Democratic and Popular Republic of Algeria
Abou Bakr Belkaid University - Tlemcen
Faculty of Science
Department of Computer Science

The final thesis of the study

For obtaining the Master's degree in Computer Science

Option: *Network and Distributed Systems (NDS)*

Topic

Virtualisation of IMS Core Using Openstack and Clearwater

Presented by:

- ABDELAZIZ Mustapha
- BELLAREDJ Menel

Presented on September 27th, 2020, in front of the jury composed of:

- Mr. BAMBRIK Ilyas	President
- Mr. ABDERRAHIM Alaa	Examiner
- Mr. BELHOCINE Amin	Supervisor
- Mr. SERRIR Boumediene	Co- Supervisor

Academic Year: 2019-2020

Acknowledgement

*First and foremost, we thank **Allah The Almighty** for granting us the strengths, the patience, and the courage to start and to complete this work.*

We would like to express our deep and sincere gratitude and appreciation to the following for their precious help, time and support:

*Thanks to the hosting **Algeria Telecom Company** for giving us the opportunity to do this project and for their support and integration of the students in their projects by providing us, the training ship, the tools and invaluable guidance throughout the research so we can improve ourselves and our country.*

*Thanks to our supervisor **Mr. BELHOCINE Amin** for his help and valuable advice, guidance, and encouragement throughout this work.*

*Thanks to our co-supervisor **Mr. SERRIR Boumediene** for his help, support and valuable knowledge, as well as for his complete availability in terms of training and supervision. He has always been encouraging us all along with this project.*

*Finally, special thanks to the honourable jury members consisting of: **Mr. BAMBRIK Ilyas** and **Mr. ABDERRAHIM Alaa** for accepting to examine our work and engaging with their propositions.*

Dedications

I dedicate this modest work

To my dear parents

The reason for everything, For their deep love, care, patience, support and encouragement and prayers,

To my brother and sisters

For their deep love, respect, tolerance, support, encouragement and happiness they share,

To my Family members

To my grandparents, uncles, and aunts, cousins and all the family for their love, support, the happiness they share,

To my teachers

To my teachers that I had in my life for their valuable, education, advice, love and help, you are the reason for who I am now,

To my dear friends

To my best friends who helped and supported me each one in his unique way, you always make my days I am very grateful to have you in my life,

to my Partner and Algeria Telecom development team

To Menel Bellaredj for her efforts, patience and help

To Ayoub Ghaffour and all the team for their support and help and advice,

To anyone helps us during this project

Abdelaziz Mustapha

Dedications

I dedicate this modest work

To my dear parents

who encouraged me and supported me and prayed for me throughout our work. Thank you for your deep love.

To my brother and sisters

For their deep love, respect, tolerance, support, encouragement and happiness they share,

To my Family members

To all the family Bellaredj, Hamnech for their love, support and the happiness they share it with me

To my teachers

To my teachers that I had in my life for their valuable, education, advice, love and help, you are the reason for who I am now,

To my dear friends

To my best friends Meriem & Amel who helped and supported me each one in his unique way, you always make my days I am very grateful to have you in my life,

to Research and development unit Tlemcen team

To Ayoub Ghaffour, who helped us a lot. To Lassouani Ibrahim and Boumeddane Oum Elkhier for their supports. To all the team,

To anyone helps us during this project

Bellaredj Menel

List of Abbreviations

Acronyms	signification
API	Application Programming Interface
AS	Application Server
CAPEX	Capital Expenditures
CIFS	Common Internet File System
CPU	Central Processing Unit
HSS	Home Subscriber Server
IaaS	Infrastructure as a Service
I-CSCF	Interrogating Call Session Control Function
IMS	IP Multimedia Subsystem
IP	Internet Protocol
ISC	Interface Switching Capability
ISO	International Organization for Standardization
KVM	Kernel-based Virtual Machine
LTE	Long Term Evolution
MGCF	Media Gateway Control Function
MRF	Media Resource Functions
NAT	Network Address Translation
NFS	Network File System
NFV	Network Function Virtualization
NIST	National Institute of Standards and Technology
OPEX	Operational Expenditure
PaaS	Platform as a Service
P-CSCF	Proxy Call Session Control Function
PSTN	Public Switched Telephone Network
QCOW2	QEMU Copy on Write
QEMU	Quick Emulator
QoS	Quality of Service
RAM	Random Access Memory
SaaS	Software as a Service
S-CSCF	Serving Call Session Control Function
SDN	Software-Defined Networking
SDP	Session Description Protocol
SGW	Signalling Gateway
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SP	Service Provider
TDM	Time Division Multiplexing
VDI	Virtual Disk Image
VHD	Virtual Hard Disk

VLAN	Virtual Local Area Network
VM	Virtual Machine
VPN	Virtual Private Network
VXLAN	Virtual Extensible LAN
XML	eXtensible Markup

Table of Contents

LIST OF FIGURES

Abstract

Cloudification of IP Multimedia Subsystem IMS is the new migration of the telecom operators, thus to its standardized architecture that provides a variety of multimedia services such as telephony, video, text, and Internet, …etc., which provides the services over a unified IP network. Operators will have more flexibility and ease to deploy new IT services or existing services to bring new added value to their networks with better performance and at a lower cost.
To cloudify the IMS, we used the open-source Openstack solution as infrastructure as a Service using a server of the Algeria Telecom of Tlemcen. Then we used the open-source IMS product: Clearwater IMS, to test the functionality of our solution, we used several free SIP-based software clients.

Keywords: IMS, Cloud Computing, Multimedia services, Clearwater, Openstack.

