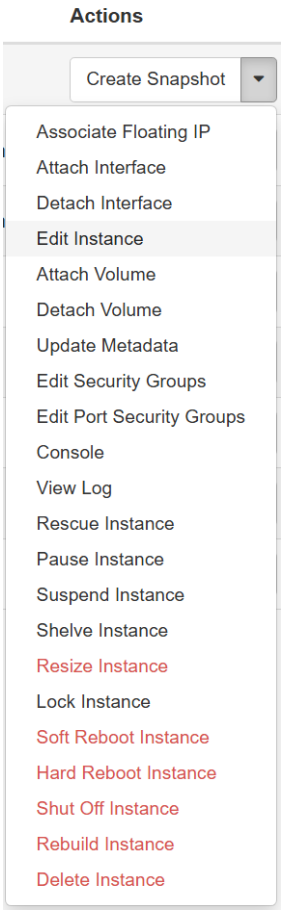

Figure 65. VM actions panel when instance is running

In the VM UE5 that has been created above in section 5.2.1 we proceed with the edition of the instance by clicking '*Edit Instance*' button, because the machine is active the configuration options may be less than expected but the name is going to be modified to '*UE5_modified*' (see Figure 66).

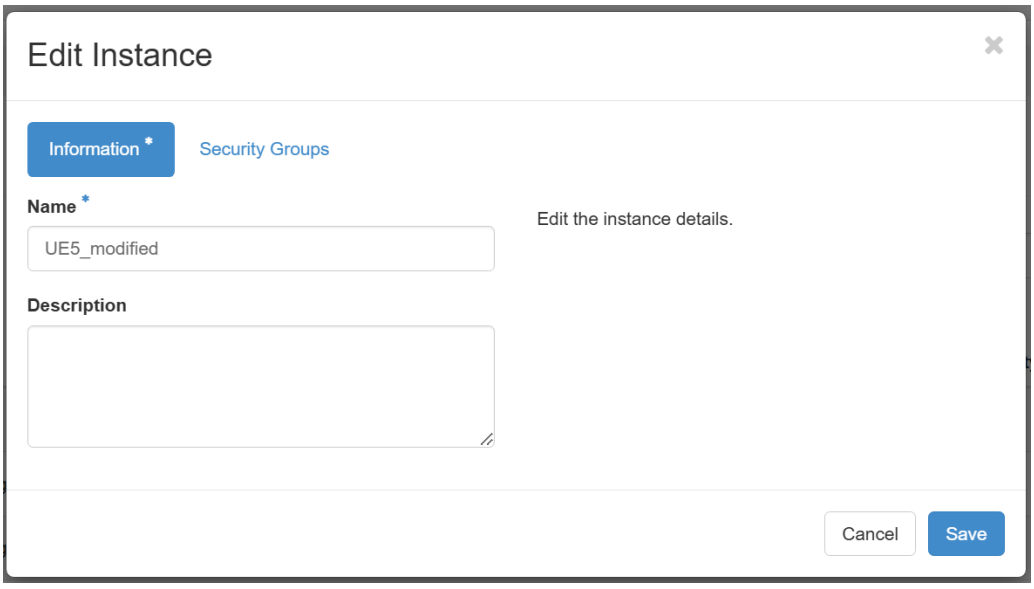

Figure 66. VM Modification test: Edit instance panel when instance is running

As we can see in Figure 67 the edition option has been executed correctly.

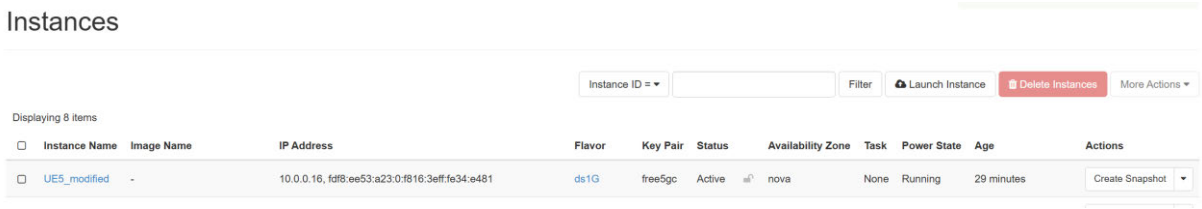

Figure 67. VM Modification test: Instance modified while running

Now we proceed to shut down the instance to see whether the Actions panel shows. As we can see in Figure 68 there are less modification options than if the instance is working.

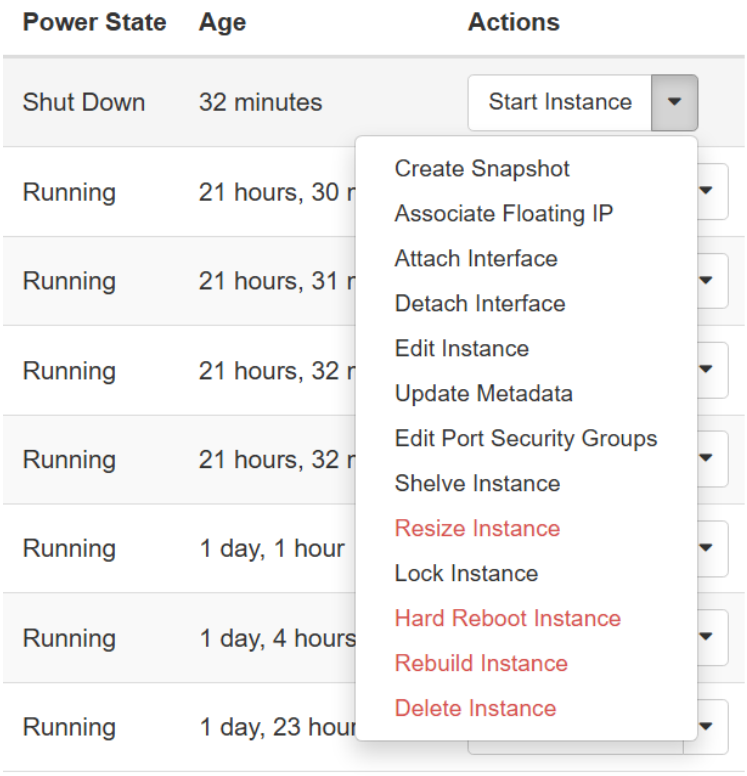

Figure 68. VM actions panel when instance is not running

When we click on 'Edit Instance' button we notice that we have the same options as in Figure 66.

A conclusion for this test, is that the virtualization solution provides some modification options as for example:

- Attach/Detach network interfaces
- Instance name
- Security groups for instance
- Security groups for ports
- Resize instances
- Attach/Detach volumes to the instance

5.2.3. Deletion of VMs

To remove the instance created in section 5.2.1 we must click in '*Delete Instance'* button in the Actions panel shown in Figure 68. Once it is clicked, it starts the deletion process shown in Figure 69.

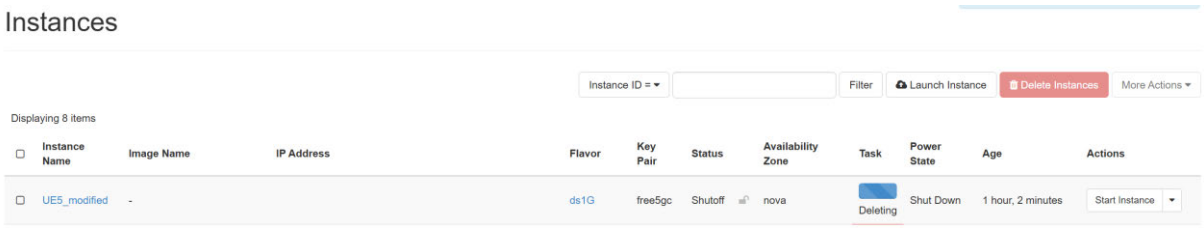

Figure 69. VM Deletion test: UE5_modified instance in deletion process

Once the process of deletion has finished, the instance no longer appears in the instance list panel shown in Figure 70.

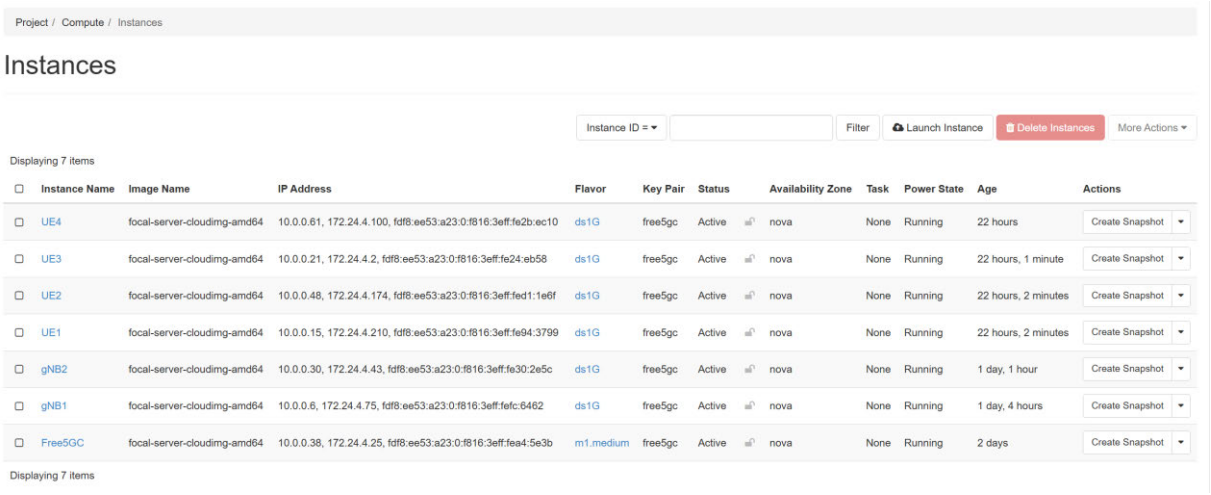

Figure 70: VM Deletion test: Instance list after deletion process finishes

From the previously seen, we can conclude that this test has yielded successful results.

5.3. Free5GC Testing

To test the 5G System implementation, the UEs must have internet connection by establishing a tunnel to the 10.60.0.0/24 network which is the one that attaches it to its serving gNodeB and indirectly gives it access to the SMF (refer to Figure 22). To perform this test, it is necessary to run Free5GC by running the run.sh script contained in the free5gc folder. After that, we will run the run_webconsole.sh script which is in the same folder. This last execution must show the messages '*Listening and serving HTTP on 0.0.0.0:5000*' and '*Webconsole-AF Registration to NRF success*' as shown in Figure 71 which indicates that Webconsole is up and running.

```
2024-07-09T12:56:57.476319425Z [INFO][WEBUI][Init] Create tenant: admin
2024-07-09T12:56:57.476729070Z [INFO][WEBUI][Init] Create user: admin
2024-07-09T12:56:57.482272062Z [INFO][WEBUI][Main] OAuth2 setting receive from NRF: true
2024-07-09T12:56:57.482392927Z [INFO][WEBUI][Init] Webconsole-AF Registration to NRF success
2024-07-09T12:56:57.808844149Z [INFO][WEBUI][Consumer] GetTokenCtx: NRF nnrf-disc
2024-07-09T12:56:57.815021703Z [INFO][WEBUI][Consumer] GetTokenCtx: NRF nnrf-disc
2024-07-09T12:56:57.820231553Z [WARN][WEBUI][BillingLog] conf map[basePath:/tmp/webconsole cert:cert/chf.pem key:cert/chf.key]
2024-07-09T12:56:57.820379153Z [INFO][WEBUI][BillingLog] Billing server Start
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Listening and serving HTTP on 0.0.0.0:5000
2024-07-09T12:56:57.820558234Z [INFO][address:127.0.0.1:2122][category:FTPServer][component:Billing] Listening...
2024-07-09T12:56:57.820664707Z [INFO][category:FTPServer][component:Billing] Starting...
```

Figure 71. Successful execution of run_webconsole.sh

In WebConsole it is needed to register the UEs in order to provide them with the hired services and also give them permissions to be allowed in the network. To do this, it is necessary to go to the Free5GC VM console (shown in Figure 72) inside the OpenStack instance list panel.

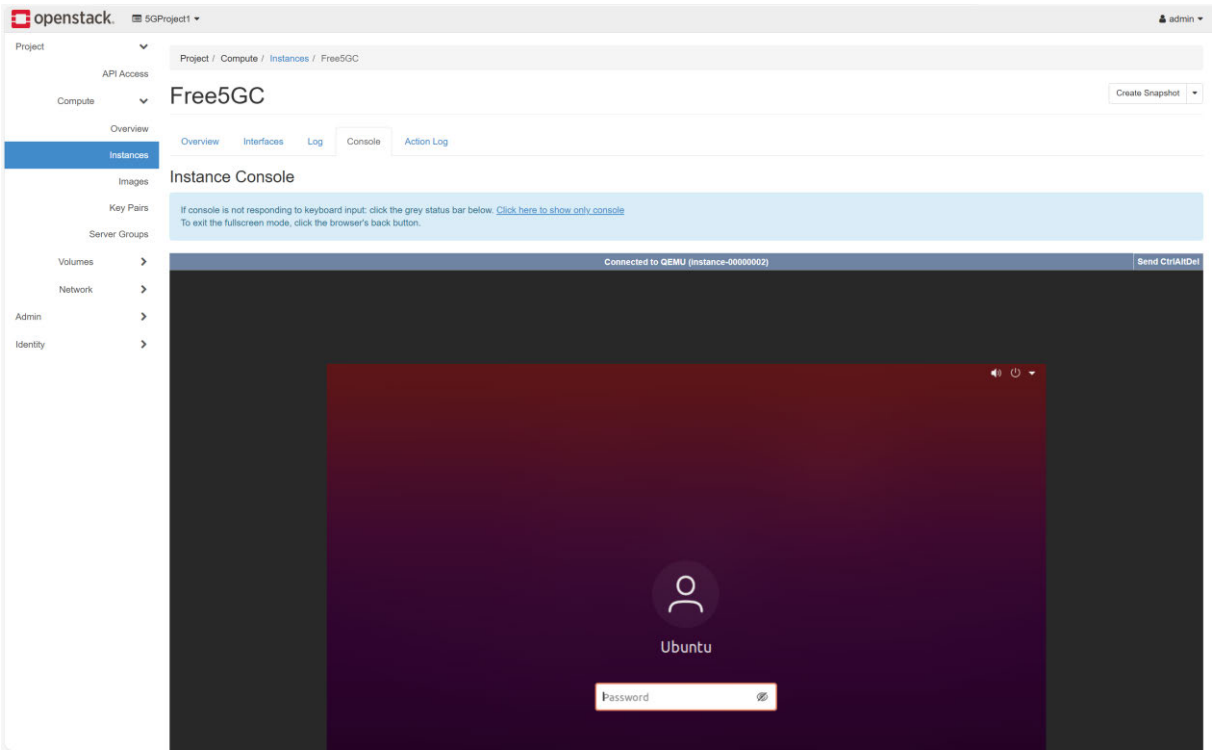

Figure 72. Entering WebConsole in OpenStack

Entering the password '*5gcore*', will automatically redirect to the Free5GC WebConsole URL (Uniform Resource Locator) and will request the credentials as shown in Figure 73. The credentials for the WebConsole are the following:

- User: admin
- Password: free5gc

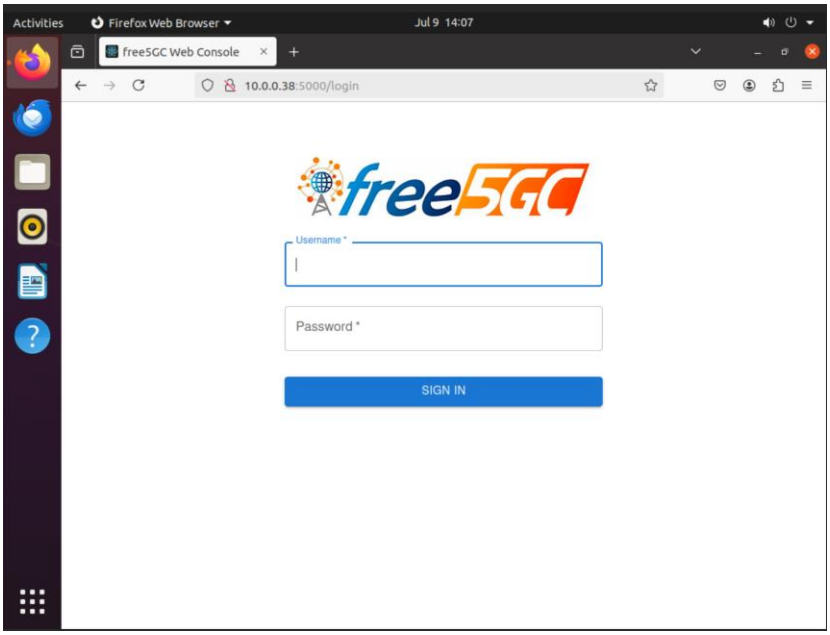

Figure 73. Webconsole login page

We must create four Subscribers in order to register UE1, UE2, UE3 and UE4 as clients. The only parameter that must be changed is the IMSI that must follow the UE-IMSI mapping found in Table 3.

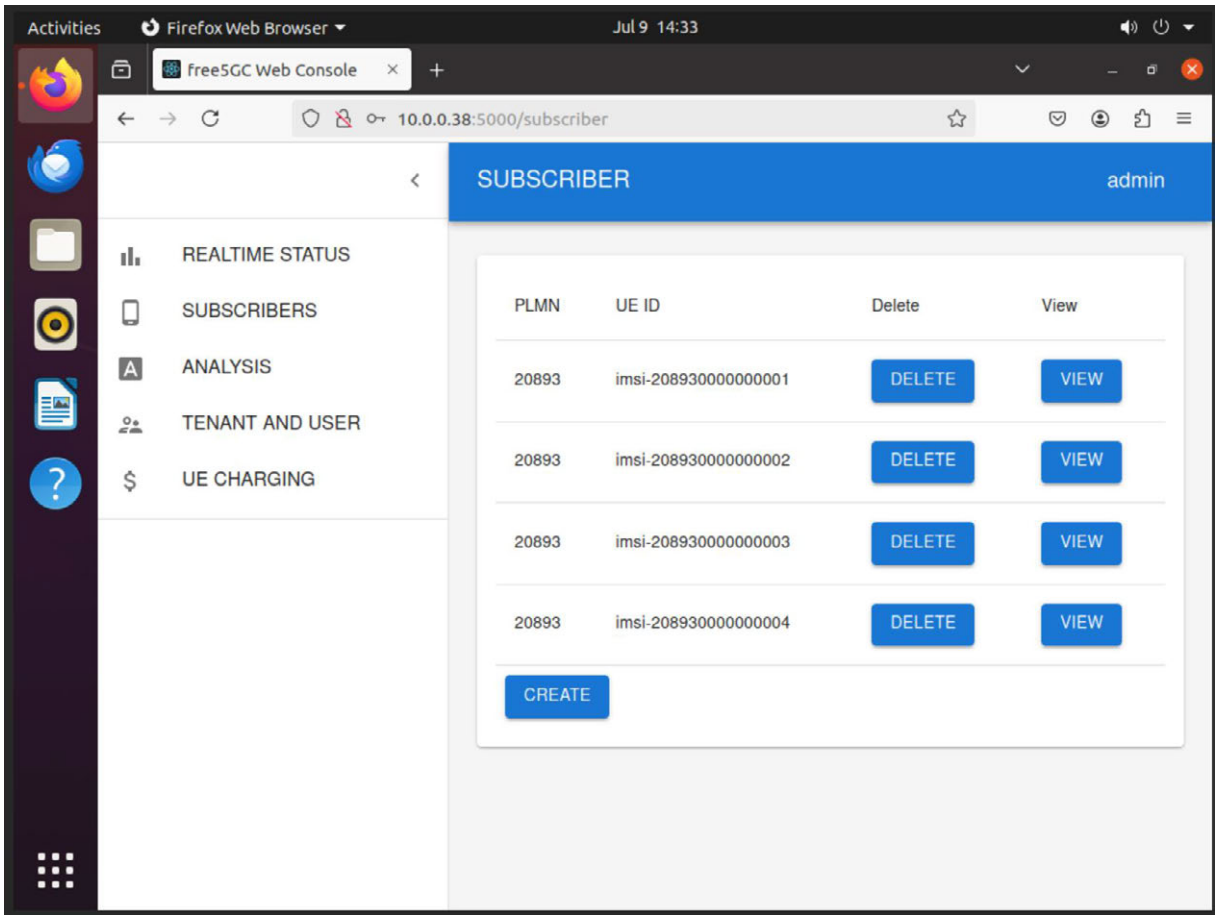

Figure 74. Webconsole capture of registered 5G Subscribers

As can be seen in Figure 74, the four UEs deployed are registered and should have access to the corresponding 5G services. Now we can continue with the test by initiating the gNB1 and gNB2.

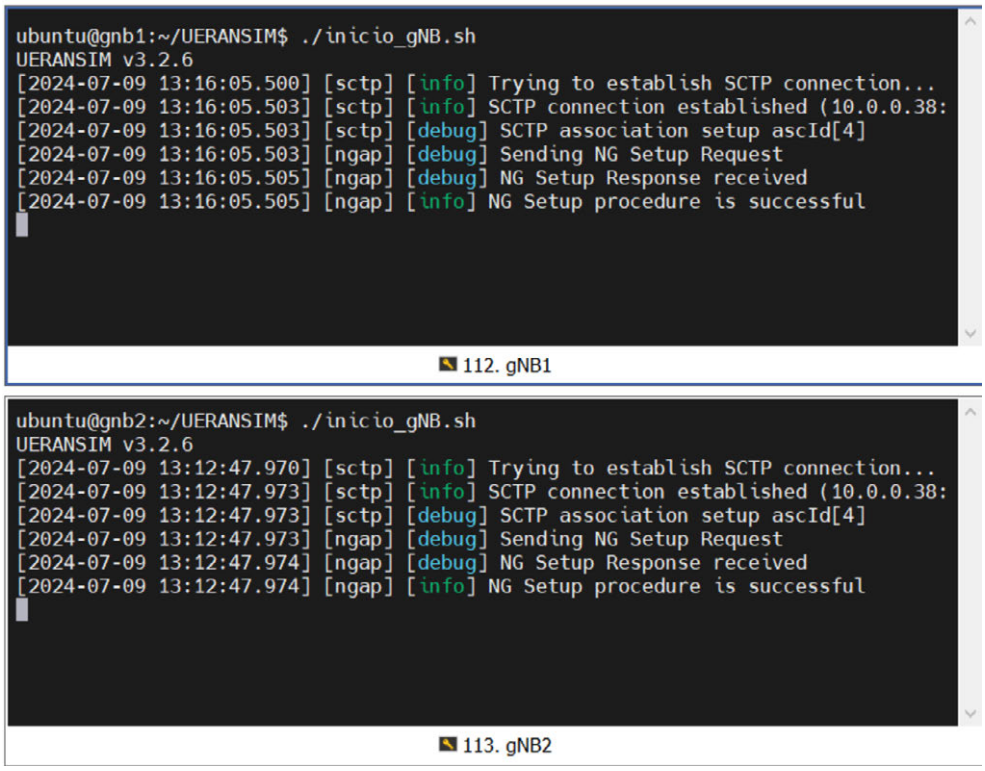

Figure 75. Successful execution of inicio_gNB.sh script in gNB1 and gNB2

In Figure 75 the gNBs establish an SCTP (Stream Control Transmission Protocol) connection with AMF, which means that they can share messages between each other. After that, the UEs can connect to the network.

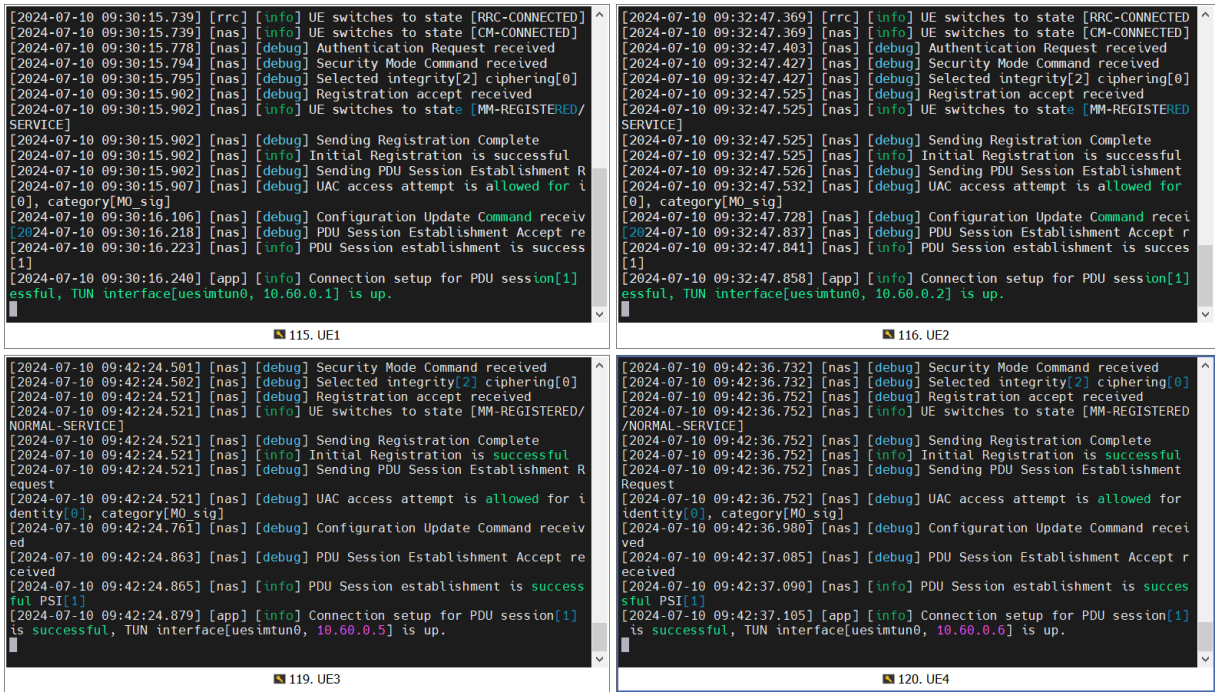

Figure 76. Successful execution of inicio_UE.sh script in UE1, UE2, UE3 and UE4

In Figure 76, it can be observed that the tunnel interface '*uesimtun0*' is up with an IP in the 10.60.0.1-10.0.0.254 range. This means that we have established the corresponding tunnel between UE-gNB-UPF-AMF in order to provide the UEs with internet connection. In Figure 77 can be observed the ping tests that have been made to different internet domains (google.com, facebook.com, Instagram.com and youtube.com), proving that each UE is provided with internet connection over 5G service.

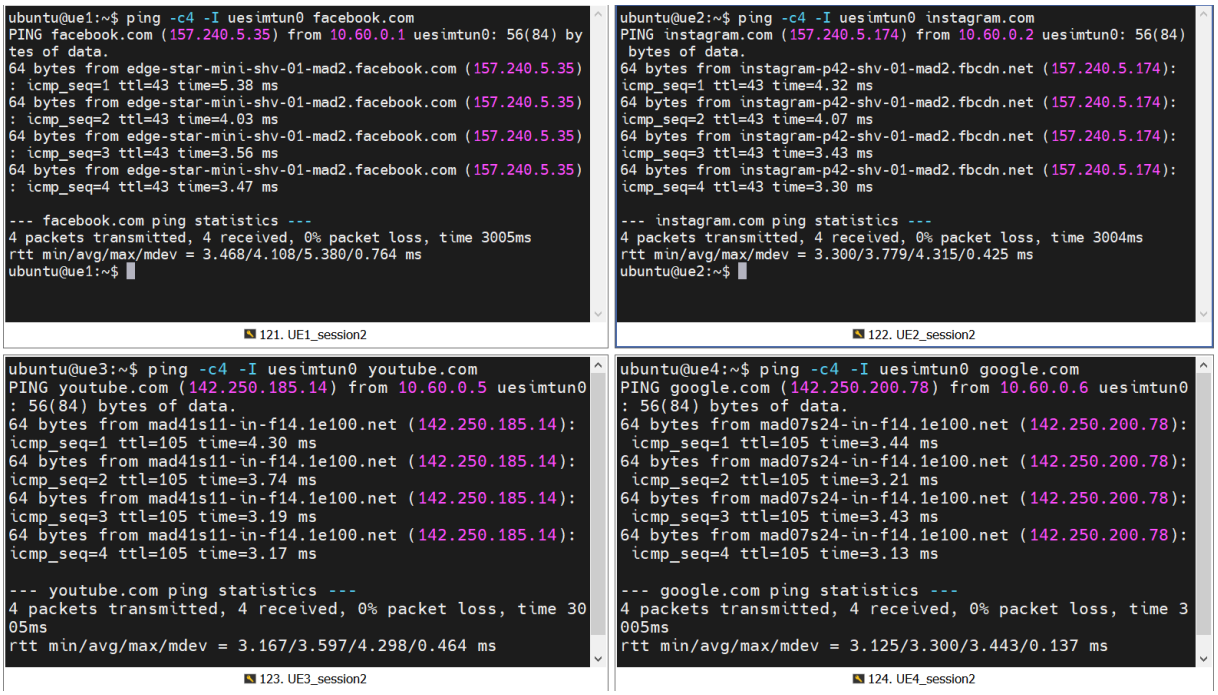

Figure 77. UEs internet connection test through GTP tunnel interface

As has been demonstrated, the 5G System described in this project works as expected providing 5G services to UEs through the OpenStack virtualization service.

6. Budget

This 5G core project has stuck to the following budget established the 12th of June 2024:

Table 4. Budget breakdown

Concept	Reference	Price
ASUS ESC4000-E10 (ESC4000-E10)	[29]	5,537.92€
Ethernet cable set	[30]	9.89€
Crossover cable	[31]	10.95€
Mini PC (Server Client)	[32]	765.83€
Monitor (Server Client Monitor)	[33]	88.99€
Salary of Junior Engineer with 340 working hours	[34]	5312.50€
	Total amount:	**6.413,58€**

It basically comprises both the equipment and the work time of the engineer. Regarding the equipment, it includes the server in which the virtualization system has been established together with the 5G emulation network, the server client equipment through which the server has been configured and accessed remotely, and the cables to connect internally server and client. With respect to the work time, it sums up to 340 hours, as was stated in the project plan, being the cost of one hour equal to 15.625 euros. This amount was calculated considering the salary of a Telecommunication Junior Engineer as 24.000€ per year (as per reference [9]) and that the working hours is set as full time (8 hours per day) without taking into account weekends.

7. Project impact

The virtualized 5G System solution proposed in this project could have a powerful impact in different areas (depicted in Figure 78), as for instance the quality of education in regard with 5G Systems, as it is difficult to deploy complete 5G solutions in a non-enterprise environment without spending lots of money. This solution could provide accessibility and affordability to 5G systems, contributing to the SDG (Sustainable Development Goals) 4 [35] (Quality education) and 10 [36] (Reduced inequalities).

Another field in which this project could be beneficial is the creation of new 5G Infrastructures (contributes to the SDG 9 [37]) as this platform impulses innovation for being an easy and quick tool to deploy 5G Systems. In this regard, this project is unique and could be marked as a reference model for the deployment of new types of solutions in mobile network communications here in Spain or abroad.

Finally, it is important to mention that virtualization solutions for 5G systems offer a green impact in test environments, as it is not necessary to have real gNBs running and consuming energy for testing 5GC networks. Simulation of 5G Systems is a good manner in which to reduce the CO_2 footprint while contributing to achieve the SDG goal 13 [38] (Climate action).

Figure 78. Project contribution for SDG goals (images from [39])

8. Conclusions and future works

In this chapter, we will see the final conclusions and recommendations for future works regarding this project.

8.1. Conclusions

This project started by defining the scope and requirements for the deployment of a 5G virtualized solution while performing the optimal configuration of the physical network.

While the physical configuration was in process, the research to find solutions started. This ends with the implementation of OpenStack as the desired solution to the problem. Although it was hard to understand how to deploy this tool, the goal was achieved with DevStack.

Once OpenStack was ready to work the first problem was that the VMs that had been created previously using Virtual Box were not compatible with Open Stack as it needed a cloud OS image instead. To solve this problem, it was decided to create the 5G System from a ready-to-use Ubuntu Cloud image which could not be accessed via Horizon console panel, as the password was randomized.

After further investigation, it was discovered that instances should be accessed via SSH from the external network. Once the goal was achieved and the instances could be accessed, the next step was to deploy 5GC and provide the proper functionality, which in fact was the easiest part.

Free5GC was configured, and then the UERANSIM VMs that were just snapshots of the same configured machine with some changes in the configuration. This demonstrates that the manageability of OpenStack is beyond the initial expectations.

As a conclusion, it is remarkable that even though the learning curve for performing this project is high, the results in general have been very satisfactory, not only because this can be extremely useful and cheap solution for small operators or research centers (among others) but also due to that it provides powerful technological tools for managing the upcoming future network designs, such as 6G Mobile Networks. For these reasons, this project could be considered a complete success in the field of global telecommunications engineering.

8.2. Future works

This project could be improved in different ways. The following is a summary of the most important items that should be considered as future works.

Physical Deployment:

- Following the model architecture from Figure 17, the PHYSICAL CLIENT must be always powered on in order to work with EXTERNAL CLIENT, this method wastes energy unnecessarily when REMOTE CLIENT is not using the RDP connection. Maybe a better

option is to set Wake on LAN (Local Area Network) system to power on the PHYSICAL CLIENT only when the EXTERNAL CLIENT is trying to access to it.