Résumé

Cloudification du IP multimédia subsystem IMS est la nouvelle migration des opérateurs télécoms, car il est basé sur son architecture standardisée qui fournit une variété de services multimédias tels que la téléphonie, la vidéo, le texte, Internet, …etc. et fournir ces services sur un seul réseau IP. Les opérateurs auront plus de flexibilités et de facilité de déployer de nouveaux services informatiques ou des services existants dans le but de fournir une nouvelle valeur ajoutée à leurs réseaux avec les meilleures performances et à moindre coût.
Afin de valider la cloudification IMS nous avons utilisé la solution open source Openstack en tant qu'infrastructure en tant que service (IaaS) en utilisant un serveur de la société Algérie Telecom de Tlemcen. Ensuite, nous avons utilisé la solution IMS open source : Clearwater IMS, pour tester le fonctionnement de notre solution, nous avons utilisé plusieurs logiciels gratuits de base sur SIP comme des clients d'IMS.

Mots clés : IMS, Cloud Computing, Services multimédias, Clearwater, Openstack.

ملخص

إن الانتقال IP multimédia subsystem IMS الى السحابيات يعد قفزة وتحدياً جديد لشركات الاتصالات، بناءًا على بنيته الموحدة التي تتيح مجموعة من خدمات الوسائط المتعددة مثل المكالمات الهاتفية والدردشة النصية ومكالمات الفيديو وعدة خدمات اخرى عبر شبكة IP واحدة، سيتمتع المشغلون بمزيد من المرونة والسهولة لنشر خدمات جديدة أو تطوير الخدمات الحالية من أجل تحقيق قيمة مضافة جديدة لشبكاتهم مع أداء أفضل وبتكلفة أقل.

من اجل تحقيق هذا التحول نحو السحابيات، استعملنا برنامج Openstack مفتوح المصدر كبنية تحتية كخدمة (IaaS) باستخدام خادم من شركة إتصالات الجزائر بتلمسان. ثم استخدمنا برنامج IMS مفتوح المصدر: Clearwater IMS، لاختبار عمل البرنامج، استخدمنا العديد من البرامج المجانية الأساسية على SIP كعملاء IMS.

الكلمات المفتاحية: السحابيات، الخدمات المتعددة الوسائط، Openstack, IMS, Clearwater IMS

General introduction

In the last past years, Cloud Computing has become one of the most exciting topics in the Information and Communication Technologies (ICT), and most companies if not all are migrating their services to the Cloud. Because Cloud computing is changing the way we deliver and create services through the Internet, and by changing the organization's approach for building and managing its IT infrastructure as well as its IT services, it can increase the productivity of companies and enable them to operate and focus on increasing profits and reducing costs.

Because Cloud computing relies on Virtualization technology whose main objective is getting most of the resource capacity, reduce the hardware investment cost, and the dependence of the hardware and application. Another aspect is the elasticity which is the scale-up and down quickly in a programmatical way. These characteristics of cloud computing make it suitable for telecom operators to migrate their services to the Cloud.

Most telecom operators have built their system based on the standardized IP Multimedia Subsystem IMS architecture, which has been defined in the context of next-generation networks (NGN) to provide IP based multimedia services. The IMS standard meets the needs of operators offering multiple services on a large scale to many customers with greater flexibility and at a lower cost. One of the main benefits of the IMS is the Fixed Mobile convergence (FMC) to ensure the interoperability of fixed and mobile networks on all IP architecture.

The absence of added value and profits have forced the telecom operators to take the initiative to invest in the Cloud, as they offer access to a wide range of services at the lowest cost.

Besides, the migration to the Cloud is a suitable choice for Telcom operator to increase the revenues and bring new value to their services. By reducing the CAPEX and OPEX deployment and maintenance costs of applications, and improve the QoS of the operators, which lead to client satisfaction and increase the revenue to the operator.

This work is organized into three chapters:

In chapter one of this paper, we focus on defining Cloud Computing and different aspects related to it, its characteristics, and its deployment and service models.

In chapter two, we dive in the most used opensource Openstack Infrastructure as a Service platform and define each of its components for the Cloud.

In chapter three, we give a brief definition of the IP multimedia subsystem (IMS) standard and its architecture. We present the clearwater IMS architecture and components. We walk through the Openstack deployments method and configurations steps, and then we focus on Clearwater deployment on Openstack. Finally, we conclude our work with a general conclusion

Chapter 1: Cloud Computing

1.1 Introduction

The evolution of information and communication technologies (ICT) has given birth to a new paradigm which is: Cloud Computing.

In this chapter, we present the necessary standards and definitions of Cloud computing. We also describe virtualization, which is an essential part of cloud computing, cloud services, types of Cloud and its actors as well as its architecture, advantages, disadvantages, main objectives of cloud computing.

So, what is cloud computing and how it works?

1.2 Definitions

1.2.1 Cloud Computing

To know the concept of cloud computing more, we put in your hand many definitions of this concept, as some define it as technology based on transferring the processing and storage space of the computer to the Cloud, which is a server device that is accessed over the Internet.

Even if experts disagree on its exact definition, most agree that it includes the concept of services available on-demand, expandable as well. In contradiction to current systems, the services are virtual and unlimited, and the details of the physical infrastructures on which the applications are based are no longer the responsibility.

NIST (National Institute of Standards and Technology):

" Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [1]

One of the essential points of these definitions is the notion of scalability on demand, elasticity, that is, we only pay for what we use. That is a considerable advantage over an enterprise-specific infrastructure where servers are very often underutilized. So, Cloud Computing is a concept which consists of deporting to remote servers' storage and computer processing traditionally located on local servers or the user's workstation. It consists of offering IT on-demand services accessible from anywhere, any time and by anyone.