- Following the reference [40] from OpenStack documentation, we can set the Asus server Link Local 2 interface from Figure 17 to provide external (from Asus server) access to the OpenStack network which could be helpful for developing new future network services that need to reach OpenStack VMs without being physically connected to ASUS ESC4000-E10 server

OpenStack Deployment:

- After ASUS ESC4000-E10 reboots, the OpenStack cloud on some occasions loses connectivity and some components fail, unless br-ex interface has the proper IP. We consider that it could be worth to try using Ansible to deploy a full version of OpenStack or RHEL (Red Hat Enterprise Linux)/CentOS with Packstack as shown in the course of reference [41].

5G System:

- Following the Free5GC documentation in [42], it could be convenient to establish an NFV MANO (Management & Orchestration) Architecture using OpenStack and Tacker to offer management and orchestration services to our 5GC.
- Configure 5GC to allow physical UEs (e.g. 5G Smartphone) and gNBs (SDRs as well-known as Software-defined radios).
- Test mobility procedures like attach, detach and handover with virtual or physical gNBs and UEs and try to capture 5G mobility management messages defined in [43] with Wireshark or similar traffic sniffer.

9. References

[1] R. S. Shetty, *5G Mobile Core Network: Design, Deployment, Automation, and Testing Strategies*, 1st ed. Berkeley, CA: Apress L. P, 2021.

[2] "ETSI TS GSM 03.40." Accessed: Jul. 02, 2024. [Online]. Available: https://www.etsi.org/deliver/etsi_gts/03/0340/05.03.00_60/gsmts_0340v050300p.pdf

[3] H. Hodara and E. Skaljo, "From 1G to 5G," *Fiber Integr. Opt.*, vol. 40, pp. 1–99, Dec. 2021, doi: 10.1080/01468030.2021.1919358.

[4] "3GPP – The Mobile Broadband Standard," 3GPP. Accessed: Jun. 03, 2024. [Online]. Available: https://www.3gpp.org/

[5] "Release 1999," 3GPP. Accessed: Jul. 02, 2024. [Online]. Available: https://www.3gpp.org/specifications-technologies/releases/release-1999

[6] "3GPP Portal > Home." Accessed: Jul. 02, 2024. [Online]. Available: https://portal.3gpp.org/#/55934-releases

[7] M. Sauter, *From GSM to LTE-Advanced Pro And 5G : an introduction to mobile networks and mobile broadband*, Fourth edition. 2021.

[8] "Ericsson Mobility Report November 2012." Accessed: Jul. 12, 2024. [Online]. Available: https://www.ericsson.com/4a9882/assets/local/reports-papers/mobility-report/documents/2012/ericsson-mobility-report-november-2012-rev-b.pdf

[9] "Ericsson Mobility Report June 2024," 2024.

[10] "TS 123.501." Accessed: Jul. 05, 2024. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/15.02.00_60/ts_123501v150200p.pdf

[11] V. Dakic, *Mastering KVM virtualization : design expert data center virtualization solutions with the power of Linux KVM*, Second edition. 2020.

[12] "KVM." Accessed: Jul. 03, 2024. [Online]. Available: https://www.linux-kvm.org/page/Main_Page

[13] "QEMU." Accessed: Jul. 03, 2024. [Online]. Available: https://www.qemu.org/

[14] "Open Source Cloud Computing Platform Software," OpenStack. Accessed: Jul. 03, 2024. [Online]. Available: https://www.openstack.org/software/

[15] "DevStack — DevStack documentation." Accessed: Jul. 03, 2024. [Online]. Available: https://docs.openstack.org/devstack/latest/

[16] "Magma – Linux Foundation Project." Accessed: Jul. 05, 2024. [Online]. Available: https://magmacore.org/

[17] "Open5GS." Accessed: Jul. 05, 2024. [Online]. Available: https://open5gs.org/

[18] F. Okuyucu, "bubblecounter/Internship-5GCN." Dec. 06, 2023. Accessed: Jul. 05, 2024. [Online]. Available: https://github.com/bubblecounter/Internship-5GCN

[19] "5G CORE NETWORK – OpenAirInterface." Accessed: Jul. 05, 2024. [Online]. Available: https://openairinterface.org/oai-5g-core-network-project/

[20] "O-RAN Software." Accessed: Jul. 11, 2024. [Online]. Available: https://www.o-ran.org/software

[21] "Features - free5GC." Accessed: Jul. 05, 2024. [Online]. Available: https://free5gc.org/guide/features/

[22] "aligungr/UERANSIM: Open source 5G UE and RAN (gNodeB) implementation." Accessed: Jul. 05, 2024. [Online]. Available: https://github.com/aligungr/UERANSIM?tab=readme-ov-file

[23] "ESC4000-E10 | ASUS Servers and Workstations." Accessed: Jul. 02, 2024. [Online]. Available: https://servers.asus.com/products/servers/gpu-servers/ESC4000-E10

[24] “How to use Remote Desktop - Microsoft Support.” Accessed: Jul. 04, 2024. [Online]. Available: https://support.microsoft.com/en-us/windows/how-to-use-remote-desktop-5fe128d5-8fb1-7a23-3b8a-41e636865e8c

[25] Mobatek, “MobaXterm free Xserver and tabbed SSH client for Windows.” Accessed: Jul. 04, 2024. [Online]. Available: https://mobaxterm.mobatek.net/

[26] “OpenStackClient — OpenStack Command Line Client 6.7.0.dev83 documentation.” Accessed: Jul. 08, 2024. [Online]. Available: https://docs.openstack.org/python-openstackclient/latest/

[27] “OpenStack Docs: OpenStack command-line client.” Accessed: Jul. 09, 2024. [Online]. Available: https://docs.openstack.org/mitaka/cli-reference/openstack.html

[28] “Wi-Fi Alliance.” Accessed: Jul. 11, 2024. [Online]. Available: https://www.wi-fi.org/

[29] “ASUS ESC4000-E10 (90SF01B3-M00500) - Configure Online.” Accessed: Jun. 12, 2024. [Online]. Available: https://www.broadberry.co.uk/3rd-gen-xeon-scalable-processor-asus-servers/asus-esc4000-e10-90sf01b3-m00500

[30] “Max Connection Pack 5 Cables Ethernet Cat.6 RJ45 24AWG 1m + 15 Bridas | PcComponentes.com.” Accessed: Jun. 12, 2024. [Online]. Available: https://www.pccomponentes.com/max-connection-pack-5-cables-ethernet-cat6-rj45-24awg-1m-15-bridas

[31] “3m RJ45 CROSSOVER Cable Cat5e Red Ethernet Lead X sobre CROSS WIRED - GRIS,” Fruugo. Accessed: Jun. 12, 2024. [Online]. Available: https://www.fruugo.es/3m-rj45-crossover-cable-cat5e-red-ethernet-lead-x-sobre-cross-wired-gris/p-291633202-651157760?language=es&ac=bing&msclkid=725fb1c427d1172edf12d7adadb6649e&utm_source=bing&utm_medium=cpc&utm_campaign=All_ES&utm_term=4574724289319414&utm_content=ES

[32] “Leotec Mini Pc Intel Core i7-1360P/16GB/1TB SSD | PcComponentes.com.” Accessed: Jun. 12, 2024. [Online]. Available: https://www.pccomponentes.com/leotec-mini-pc-intel-core-i7-1360p-16gb-1tb-ssd

[33] “LG 25MS500-B 24.5" LED IPS FullHD 100Hz | PcComponentes.com.” Accessed: Jun. 12, 2024. [Online]. Available: https://www.pccomponentes.com/lg-25ms500-b-245-led-ips-fullhd-100hz

[34] “Sueldo: Ingeniero De Telecomunicaciones en España 2024,” Glassdoor. Accessed: Jun. 20, 2024. [Online]. Available: https://www.glassdoor.es/Sueldos/ingeniero-de-telecomunicaciones-sueldo-SRCH_KO0,31.htm

[35] Copyright © United Nations, “Education,” United Nations Sustainable Development. Accessed: Jul. 06, 2024. [Online]. Available: https://www.un.org/sustainabledevelopment/education/

[36] Copyright © United Nations, “Reduce inequality within and among countries,” United Nations Sustainable Development. Accessed: Jul. 06, 2024. [Online]. Available: https://www.un.org/sustainabledevelopment/inequality/

[37] Copyright © United Nations, “Infrastructure and Industrialization,” United Nations Sustainable Development. Accessed: Jul. 06, 2024. [Online]. Available: https://www.un.org/sustainabledevelopment/infrastructure-industrialization/

[38] Copyright © United Nations, “Climate Change,” United Nations Sustainable Development. Accessed: Jul. 06, 2024. [Online]. Available: https://www.un.org/sustainabledevelopment/climate-change/

[39] R. Neshovski, “Home,” United Nations Sustainable Development. Accessed: Jul. 11, 2024. [Online]. Available: https://www.un.org/sustainabledevelopment/

[40] “DevStack Networking — DevStack documentation.” Accessed: Jul. 06, 2024. [Online]. Available: https://docs.openstack.org/devstack/latest/networking.html

[41] “OpenStack Certification, 2/e.” Accessed: Jul. 06, 2024. [Online]. Available: https://learning.oreilly.com/course/openstack-certification-2-e/9780134836959/

[42] B. Chatras, U. S. Tsang Kwong, and N. Bihannic, “NFV enabling network slicing for 5G,” in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, Paris: IEEE, Mar. 2017, pp. 219–225. doi: 10.1109/ICIN.2017.7899415.

[43] “TS 124.501.” Accessed: Jul. 10, 2024. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/124500_124599/124501/15.01.00_60/ts_124501v150100p.pdf

[44] “Get Ubuntu | Download,” Ubuntu. Accessed: Jun. 04, 2024. [Online]. Available: https://ubuntu.com/download

[45] “balenaEtcher - Flash OS images to SD cards & USB drives.” Accessed: Jun. 04, 2024. [Online]. Available: https://etcher.balena.io/

[46] “Get images — Virtual Machine Image Guide documentation.” Accessed: Jun. 20, 2024. [Online]. Available: https://docs.openstack.org/image-guide/obtain-images.html#

[47] “Create images manually — Virtual Machine Image Guide documentation.” Accessed: Jun. 20, 2024. [Online]. Available: https://docs.openstack.org/image-guide/create-images-manually.html

Annex 1. Access server dashboard

To access the server dashboard it is necessary to connect to PHYSICAL CLIENT which is a Windows machine. Once logged in this machine open a web browser and open and enter the following URL: https://192.168.1.2/

After accessing this web, the login page will be shown (see Figure 79).

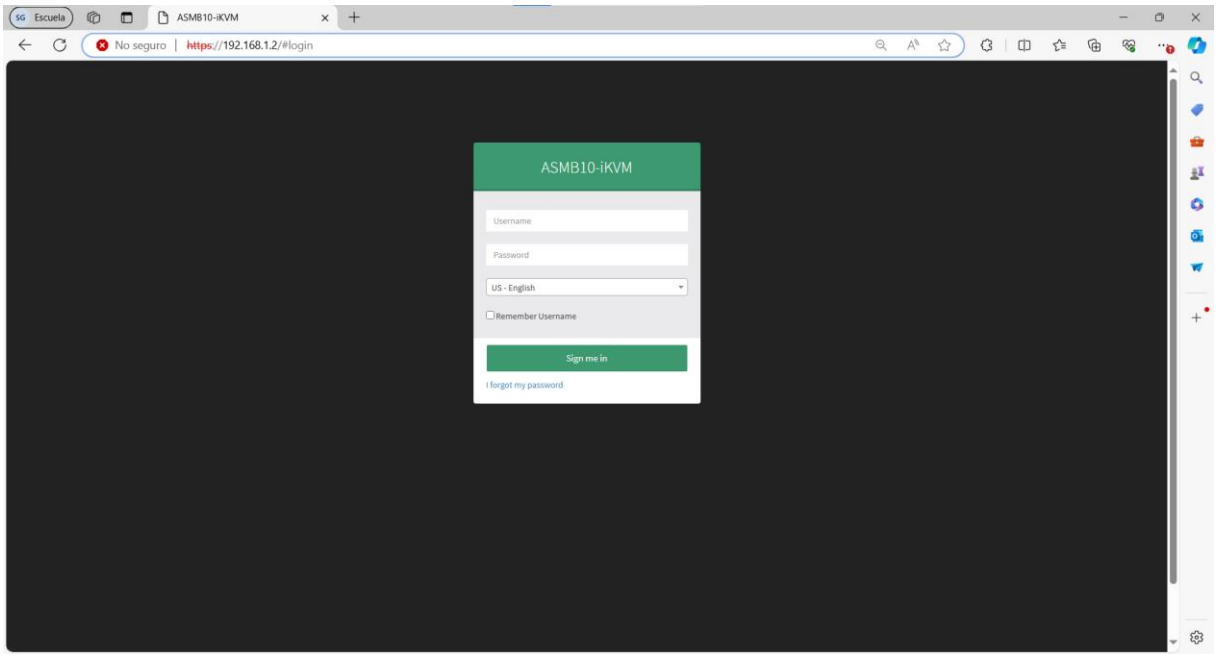

Figure 79. Screenshot of ASUS ASMB10-iKVM login webpage

Enter the credentials and when the "Sign me in" button has been clicked it automatically redirects to Asus ASMB10-iKVM dashboard (see Figure 80) in which shows the status of the server in real time and lots of parameters can be configured, as for instance the speed of fans, network interfaces, etc.

This dashboard has unique special features that allow BIOS control remotely. This will require the same credentials used to enter in previous login page (Figure 79).

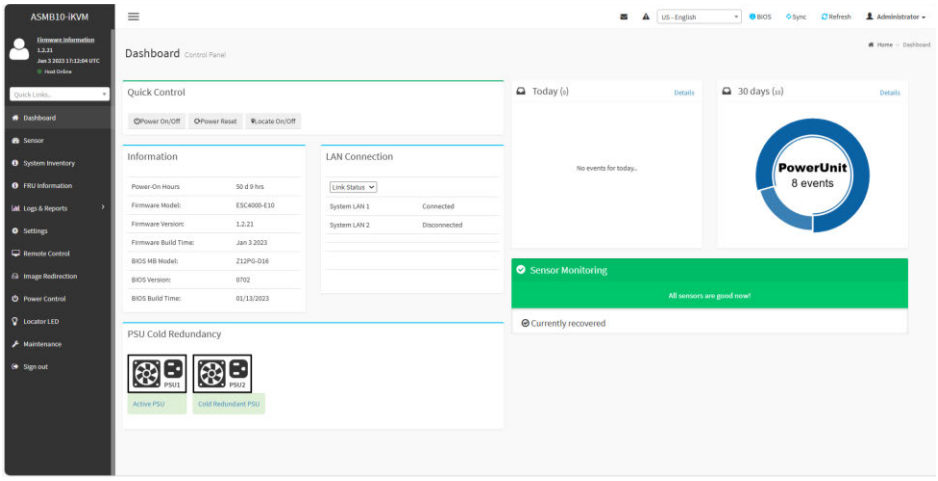

Figure 80. Screenshot of ASUS ASMB10-iKVM dashboard

Annex 2. Server configuration

To configure ASUS ESC4000-E10 server, first it is needed to install the operating system, in this case we proceed to create a bootable USB (Universal Serial Bus) through the .iso file downloaded from [44]. This iso file contains the image of Ubuntu Server 22.04.4 LTS (Jammy Jellyfish) which will be flashed into an USB device via BalenaEtcher app [45]. Once we have the USB ready, it is needed to enter the BIOS settings and change boot options to boot the operating system with the bootable USB device created previously.

Once the key and language settings have been configured, it is time to set a fixed IP for the server and provide internet access. This is made through the configuration of both the default gateway IP address and at least one DNS IP address (as seen in Figure 81).

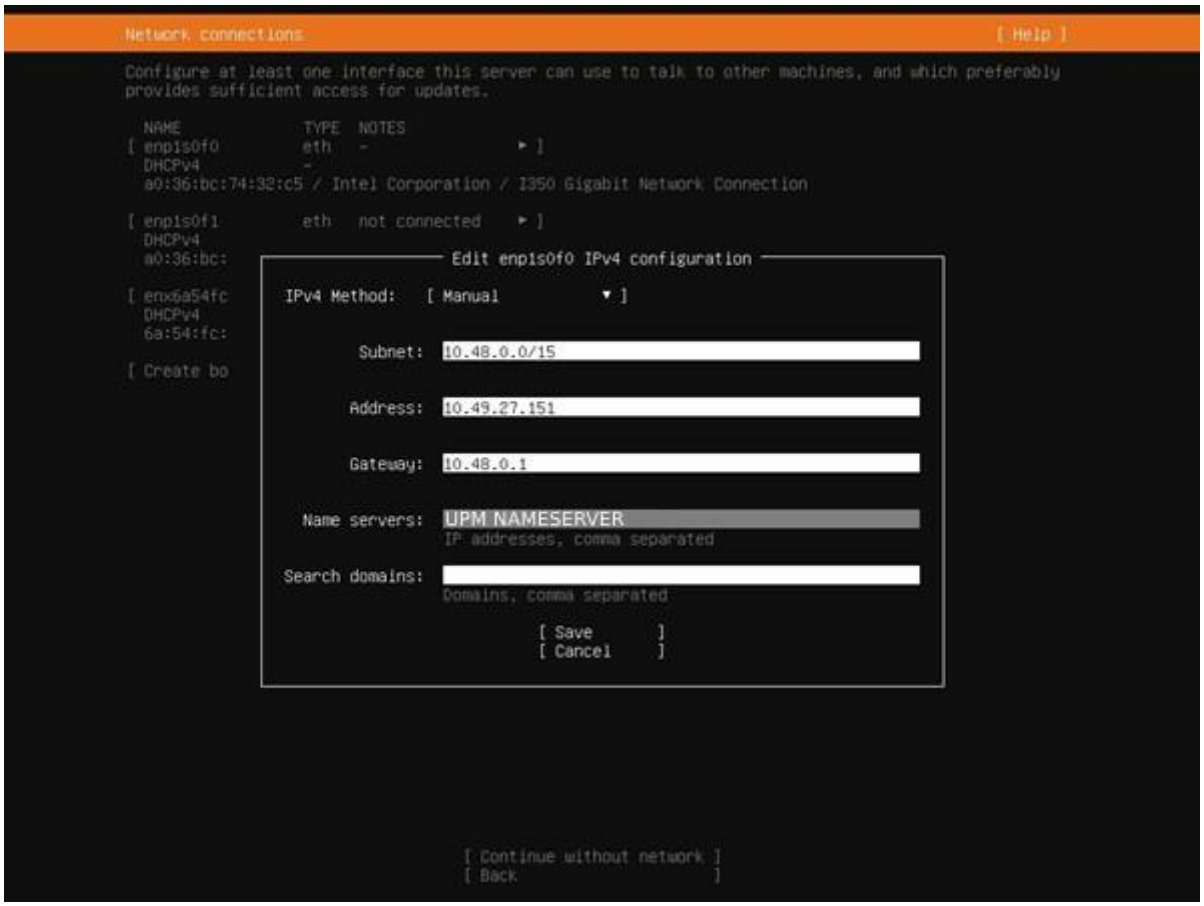

Figure 81. ASUS ESC4000-E10: Ubuntu Server network configuration

After setting the IP address it is necessary to set the storage configuration (see the details in Figure 82 and Figure 83). This is important because OpenStack will use LVM (Logical Volume Manager) volumes to find out the storage available on the host machine. Due to that, we have set the following LVM volume groups lv-0 and ubuntu-lv, the first one is for hosting OpenStack and the second one is created automatically by the installer to handle the ASUS ESC4000-E10 Operating System, in our case Ubuntu LTS 22.04 (Jammy Jellyfish).

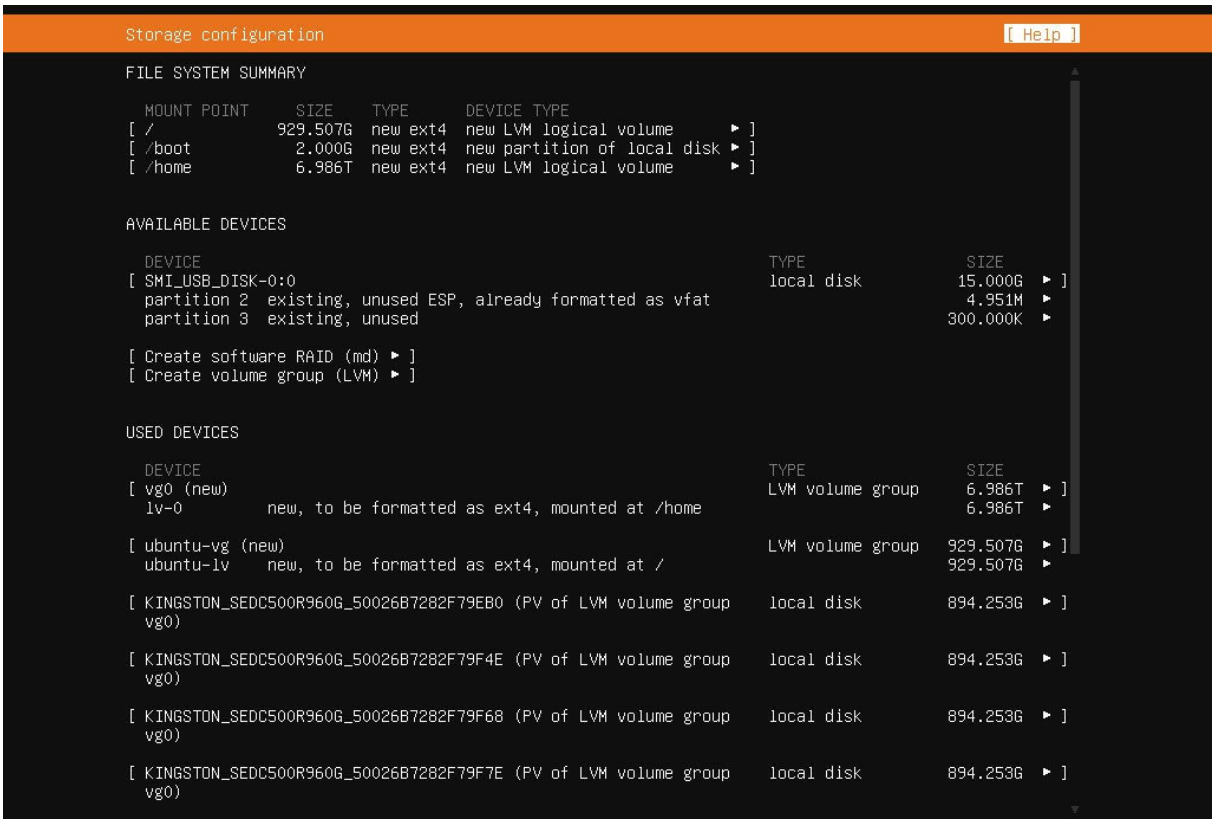

Figure 82. ASUS ESC4000-E10: Ubuntu Server storage configuration (Part 1)

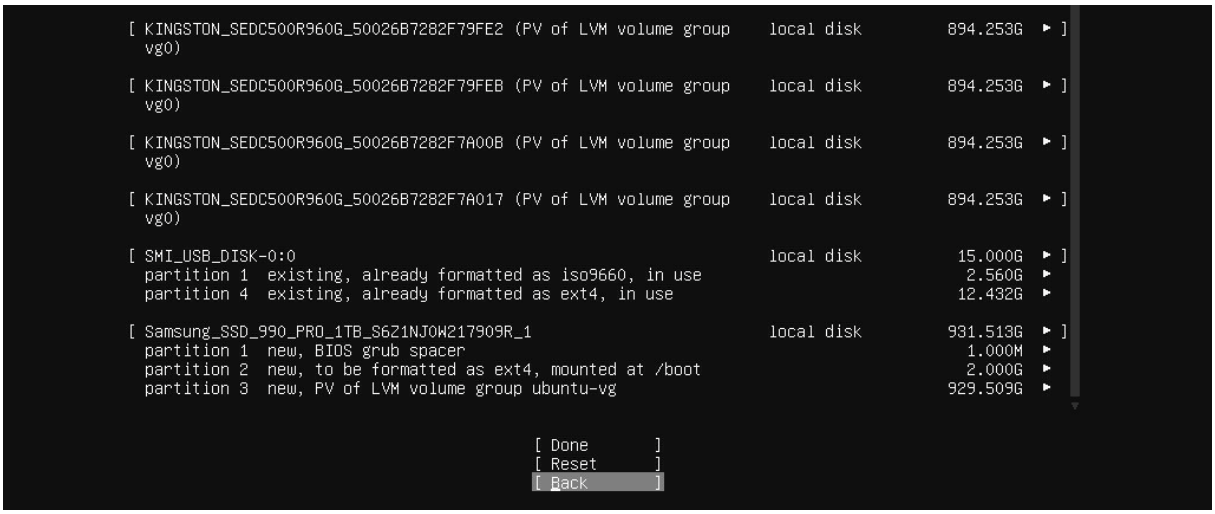

Figure 83. ASUS ESC4000-E10: Ubuntu Server storage configuration (Part 2)

Once partitions and volumes are set, then it is needed to set the name server, the username and password.

The credentials for the ectics user are set in Figure 84.

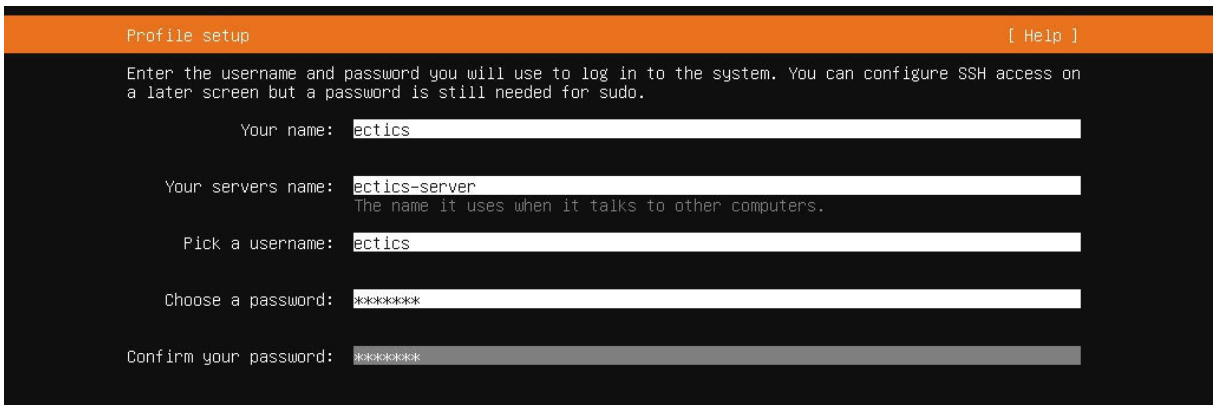

Figure 84. ASUS ESC4000-E10: Ubuntu Server credentials configuration

The last step is to set the SSH server (as shown in Figure 85) in order to provide conectivity with the SSH clients used in this project.

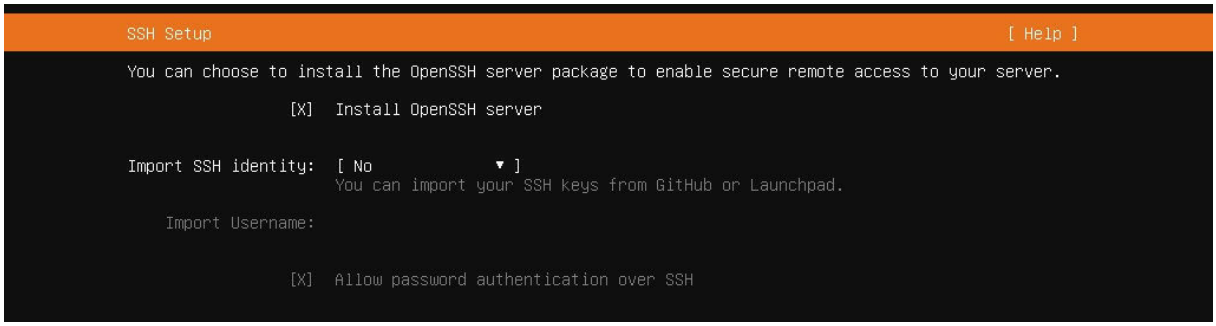

Figure 85. ASUS ESC4000-E10: Ubuntu Server SSH server configuration

Annex 3. How to launch instances in OpenStack

A.1 Launch instance guide

As an initial step towards launching an instance in the OpenStack Horizon web, it is needed to download a pre-prepared cloud image [46] or build a custom image [47].

Once the image is ready, the next move is to enter the OpenStack login page (shown in Figure 86) in PHYSICAL CLIENT by following the next link on any web browser:

http://10.49.27.151/dashboard

In this login page we must write the following credentials:

User: admin

Password: OpenStack33

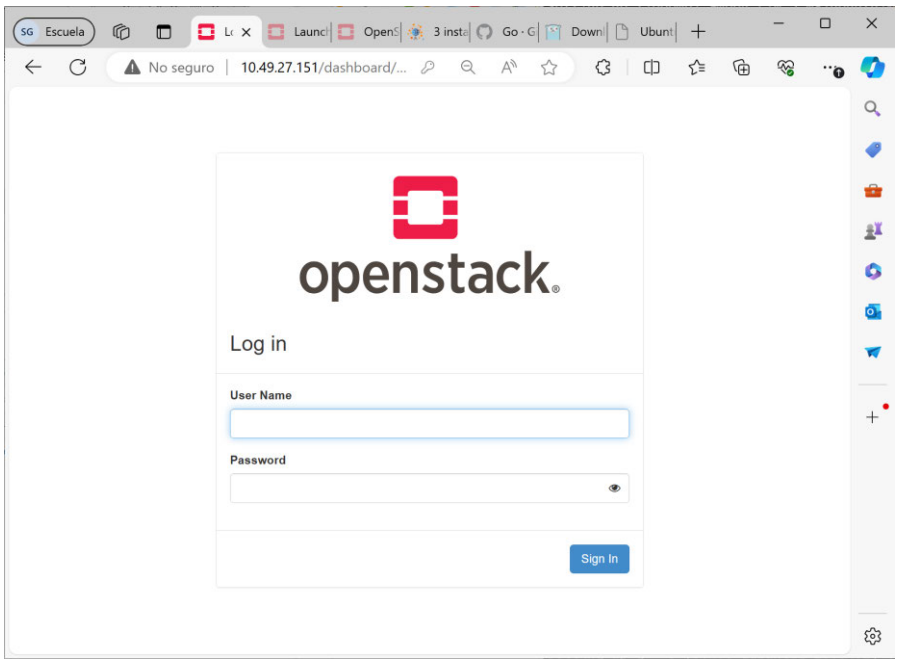

Figure 86. OpenStack GUI: login page

To launch an instance, it is necessary to see the list of available images in Project > Compute > Images (if there are no images refer to section 4.5.1). Figure 87 shows four instances that can be launched.

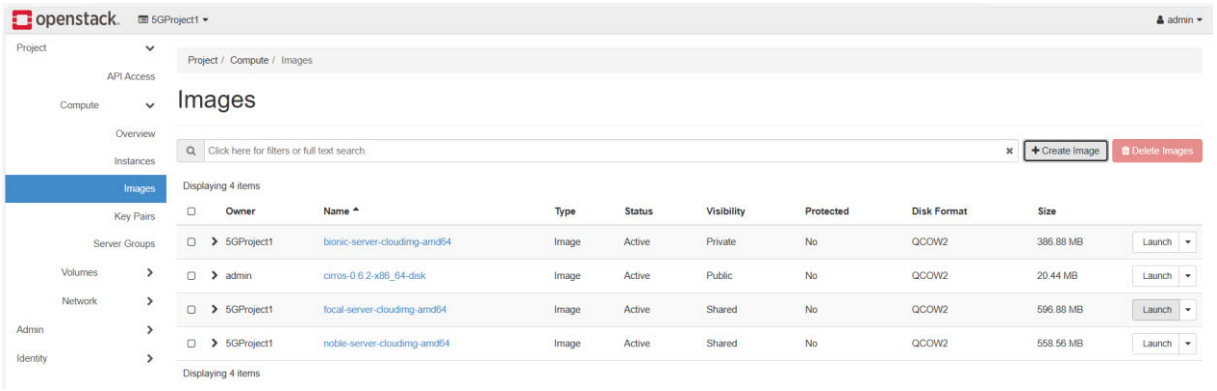

Figure 87. OpenStack GUI: Image dashboard

Clicking on noble-server-cloudimg-amd64 '*Launch*' button, the launch instance (Details) window will appear (this is represented in Figure 88). In this window we will write the instance name and click the '*Next*' button to continue with the process in the Source settings shown in Figure 89, where preferably must be set the '*Delete Volume on Instance Delete*' parameter to Yes in order to avoid lack of storage issues later. In this section, can be set the image that will be allocated to the instance and whether to run it both from a volume and from an image. Volumes in our case have presented some issues, so we will be using Images.