This cloud model is composed of five essential characteristics, three service models, and four deployment models. [1]

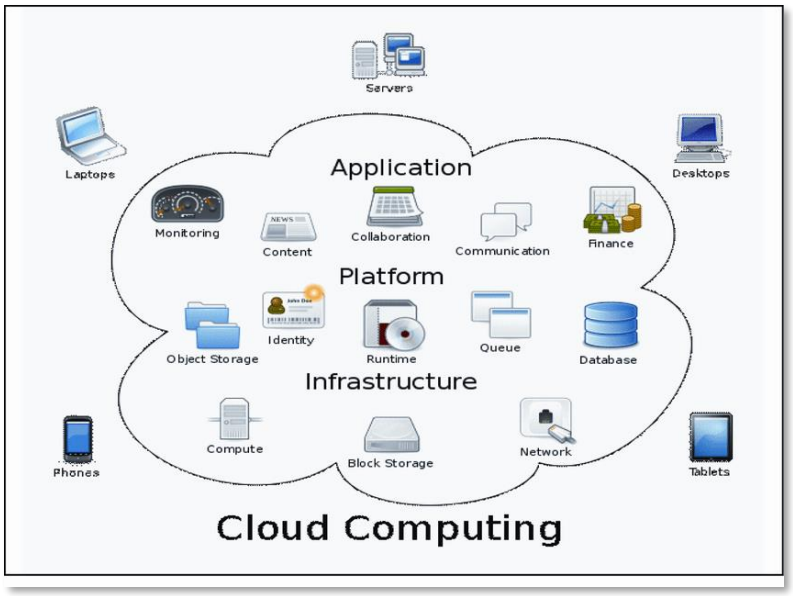

Figure 1.1: Cloud Computing Environment *[36]*

1.2.1.1 Pros

- **Ease of access**: to all the applications and services from anywhere and anytime via the Internet. Because the data is stored on the servers of the company providing the service and not in our device.

- **Reducing costs**: For companies, as it is not necessary to pay attention to purchasing computers with high specifications of memory and storage space, any device as a regular computer and by using any web browser can access the company's cloud services (editing documents, storing files, editing pictures), companies also no longer have to buy expensive equipment (servers) Provides email service to its employees or massive volumes to back up company data and information. [2]

- **Ensuring the service works permanently**: The commitment of the company providing the cloud storage service to ensure that the service works with the latest equipment and knowledge that is around the clock to protect our data, as well as fix any emergency malfunctions as quickly as possible. That saves us a lot of time and cost as a user or owner of a company that is responsible for managing its hardware and software. [2]

- **Leveraging the huge infrastructure provided by the cloud services**: to carry out scientific experiments and experiments, some complex calculations require years to conduct them on regular computers, while companies like Google and Amazon allow their clouds of thousands of servers linked to each other to conduct such calculations in minutes or hours.

1.2.1.2 Cons

- **Interruption of the Internet service**: the interruption to the Internet affect the ability to perform a work, cloud applications need an internet connection.

- **Security concerns**: Some fear that all its information and files placed with the companies that provide cloud services. If the service is subject to a successful operation, the hacker may be able to obtain the users' information, as if the company resorted to selling our information or benefiting from it, then this would be a real problem. the only guarantee is to turn to large companies with high reliability and a good reputation in this field. [2]

- **Cloud applications have not reached the required level**: Most cloud applications have not yet reached the level of traditional desktop applications, so far image editing applications across the web have not reached levels comparable to, for example, the traditional Photoshop application and document editing applications across the web have not reached the level of Microsoft Office, but it is gradually approaching this with the passage of years. [2]

1.2.2 Virtualization

Virtualization is the technology that allows us to run multiple operating systems, servers, or applications on the same physical machine, so they work in multiple simulated environments known as virtual machines (VMs) as if they were on separate computers. This technology allows us to share resources from a single physical hardware system, and, to run multiple systems on the same physical hardware at the same time. Virtualization allows us to make use of the full hardware capacity. Virtualization is based on:

The physical hardware, equipped with a hypervisor, is called the host, while the many VMs that use its resources are guests. These guests treat computing resources like CPU, memory, and storage as a pool of resources that can easily be relocated. Operators can control virtual instances of CPU, memory, storage, and other resources, so guests receive the resources they need when they need them. [3]

A Software called a Hypervisor is used for creating and running virtual machines. This hypervisor can either be installed directly on the hardware without a hosting operating system which is known as **type one** also known as **bare-metal** hypervisor; or **type 2** or **hosted** hypervisors which they are installed on a hosting operating system [4]. Figure 1.2 compares its high-level architectures.

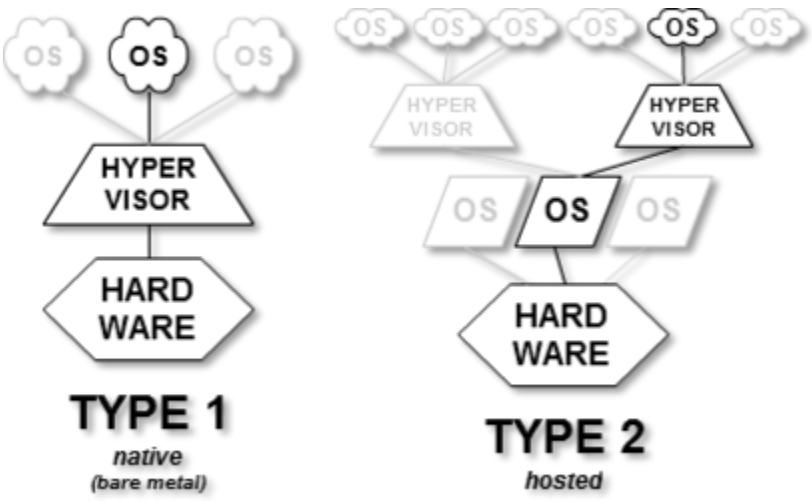

Figure 1.2: The two forms of full virtualization *[32]*

1.2.3 Software Defined Networking

Software-Defined Networking (SDN) is a network architecture approach that enables the network to be intelligently and centrally controlled, or 'programmed,' using software applications. This helps operators manage the entire network consistently and holistically, regardless of the underlying network technology. [5]