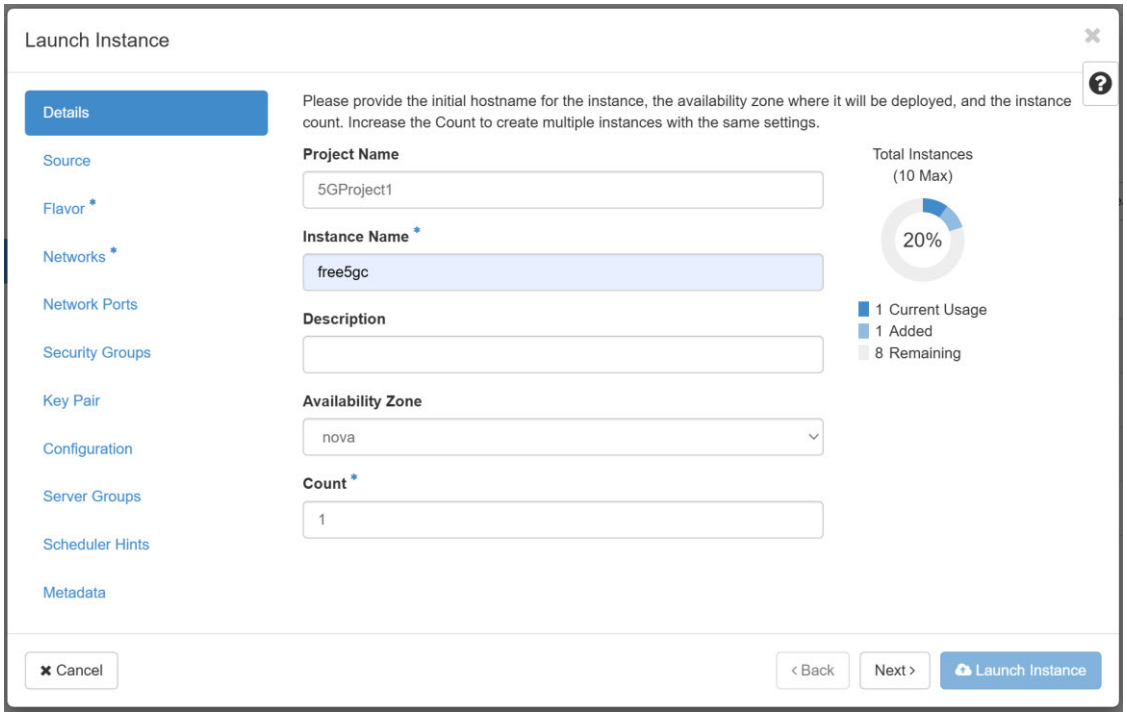

Figure 88. OpenStack GUI: Launch instance - Details

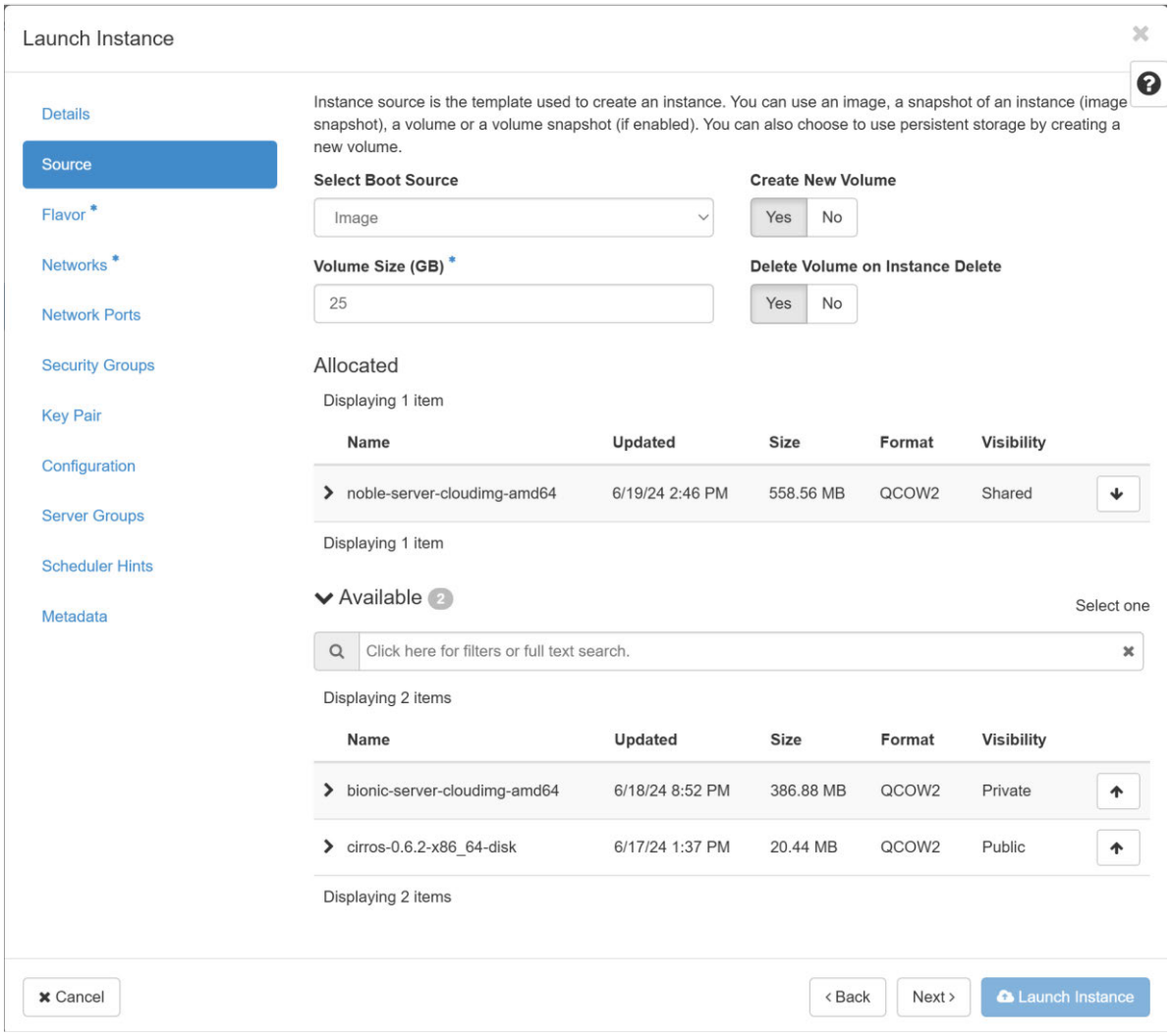

Figure 89. OpenStack GUI: Instance configuration - Source

By clicking *'next'* button again we will move forward to the flavor section (depicted in Figure 90) in which we must allocate one of the following set of resources to the instance.

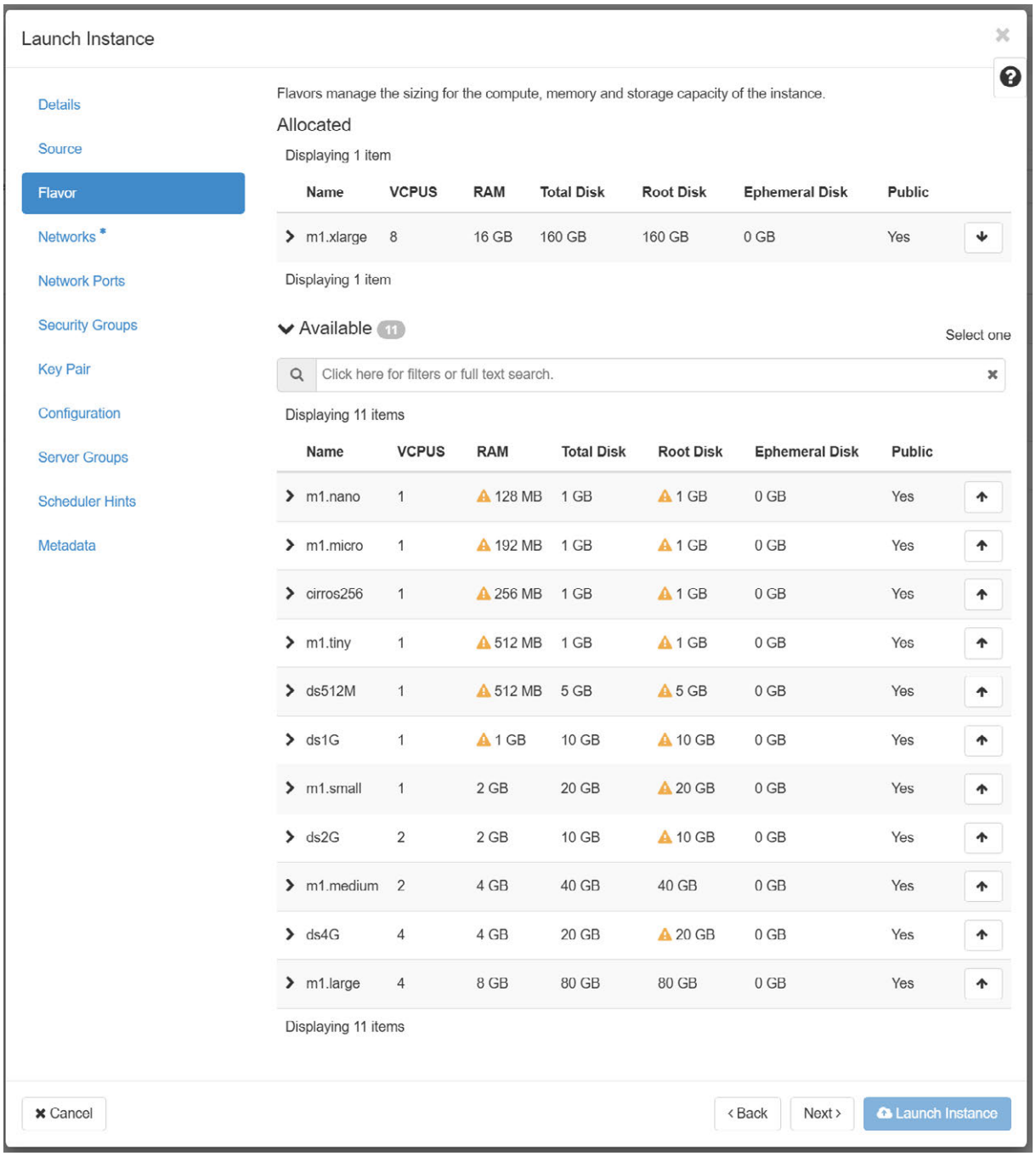

Figure 90. OpenStack GUI: Instance configuration – Caption

Following the same procedure of clicking the *'next'* button we will see the Networks section (Figure 91) in which we could allocate the instance into one or more private networks. This will create automatically a VNIC (Virtual Network Interface Card) with a random IP address in the range established in the pool address set in Figure 37 for that specific network.

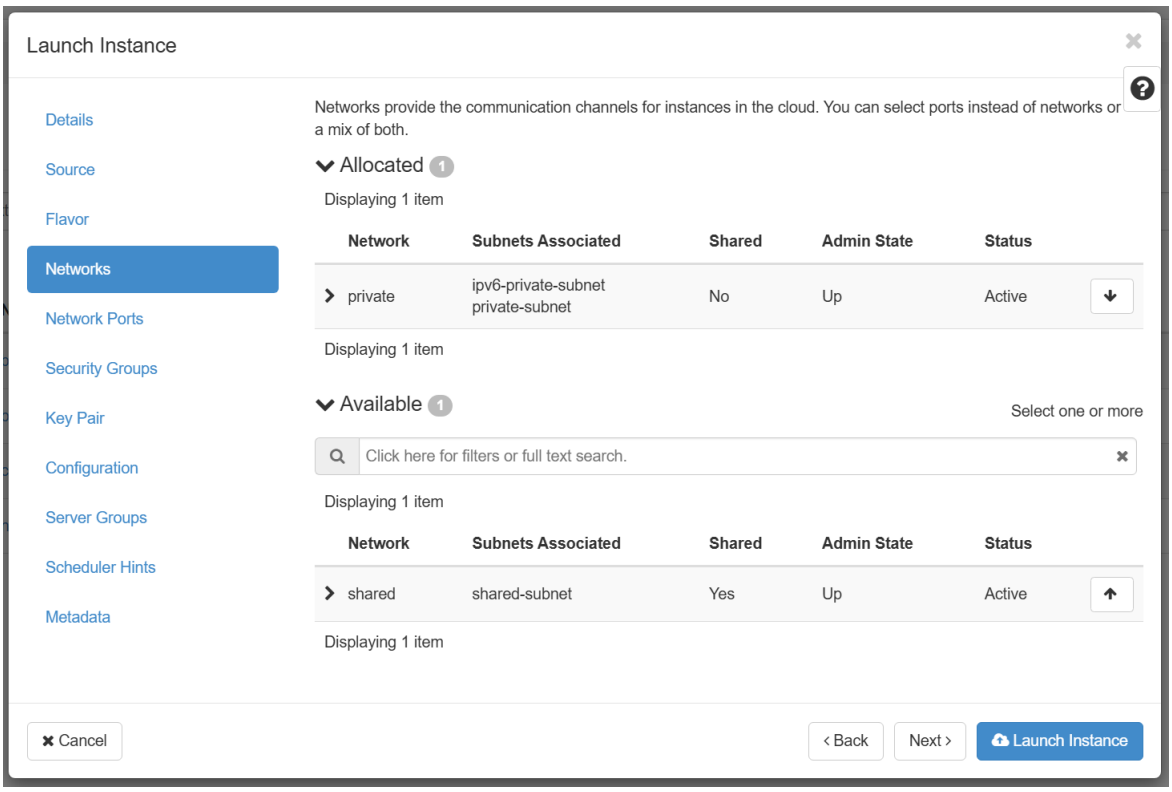

Figure 91. OpenStack GUI: Instance configuration - Networks

The next step is setting the security group for the instance (Figure 92), this is useful in order to allow or deny some traffic like DNS, ICMP, HTTP, etc.

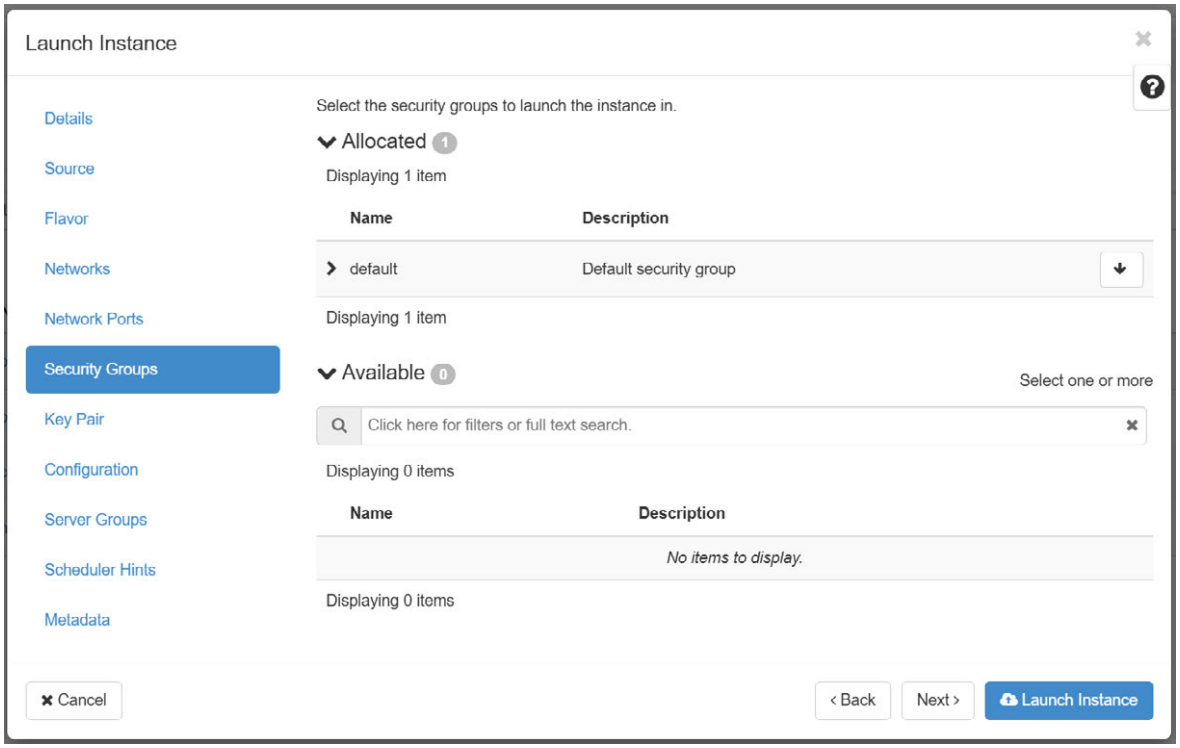

Figure 92. OpenStack GUI: Instance configuration - Security Groups

To allow the machines to be accessible via SSH a Key Pair must be created. In Figure 93 is shown the private key to access the VMs we will be using. We must copy it and create a free5gc.key file in which we will paste the content. This process will be performed in ASUS ESC4000-E10 server which is the SSH gateway that we will be using to access the instances.