1.2.4 Network Function Virtualization

Network functions virtualization (NFV) is an approach to virtualize network services, such as routers, firewalls, and load balancers, that have traditionally been deployed on proprietary hardware. These services are delivered as virtual machines (VMs) on commodity hardware, which allows service providers to run their network on standard servers instead of proprietary ones. With NFV, there is no need for dedicated hardware for each network function. NFV improves scalability and agility by allowing service providers to deliver new network services and applications on demand, without requiring additional hardware resources. [6]

1.2.5 Datacentre

At its simplest, the data centre made up of many huge servers, basic and backup power supplies, and it is connected to the Internet at very high speeds. It is also often in a private building equipped with its own equipment, including temperature control, fire extinguishers, electricity regulation, and high-security specifications, so such centres must contain a high level of security protection, in addition to that it should not cut off the power supplies because some devices and servers lose their data if the power source is cut off from them even for a short period of seconds, so we find that they are equipped with a basic power supply, as well as a backup provider such as batteries, to be continuous and not cut off completely. The basic components of a datacentre design include routers, switches, firewalls, storage systems, servers, and application delivery controllers. [7]

1.2.6 Service provider

The service provider (SP) is a company that provides services to the customer, virtual hardware, software and other services leased and managed by the supplier based on a service level agreement (Service Level Agreement or SLA) and a negotiation between the service provider and the consumer. [8]

1.2.7 Virtual Machine

A Virtual Machine (VM) is a computing resource that uses software instead of a physical computer to run programs and deploy apps. One or more virtual "guest" machines run on a physical "host" machine [9]. The Virtual Machine provides the user with an integrated environment for a specific operating system identical to using any other computer. Thus, the user can use two or more operating systems on one computer [10]. It also allows the possibility of using one computer by more than one user, as each user works on an operating system and programs different from others.

1.3 Cloud deployment models

1.3.1 Private Cloud

The cloud infrastructure is provisioned for exclusive use by a single organization. It could be operated and managed by the organization itself and physically located at the company's datacentre (on-premises) or managed and hosted by a third-party provider (off-premises). [1]

1.3.2 Public Cloud

A public cloud is based on the cloud computing standard model in which the service provider makes resources accessible to the public. It offers to its users the scalability, flexibility and also additional control services. Public cloud services can be free or pay-as-you-go model. [1]

1.3.3 Hybrid Cloud

Hybrid Cloud is a combination of public and private clouds. A company can have part of its services in its own infrastructure, but also in public clouds. Or can use the public Cloud just when there is an overload in usage. It is also a good option for companies that want to protect their critical data locally and do not want to invest too much in infrastructure. the benefits of a hybrid Cloud. [1]

- It combines the characteristics of public and private clouds,
- Use in the facility with small data or need applications of its own,
- Customer can choose between public or private cloud applications and services,
- The facility has the option to maintain control and security.

1.4 Cloud service models

As identified in the NIST cloud computing reference architecture, they classified the services into three categories, which presents the type of services that could exist in the Cloud with the level of abstraction provided to the clients. Figure 1.3 shows the three most common service models:

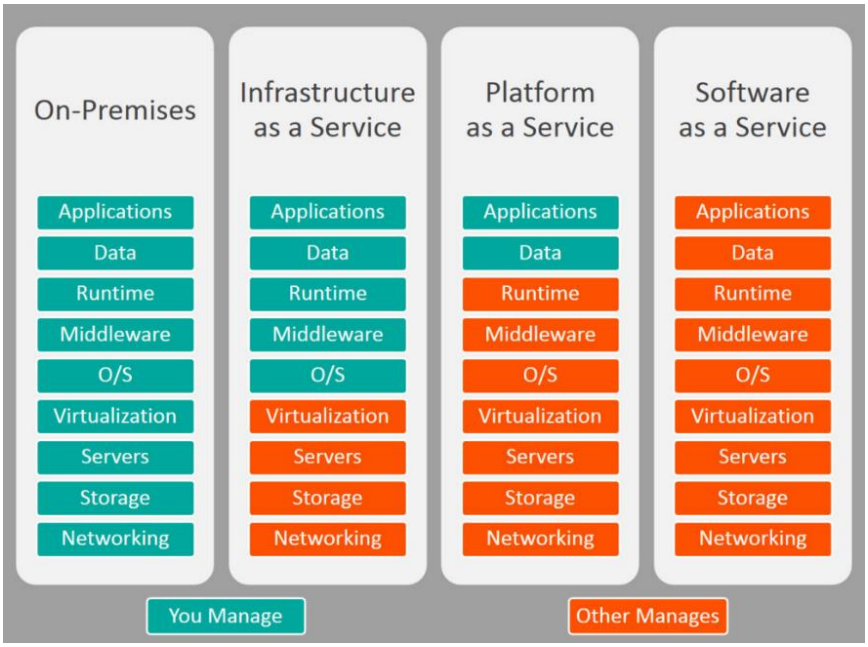

Figure 1.3: The three most common service models *[31]*

1.4.1 Infrastructure as a Service

Infrastructure as a service (IaaS) provides customers with basic IT resources such as storage, networks, and possibly operating systems. Here we treat the Cloud as a limited infrastructure with specific processing capacity, memory size, storage space, and the number of specific users, and we are free to use it the way we like. For example, we can install multiple operating systems, install multiple applications on each system, and allow a certain number of users to access each operating system to use its applications without letting them mix. [11]

1.4.2 Platform as a Service

Platform as a service (PaaS) allows customers to develop applications using programming tools provided by the Cloud. Google App Engine and Windows Azure offer this kind of solution. The customer gains more flexibility, but in a technical environment defined by his supplier.

Using the Cloud as a platform to put several applications on it, and we can work on all of them as well as we can put a complete operating system also. There is an integration between the applications, for example, if we design something in Photoshop and then it is useful for another application, so it moves and adds effects, so we get a video clip with sound. [11]

1.4.3 Software as a Service

The software as a service model allows customers to use the provider's applications that run on a cloud infrastructure. Generally, these applications are accessible from different client devices through a thin client interface, such as a web browser (for example, web-based email).

The user here does not control the operating system in the Cloud and does not control the hardware or network connection. YouTube can be considered within this classification, the video browser on the site is the application loaded on the Cloud, and through it, we can access the existing video clips. However, we cannot change anything on the site; the end-user does not require any specific IT knowledge. [1]

1.5 Conclusion

During this chapter, we have provided a theoretical basis on Cloud Computing, by presenting its types, its architecture, its services (IaaS, PaaS, SaaS), its advantages and disadvantages, and other fundamental concepts that are related to the Cloud and required to understand this topic like virtualisation, VMs, datacentres etc. to apply its concepts to our context. In the next chapter, we define the Openstack Infrastructure as a Service platform and its components.

Chapter 2: Openstack

2.1 Introduction

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a data centre, all managed and provisioned through APIs with common authentication mechanisms. [12]

Openstack is an open-source **IaaS** (Infrastructure as a Service) platform that enables administrators to create, control and monitor their cloud infrastructure easily with the help of interconnected components that provide different services using a web interface or Command-Line. [13]

Rackspace and NASA founded Openstack **in 2010**. Now it's managed by The **Openstack Foundation**, established in 2012, many companies have joined the OpenStack foundation. Among these are **Canonical, Red Hat, SUSE, Cisco, Dell, HP, IBM, Yahoo, Oracle, Orange, Cloudwatt, EMC, VMware, Intel, NetApp.**

2.2 Openstack Components

OpenStack has a modular design. The OpenStack project includes several interrelated subprojects that help manage different aspects of hardware resources, including compute service, storage, networking, and other related services, each offering their own set of APIs to facilitate integration. At a high level, OpenStack services are categorized as core services and optional services. As the name mentions, the core services have the essential functionality of providing IaaS features for OpenStack. The optional services are just extensible functionality of OpenStack that improve the Openstack Platform, which provides the extended functionality for the IaaS platform [14]. The core services are those essential for deploying a Cloud under Openstack such as Nova, Glance, Keystone, and Neutron.

The optional services as their name suggests are, those which may or may not be chosen and used in an Openstack Cloud, this depends on the use cases: Cinder, Swift, Horizon, Ceilometer, Heat, Sahara, Trove, etc. Figure 2.1 below illustrates the service relationship. As we can see, all services provide a service for the VMs. As Examples Glance provides Images, Neutron Provide network connection, etc., all these services work together to ensure the good functioning of the VMs in particular and the Cloud in general.

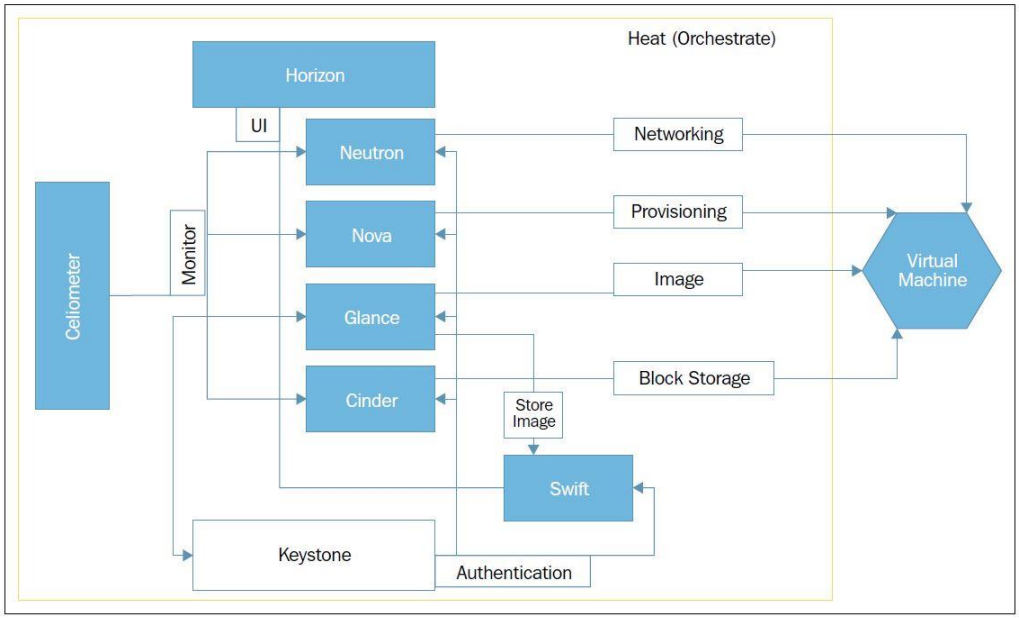

Figure 2.1: Service relationships *[35]*

2.2.1 OpenStack Compute (nova)

The Compute service is a mandatory part of the Openstack cloud platform, and it's been there since the birth of Openstack. It provides provision and management of VMs, and the technical name is Nova. Nova needs additional services to function and serves VMs correctly, it interacts with Identity service for authentication, and with Image service for disk and server images. [15]

Nova provides the ability to define Virtual hardware templates or Flavor which represents the amount size of RAM, disk, and CPU, and so on. The default installation offers five flavours, and are configurable by admin users.