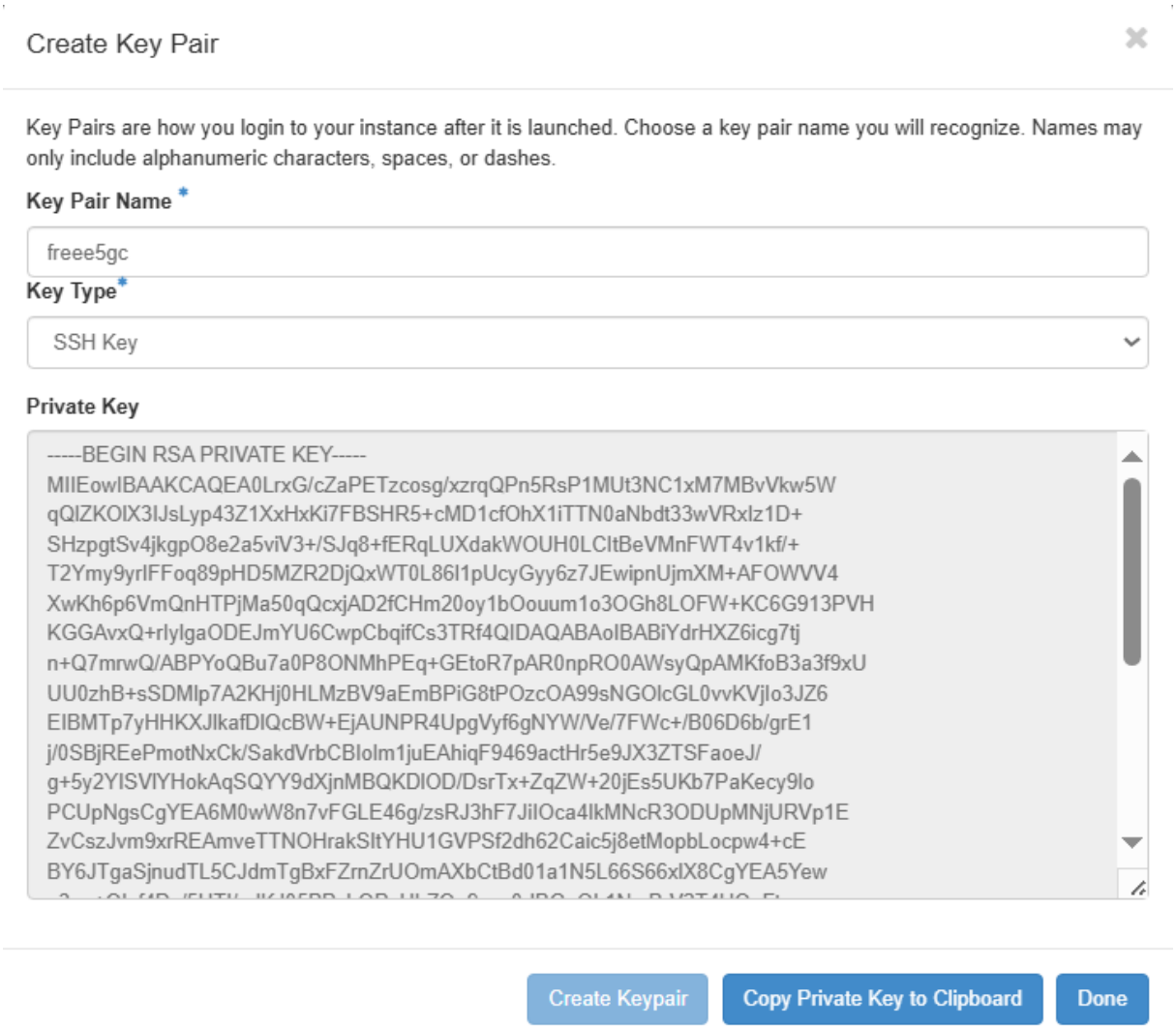

Figure 93. Key Pair creation for SSH

Once the Key Pair is created, we must allocate it to the instance as can be seen in Figure 94.

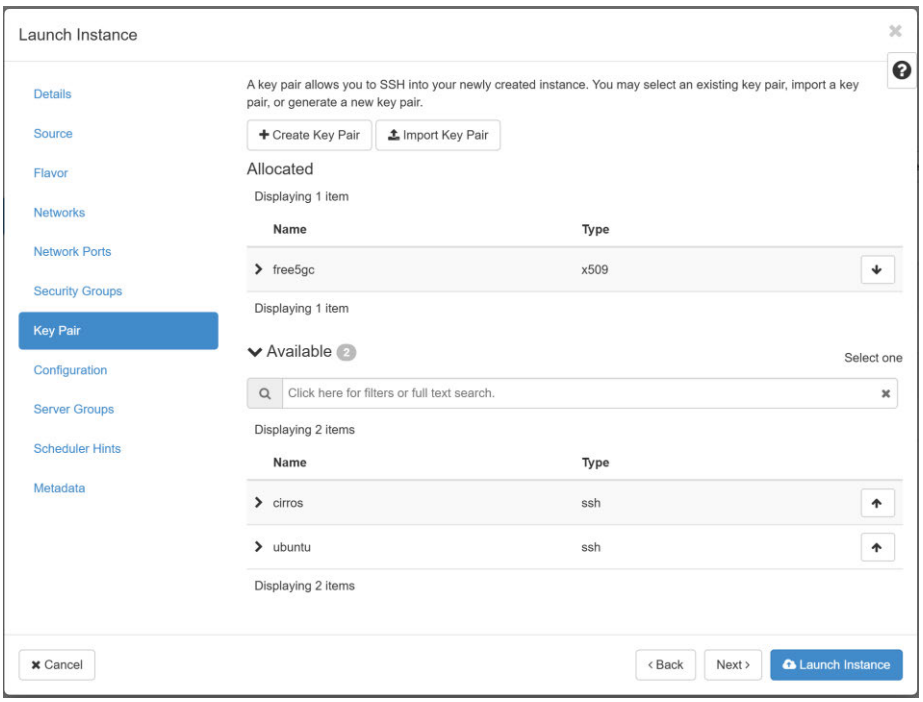

Figure 94. OpenStack GUI: Instance configuration - Key Pair

There are some additional steps configurations but will not be covered in this annex. To launch the instance, we must click '*Launch Instance'* button (in Figure 94) which will start the creation process. If the instance creation process succeeded, it will be shown in the Instance panel dashboard with the '*Status'* property set as Active (as can be seen in Figure 95).

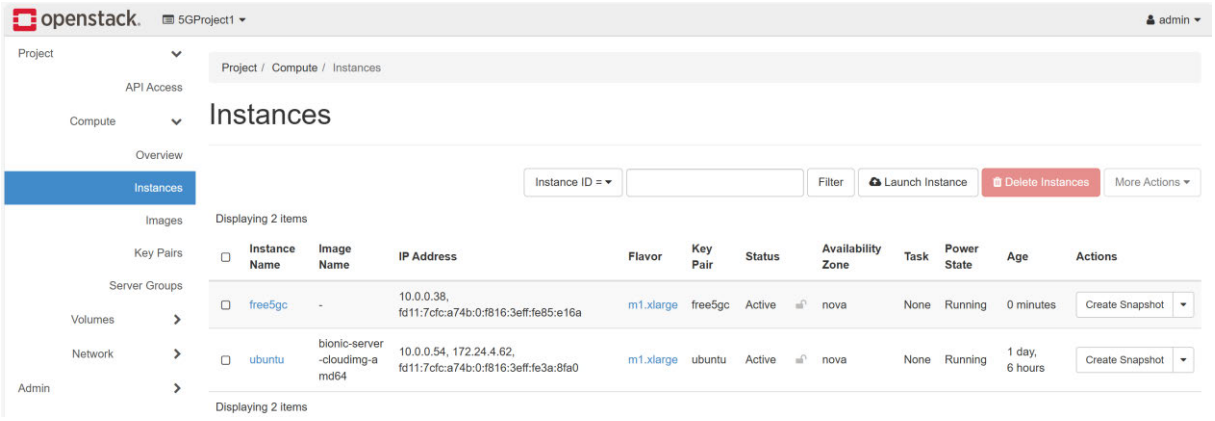

Figure 95. Instances dashboard

The VM deployed in Figure 95, by default has a VNIC assigned to the private network with an IPv4 address equal to 10.0.0.38. To make VMs accessible from ASUS ESC4000-E10 server (which is the SSH gateway we will be using later), we must assign a floating IP to the instance by clicking in '*Associate Floating IP*' button shown in Figure 96. This will open the IP

management panel shown Figure 97 in which we can select or create Floating IPs to be assigned to the 10.0.0.38 interface.

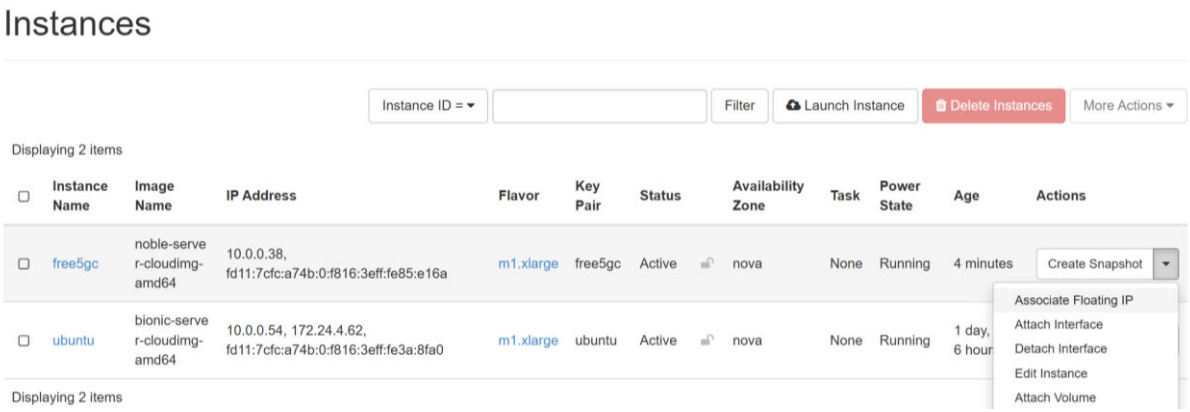

Figure 96. OpenStack GUI: Instance configuration - Associate Floating IP

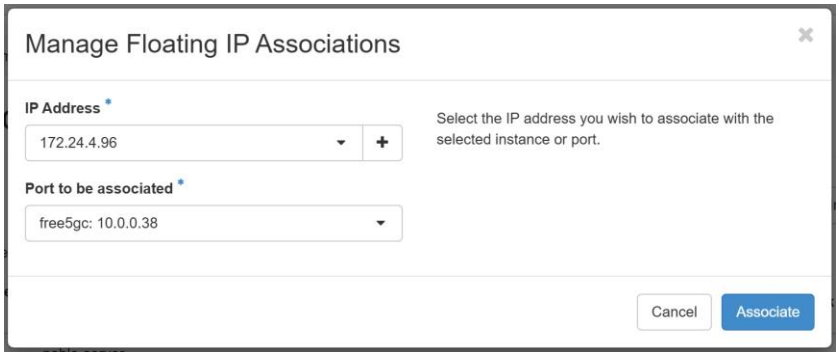

Figure 97. Floating IP management panel

A.2 Access instances with MobaXTerm

In ASUS ESC4000-E10 server command promt, enter the **/home/ectics/openstack-ssh-keys/** folder and generate a free5gc.key file with private key obtained in the keypair creation step mentioned in the previous section. Be sure that the file created has the right permissions by using the next command:

```
$ chmod 600 free5gc.key
```

Then try to connect the remote OpenStack instance by using the Floating IP (172.24.4.96) assigned in Figure 97. This can be done by using the following command:

```
$ ssh -i free5gc.key ubuntu@172.24.4.96
```

If the SSH session is stablished it is recommended to set the virtual machines in MobaXTerm. In order to do that, establish an SSH session connection to ASUS ESC4000-E10 server via MobaXTerm in the PHYSICAL CLIENT (shown in Figure 17). Then, download free5gc.key file by right clicking in FTP tool that Moba app offers (see Figure 98).

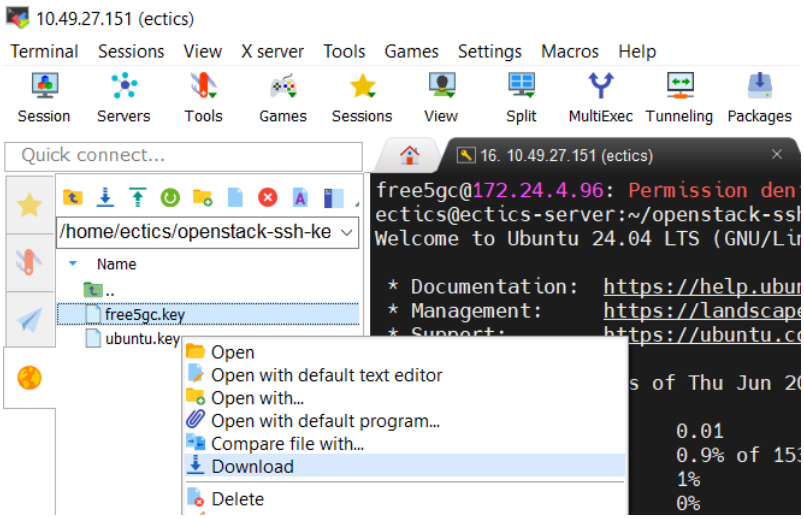

Figure 98. Download Key Pair from MobaXTerm SSH client

And after that, click in Session button located in top-left corner of Figure 98 and set the following SSH parameters:

- Remote host: Floating IP assigned to instance
- Specify username: ubuntu (OpenStack assignes it automatically)
- Use private key: It is needed to provide the free5gc.key file

It is also necessary to configure an SSH gateway, as from the PHYSICAL SERVER (in Figure 17) we have no reachability to the instance. This can be done by clicking Networks settings > SSH Gateway (jump host) button which will open the window in Figure 100. In this window we must set the ASUS SERVER IP (10.49.27.151) and user (ectics) that will reach the instances.

Now that we have set and saved the SSH session in MobaXTerm we could access the instance by just clicking the button that has been created in '*Sessions*' panel (the star in Figure 98).

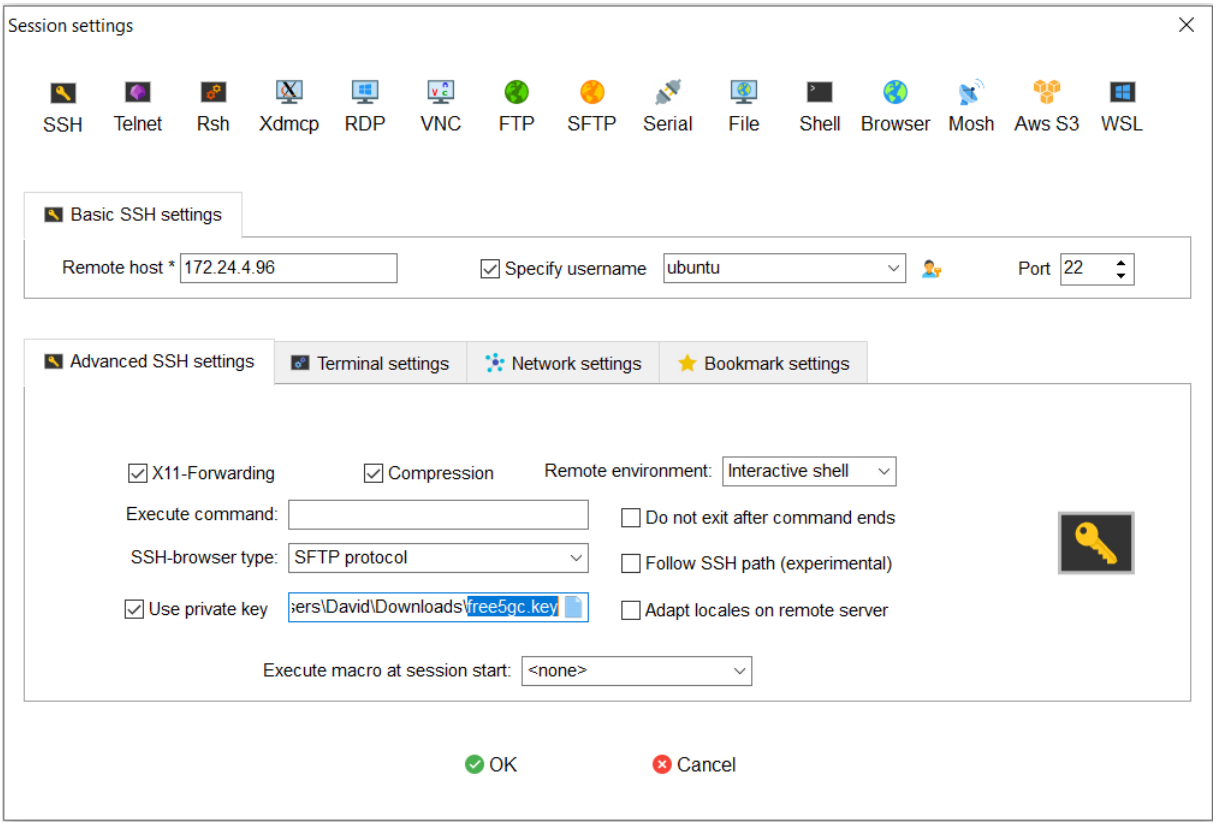

Figure 99. SSH Configuration to access the instance (1)

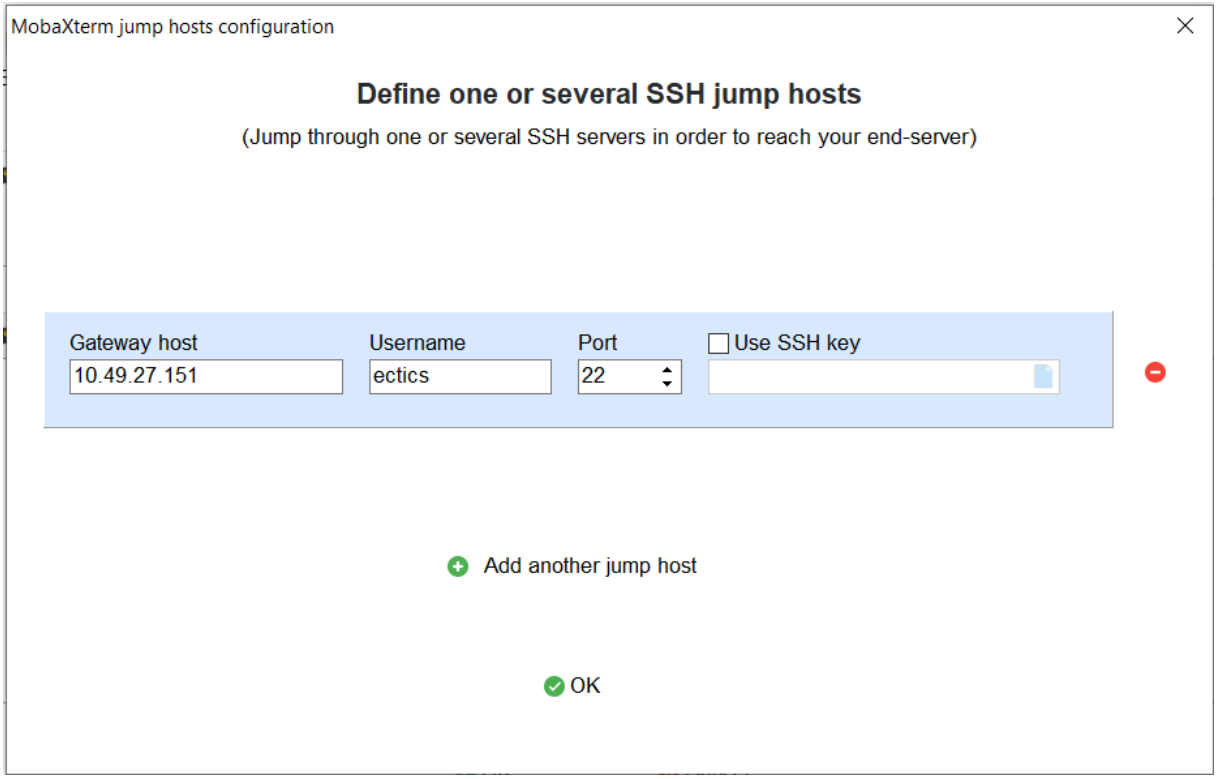

Figure 100. SSH Configuration to access the instance (2)

Annex 4. Validation

A.1 TestRegistration results

In Figure 101 can be seen the results of a simulated UE attach procedure test.

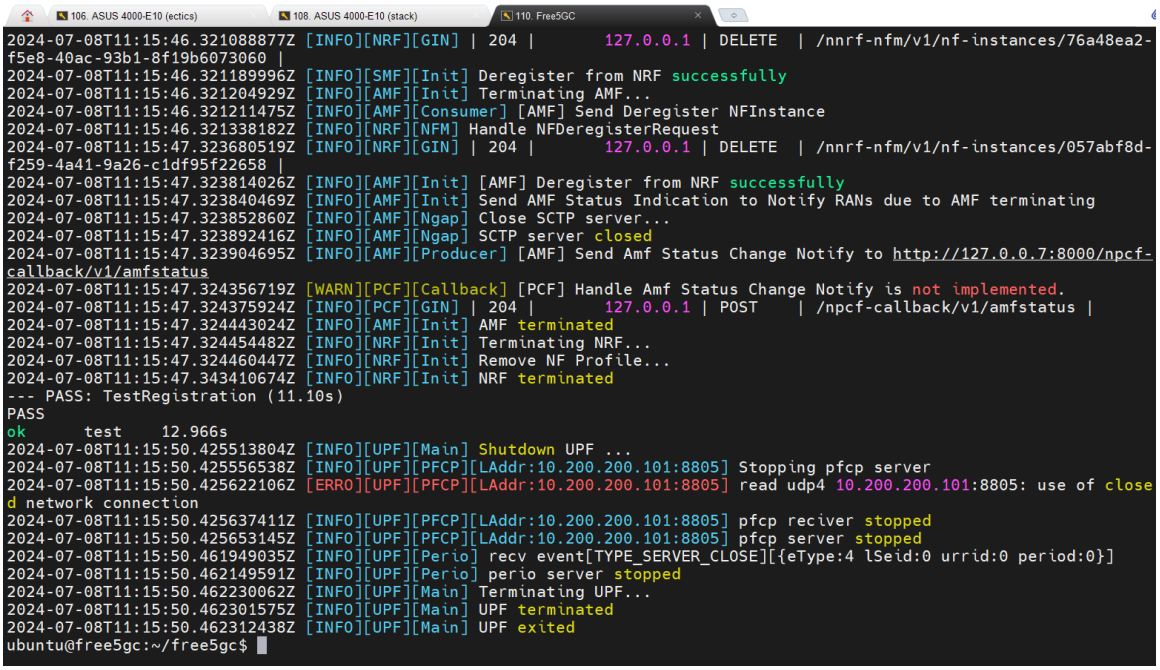

Figure 101. TestRegistration execution

A.2 GUTIRegistration results

In Figure 102 can be seen that GUTI (Global Unique Temporary Identifier) registration procedure with the AMF has been successfully passed.

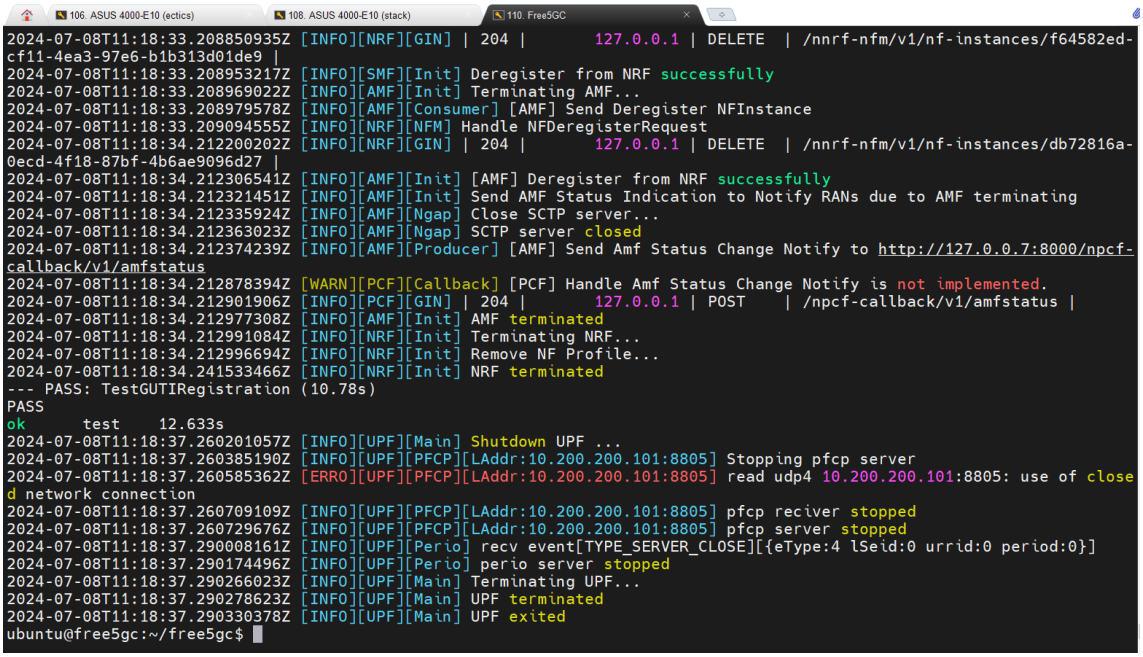

Figure 102. TestGUTIRegistration execution

A.3 TestServiceRequest results

In Figure 103 it is tested that UEs can request the AMF the use of some services as for instance the PDU session establishment.

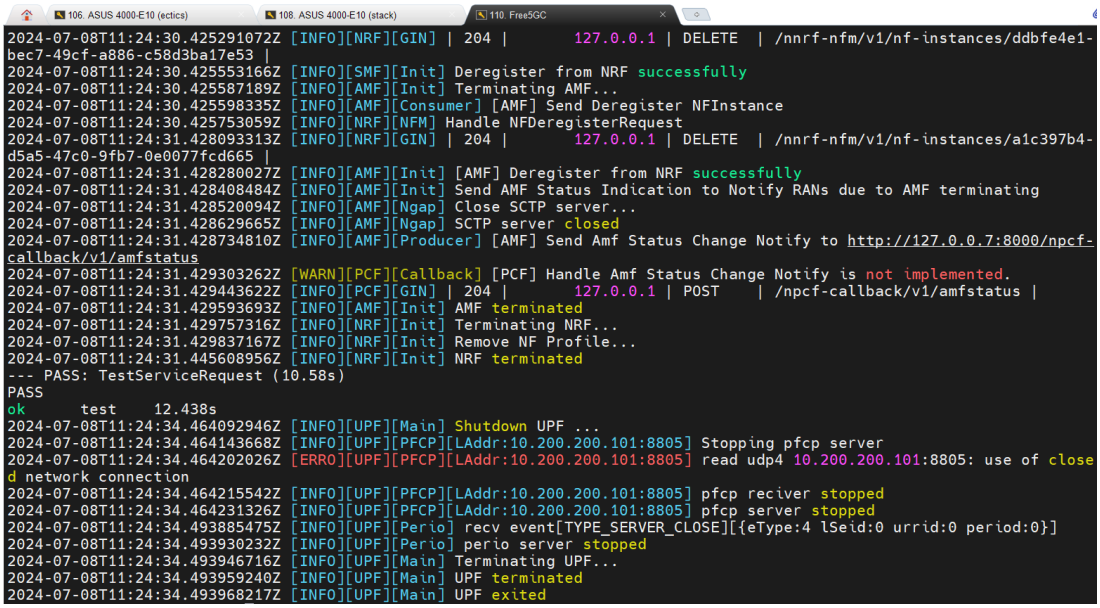

Figure 103. TestServiceRequest execution

A.4 TestXnHandover results

In Figure 104 we can see the results of testing the handover procedure from Xn interface. This simulates that a UE moves from one gNB to another without having to relocate the UPF. The test results are correct.

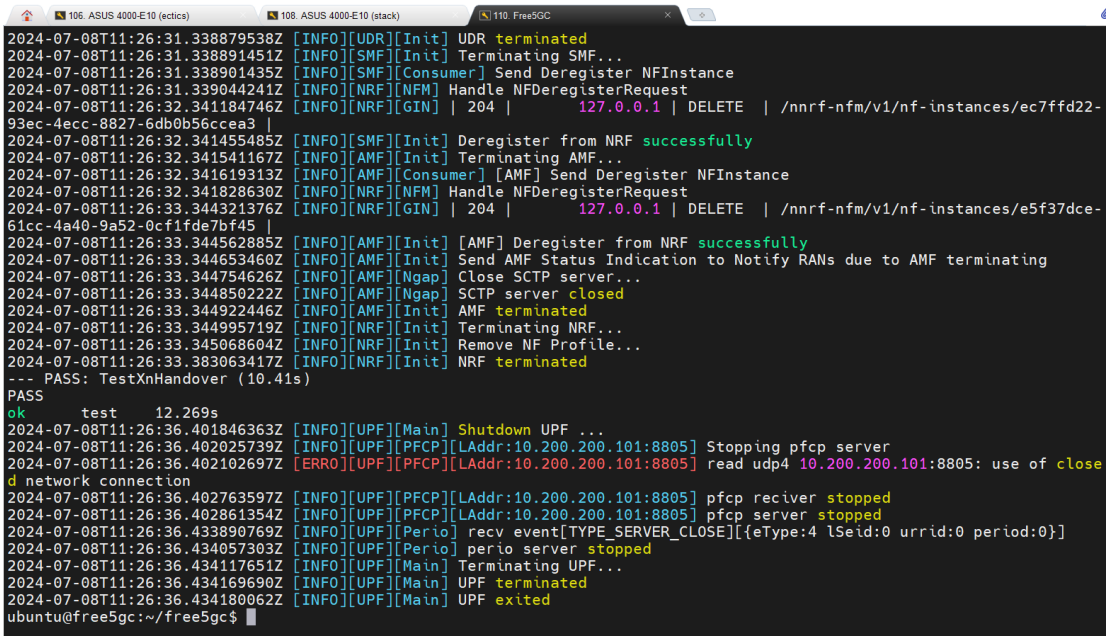

Figure 104. TestXnHandover execution

A.5 Deregistration results

Figure 105 contains the results of UE detach test which has been passed correctly.

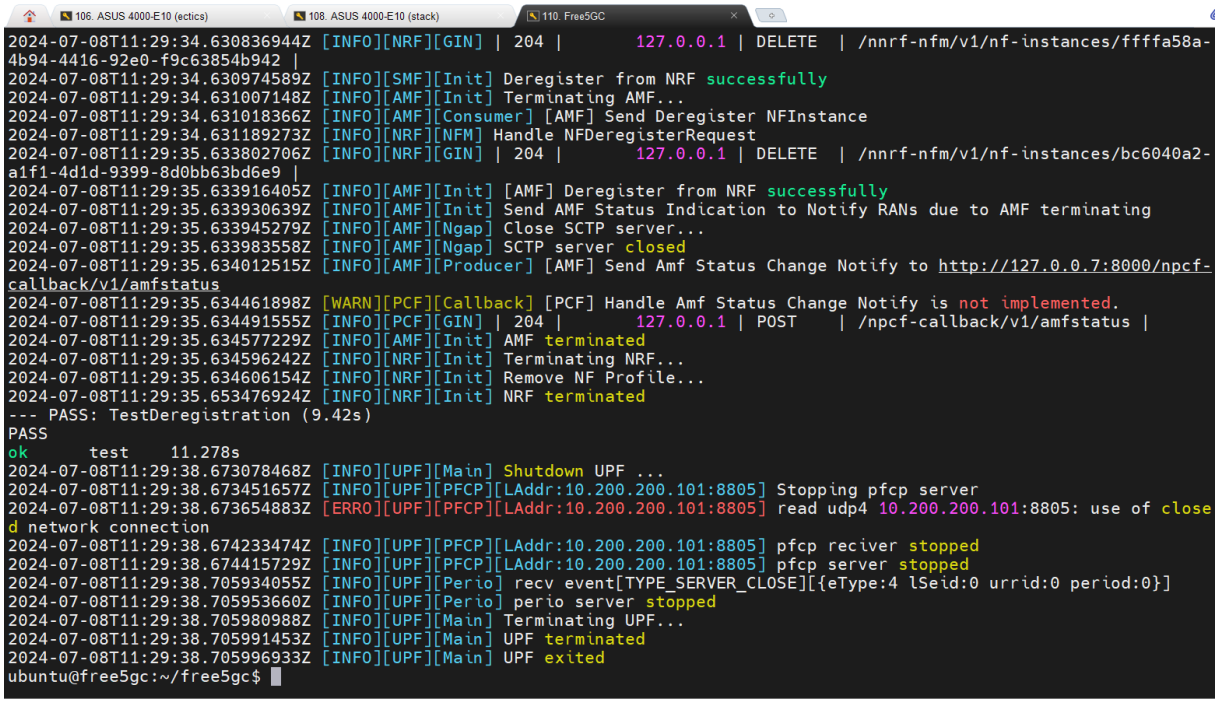

Figure 105. Deregistration execution

A.6 TestPDUSessionReleaseRequest results

In Figure 106 we can see the test results for PDU Session Release Request procedure in which the AMF releases a PDU session. Test result is fine.