Nova daemons run on top of Linux servers (Compute Node) to provide the compute service. Figure 2.2 shows how various processes and daemons work together to form the compute service and the relationship between them. The compute node runs a hypervisor that operates instances. By default, Compute uses the KVM hypervisor, it supports many other hypervisors like VMware, Xen and Hyper-V and other virtualization technologies.

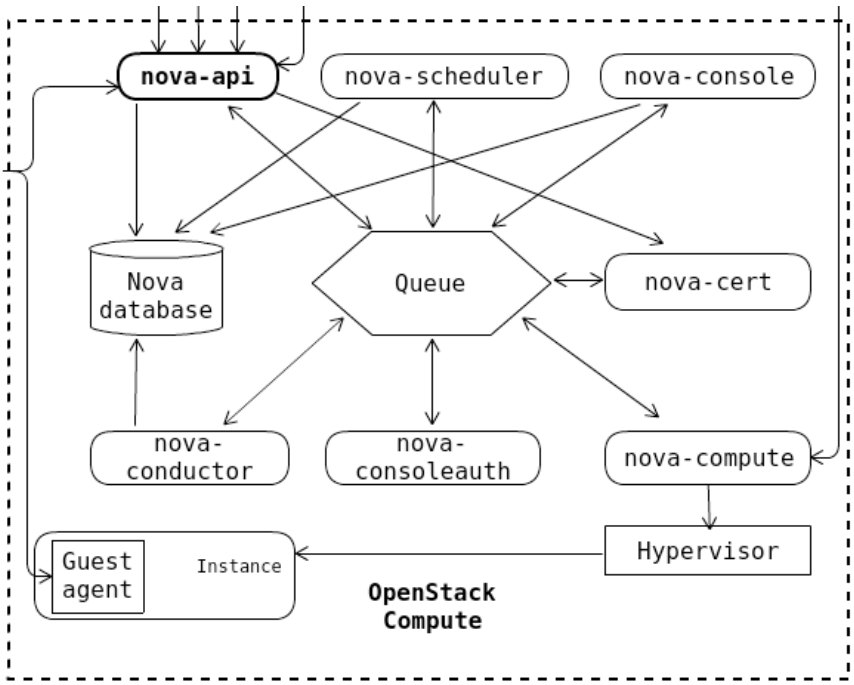

Figure 2.2: The Openstack Compute daemons *[34]*

2.2.2 Image Service (Glance)

Glance provides disk images to instances. It also allows users to store backups of these disks, it can store these disk images in several ways: in a folder on a server, but also through the OpenStack object storage service or in decentralized storage such as Ceph. Glance not only stores images but also information about them, the metadata. These metadata are for example the format of the disc (such as QCOW2 or RAW) or ISO. There are several VM images formats, such as **RAW, QCOW2, ISO, VDI, VHD.**

2.2.3 Storage Service

Cloud providers have developed services to fill the storage needs of modern enterprise application deployments, and they mostly fit into three categories: Object storage and Block storage, and File-based storage [16]

2.2.3.1 OpenStack Block Storage (Cinder)

Cinder is a Block Storage service; it's designed to present storage resources to instances. It manages the creation, attachment and detachment operations of these devices on the servers. In addition to local storage on the server, it can use multiple storage platforms such as Ceph, GlusterFS, and many others.

Block storage is used for a high-performance scenario like database storage, but also to provide the server with low-level access to the storage device. Cinder also manages the creation of

snapshots, very useful for backing up data contained in block devices. Snapshots can be restored or used to create new volumes.

Block Storage service offers two types of block resources: volumes and snapshots, which are derived from volumes and volume backups [17]:

- **Volumes**: they are like logical disks that can be attached or detached to instances as a secondary disk, or they can be used as the root disk to boot instances.
- **Snapshots**: Snapshots are a specific state of the volume they are used to save a state of a volume or to do backups they are Read-Only point in time copy of a volume.
- **Backups**: A saved copy of a volume currently stored in OpenStack Object Storage (Swift).