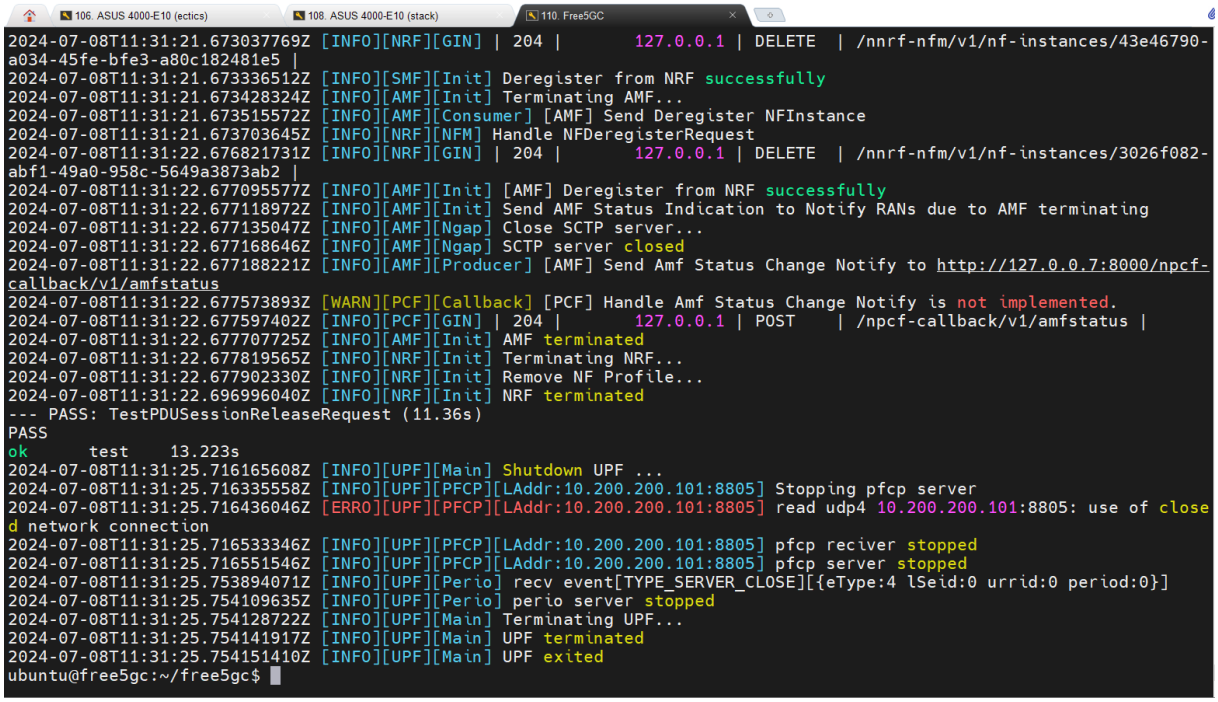

Figure 106. TestPDUSessionReleaseRequest execution

A.7 TestPaging results

Figure 107 shows the correct execution of a paging test in which is simulated a call to one of the UE that have been registered to the core

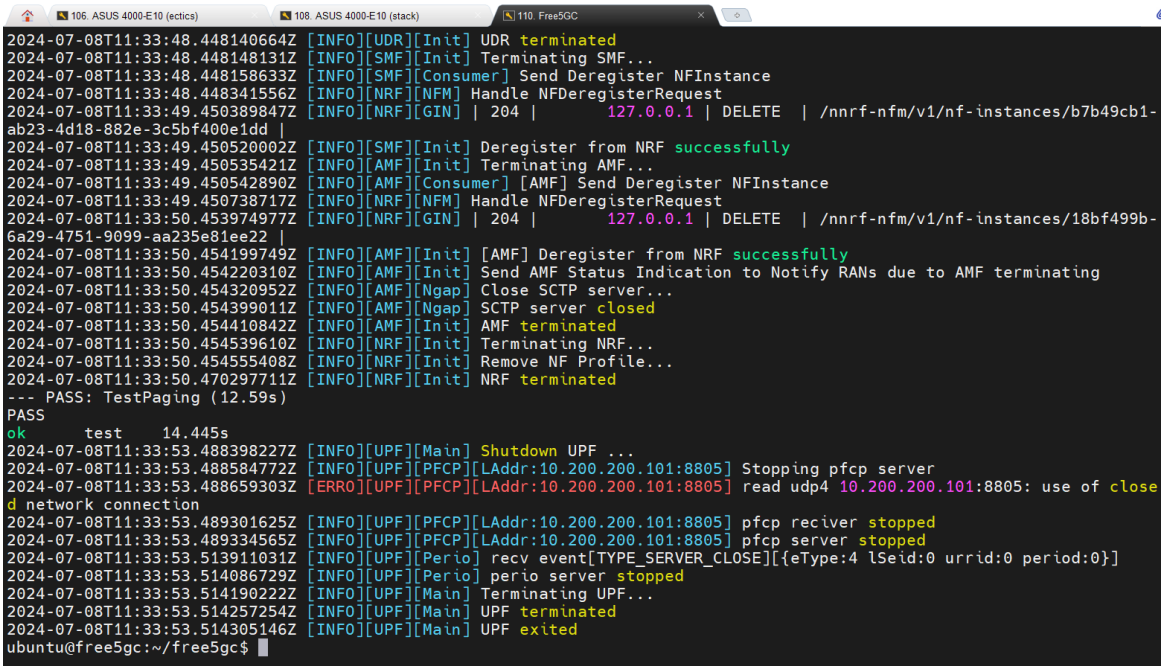

Figure 107. TestPaging execution

A.8 TestN2Handover results

Figure 108 shows the test results of N2 handover in which the UE changes from gNB without relocating AMF and UPF.

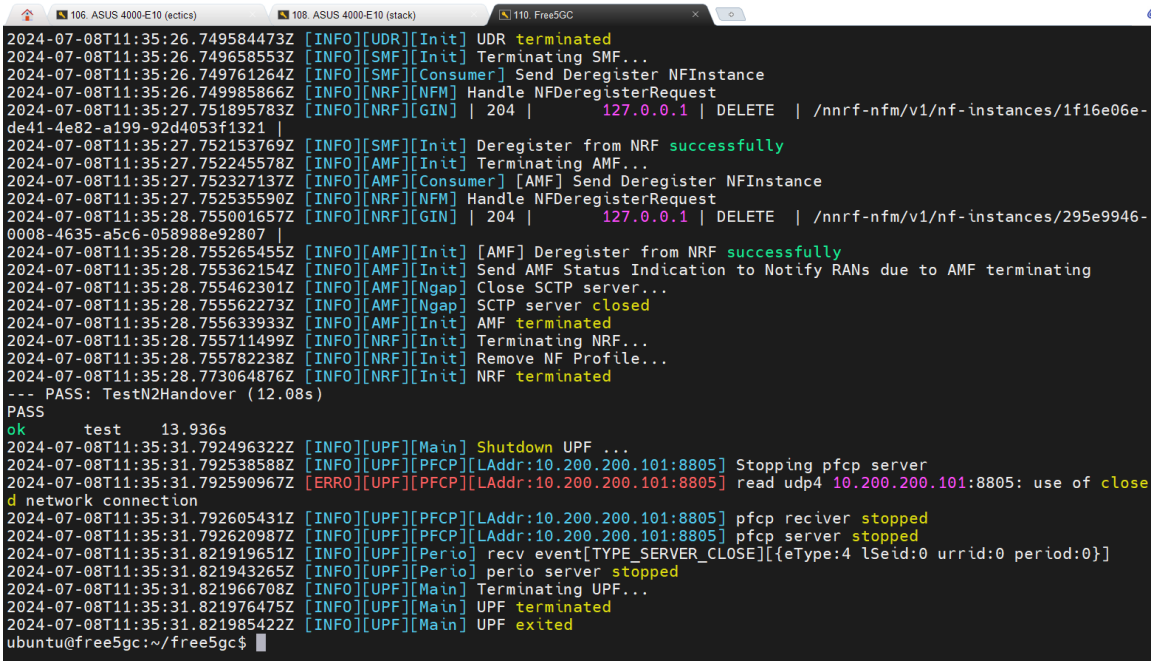

Figure 108. TestPaging execution

A.9 TestNon3GPPUE results

In Figure 109 it is shown the results of a test in which the compatibility with other UE that does not support 5G-NR standards (like GSM, UMTS, or LTE) is proved.

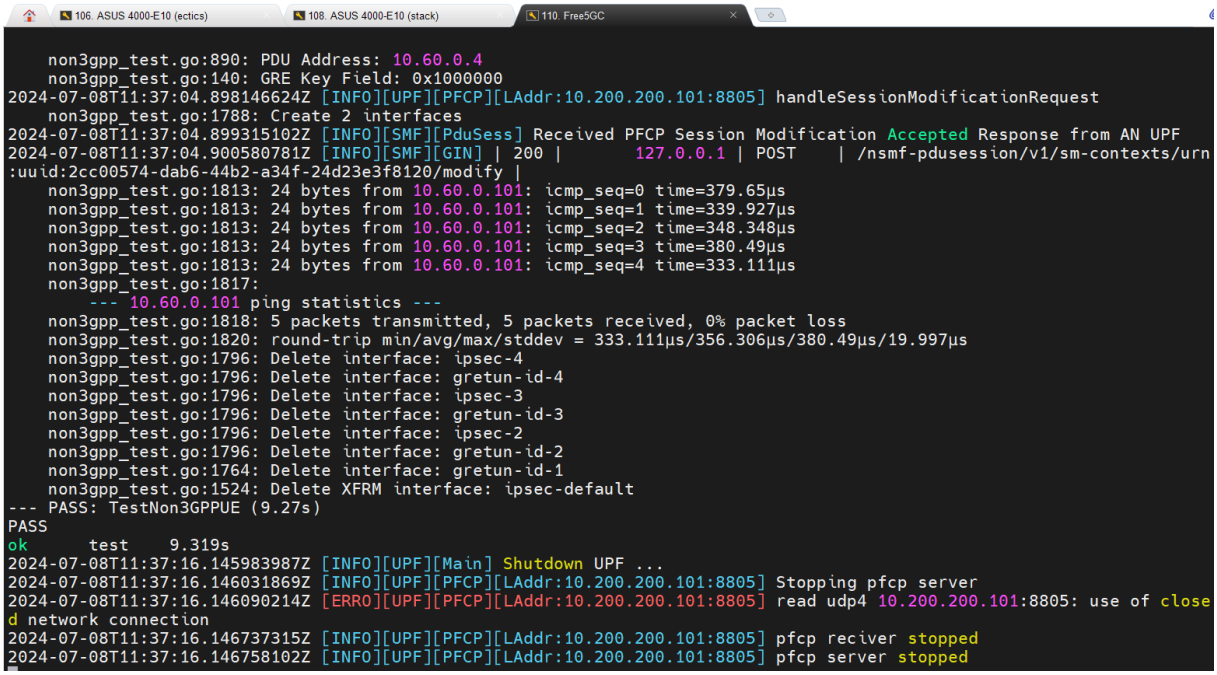

Figure 109. TestNon3GPPUE execution

A.10 TestReSynchronization results

Figure 110 shows the results of ReSynchronization test in which is simulated the correct functioning of the 5GC when UEs have to synchronize with gNBs.

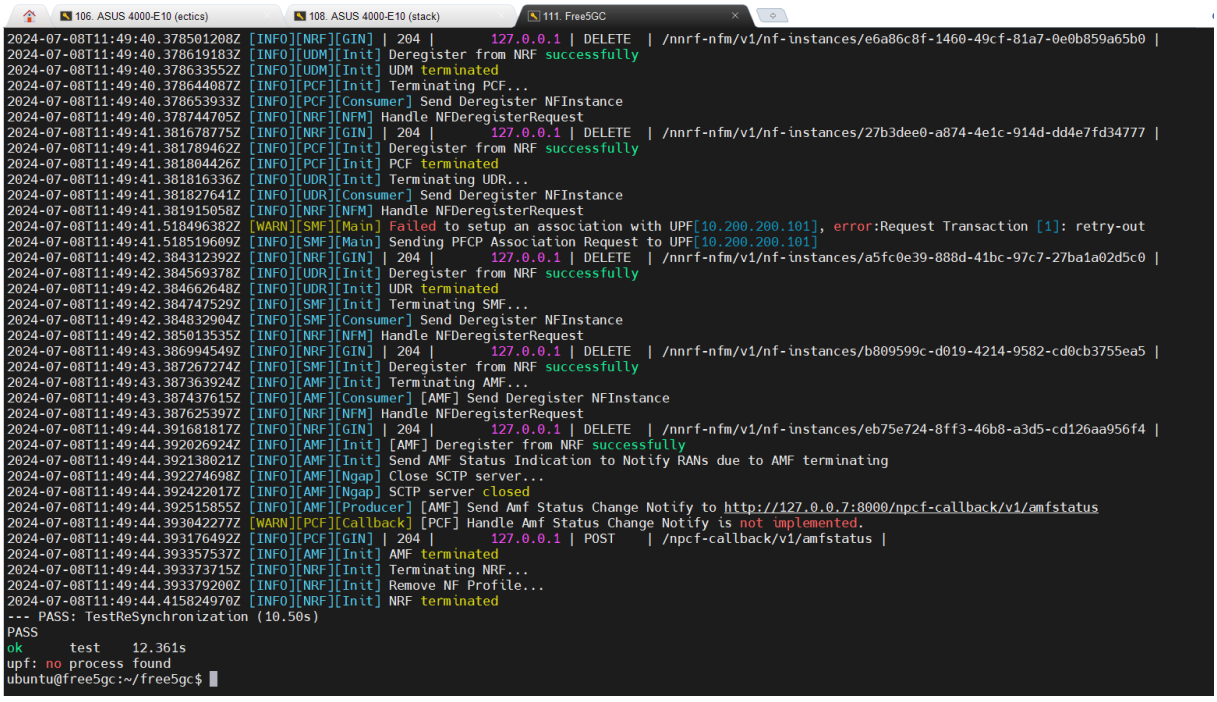

Figure 110. TestReSynchronization execution

A.11 TestRequestTwoPDUSessions results

In Figure 111 we can find the test results in which two PDU sessions are trying to be established

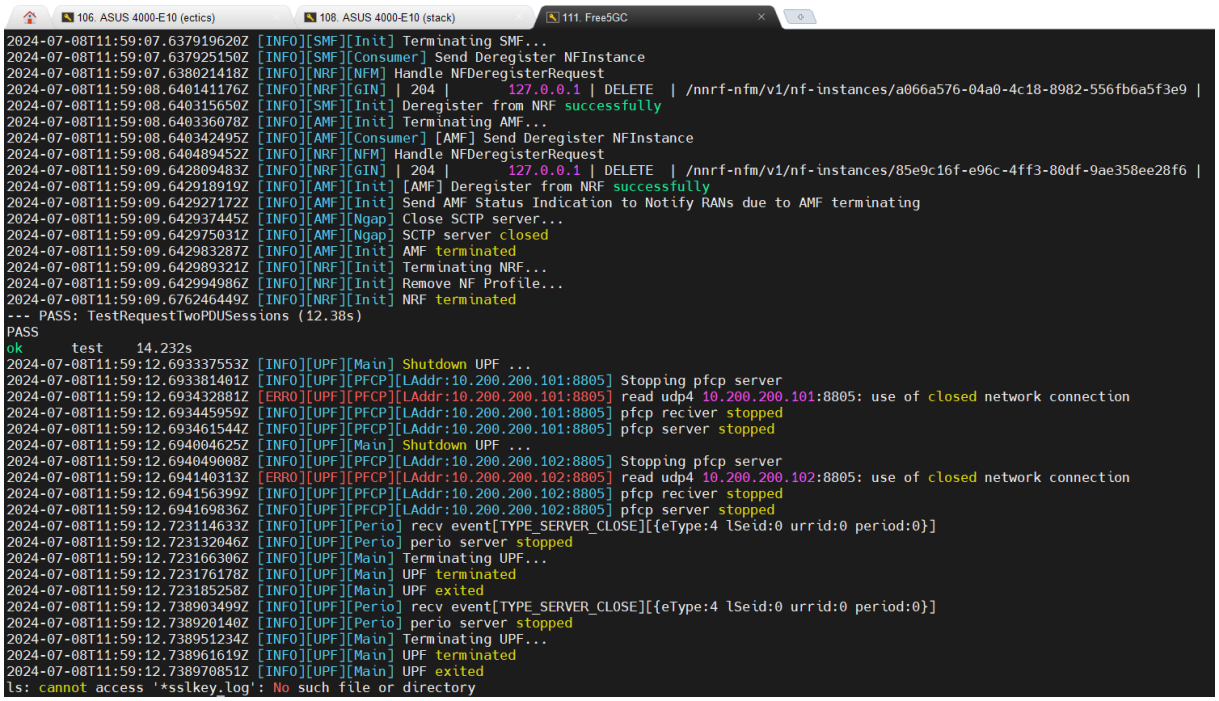

Figure 111. TestRequestTwoPDUSessions execution