2.2.3.2 OpenStack Object Storage (Swift)

OpenStack Object Storage, also known as Swift, was implemented by Rackspace as cloud files. Object Storage allows scalable and highly redundant storage on commodity hardware. It is managed similarly under OpenStack. With OpenStack Object Storage, we can store many objects of virtually unlimited size restricted by the available hardware and grow our environment as needed to fit our storage needs. The redundancy nature of Object Storage is ideal for archiving data (such as logs) as well as providing a storage system that OpenStack Compute can use for virtual machine instance templates. [18]

2.2.3.3 File Systems service (Manila)

In a multi-tenant OpenStack cloud environment, the Shared File Systems service (manila) provides a set of services for the management of shared file systems. The Shared File Systems service supports multiple backends in the form of drivers and can be configured to provision sharing from one or more backends. Shared servers are virtual machines that export file shares using different file system protocols such as **NFS, CIFS, GlusterFS, or HDFS**. [19]

2.2.4 OpenStack Networking (Neutron)

The Neutron networking service provides networking as a service for other OpenStack services, e.g., OpenStack Compute. Neutron is very similar to networking in the real world. The Virtual machines require Layer 2 (L2) network connectivity minimally, and also may need internet/external network access. For the external network connectivity, in the real world, we use a router (L3 functionality), Neutron handles the virtual networking infrastructure which includes creation and management of networks, switches, subnets, routers, firewalls and virtual private networks (VPNs) [20]. When a new VM is created, the Nova compute API communicates with the Neutron API to connect the VM correctly to the specified networks. It does so by plugging in the virtual network interface cards (vNICs) of the VM into the particular virtual networks with the use of Open vSwitch (OVS) bridges.

2.2.4.1 Network Types

The network types in OpenStack Neutron is broadly classified into two types:

- Provider networks
- Self-service networks

2.2.4.1.1 Provider networks

The provider network is a network that directly associated with a physical network in the data centre. It allows VMs to connect to the external network, and the provider network can be shared by many tenants/projects in OpenStack. The provider network can only be created and managed by the OpenStack admin who has access to manage the actual physical networks in the datacenter. This is because configuring the provider network requires configuration changes in the physical network infrastructure level. [20]

2.2.4.1.2 Self-service networks

They are also called tenant networks, that are related to the user's tenant/project which enables the users with **Member** role to create and manage their network topology without involving the help of the Openstack administrator. These virtual networks can be connected by a virtual router and linked to an external network; the self-service networks are created by tenants members to attach them to instances and cannot be shared with other tenants. [20]

2.2.4.2 Network traffic flow

To understand the Neutron clearly, one must know the clear picture of how Neutron manages the packets flow in OpenStack. The following Figure 2.3 shows the components and connectivity of the self-service network, using the Open vSwitch plugin:

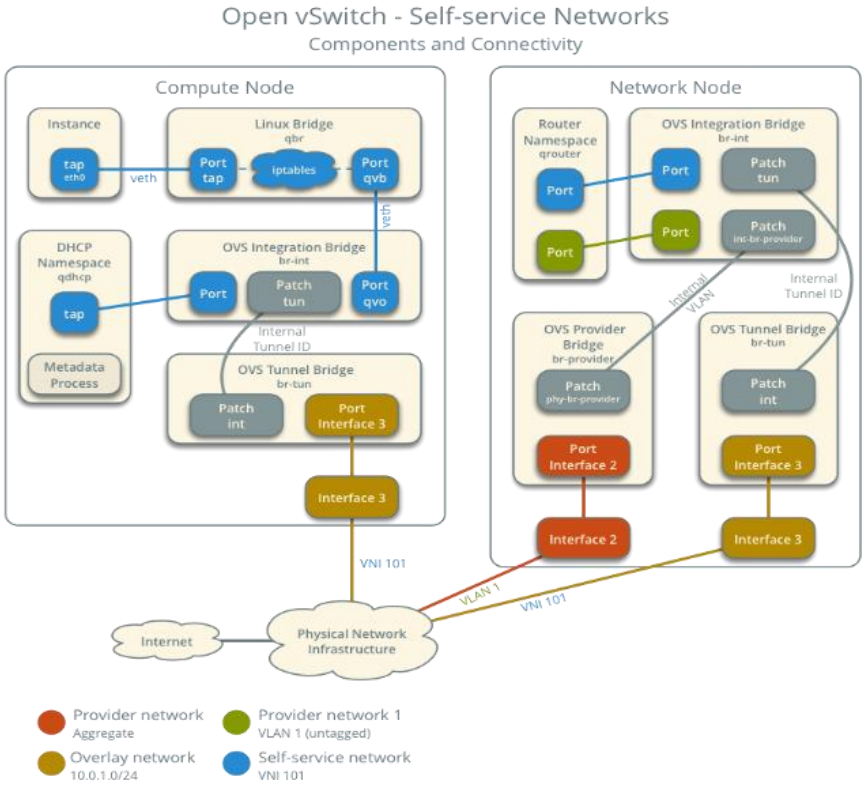

Figure 2.3: Open vSwitch - Self-service network *[30]*

2.2.5 Authentication Service (Keystone)

As we know the easiest way to authenticate a user must ask for credentials and then verify their existence on the database, which is very common and used on any type of authentication on personal devices or other.

When it comes to a lot of separate services working together like Openstack, we have to reconsider this approach. The main problem is the inability to use single-user identity to authenticate anywhere. Keystone is The OpenStack Identity service that authenticates users and tenants by sending an authorization token between all OpenStack services. This token is used for authentication so that we can use that service. It provides a central directory of services and users as well as their roles and authorization. Each action must be authenticated from Keystone (verification by token).

In Keystone, we have the concepts of tenants, roles and users. A tenant is a project, which contains resources such as users, images, and instances, as well as networks that are known only to that particular project. A user can belong to one or more tenants and can switch between these projects to access these resources. Different roles can be assigned to users of a tenant. In the most basic scenario, a user can be assigned the role of an administrator or simply be a member. When a user

has administrator privileges within a tenant, he can use features that may affect the tenant (such as editing external networks), while a regular user is assigned the role of member, typically assigned to perform user-related roles, such as multiplying instances and creating volumes. [21]

2.2.6 Dashboard Service (Horizon)

Horizon dashboard service is a web-based interface that enables users and administrators to perform various OpenStack services such as instances management and other configurations, etc. It allows users to access virtual machines, manipulate their private networks and other OpenStack resources through a web-based graphical user interface (written in Python using Django). This makes it one of the most popular components of OpenStack, as it is an ideal platform for any user doing their service freely. Figure 2.4 below illustrates the Dashboard service, with different functions Openstack services.

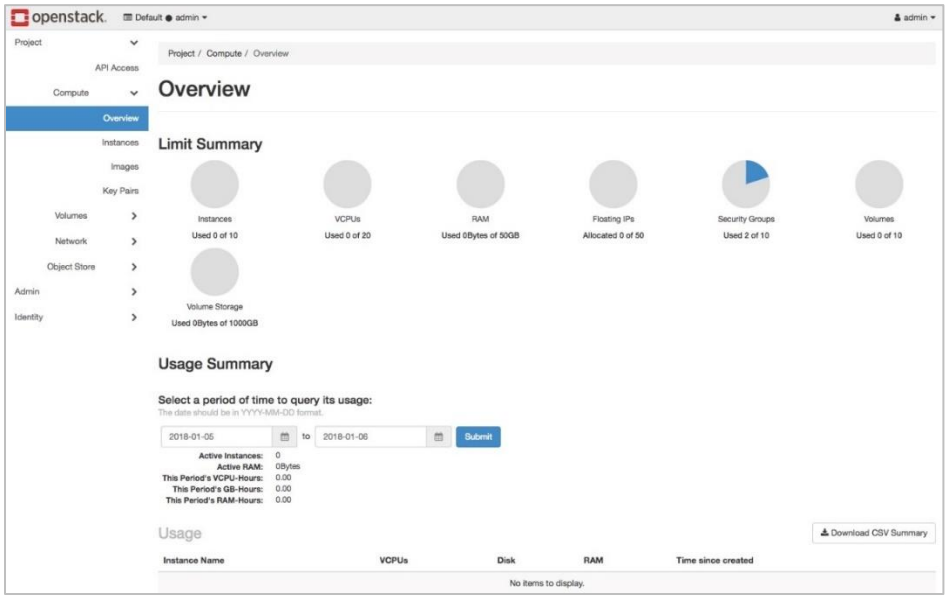

Figure 2.4: Horizon Dashboard *[29]*

2.2.7 Orchestration Service (Heat)

Heat is the orchestration component of OpenStack. It allows describing an infrastructure in the form of models. In Heat, these models are called "stacks". Heat then uses these models to deploy the infrastructure described on OpenStack. He can also use the metrics provided by Ceilometer to decide to create additional instances depending on the workflow of an example application. In OpenStack [14], the Heat service takes care of the arranging and coordination of automated tasks, ultimately resulting in the process workflow for managing the entire lifecycle of infrastructure and applications within OpenStack. Heat Orchestration Template or **Hot template** is the human-readable format code in a text file using YAML language which is a descriptive language like JSON, is used by the Heat service to manage any sequence of activities carried out by the end-user using OpenStack Horizon, theses instructions can be coded in a template and can be accomplished by heat service by processing that template in a heat engine.

2.2.8 Telemetry Service (Ceilometer)

OpenStack Telemetry service is called Ceilometer. It collects various metrics on the use of the Cloud. For example, it allows us to collect the number of instances launched in a project and for how long. These metrics can be used to provide the information needed for a billing system, for example. These metrics are also used in applications or by other components of OpenStack to define actions according to certain thresholds as with the orchestration component, a scenario of that is collect the usage of CPU of an instance, and when it reaches the 80 %, it sends a request to Nova to launch a new instance and share the workload between the two VMs.

2.3 Conclusion

Openstack is leading open source software used to run private clouds. Its popularity has grown exponentially since its founding by Rackspace and NASA. The result of this committed community is amazing, which allows many new features to be found in Openstack with each version. The project is now so large, and it’s so hard to know the details of each service. When working with such a complex project, we will probably encounter problems, bugs, errors. In this chapter, we learned about the design features of Openstack and the essential components of such an ecosystem. We also highlighted the design considerations for Openstack and discussed the various possibilities for extending its functionality. In the next chapter, we discuss how to deploy OpenStack and configure it to deploy the IMS clearwater.

Chapter 3: Deployment of Clearwater and Openstack

3.1 Introduction

In this chapter we give a brief definition of IP multimedia subsystem standard and its architecture, then we present the clearwater IMS architecture and components, we walk through the openstack deployments method and configurations steps. finally, we deploy clearwater on openstack using the All-in-one image and make calls using a Zoiper softphone to register SIP client.

3.2 IP Multimedia Subsystem IMS

IP multimedia subsystem is a standard network architecture which has been defined in the context of next-generation networks to provide IP based multimedia services. Initially, this architecture was designed for mobile network as specified in version 5 of 3GPP, whose main objective is to provide multimedia services to subscribers of the third generation of cellular networks (UMTS). The IMS standard meets the needs of operators offering multiple services on a large scale to many customers with greater flexibility and at a lower cost. [22]

However, IMS has quickly converged to the fixed network to ensure the interoperability of fixed and mobile networks on all IP architecture. IMS uses the application protocol standardized by the IETF, SIP (Session Initiation Protocol) for session signalling and control in addition to SDP. This protocol does not handle the transmission of session data, and IMS uses other protocols such as RTP / RTCP for the transmission of real-time services such as VoIP.

3.2.1.1 IP Multimedia Subsystem Architecture

The IMS architecture decomposes the network into three layers: application layer, control layer and transport layer with standardized interfaces to help scalability, flexibility and extensibility. A simplified architecture is shown in Figure 3.1.

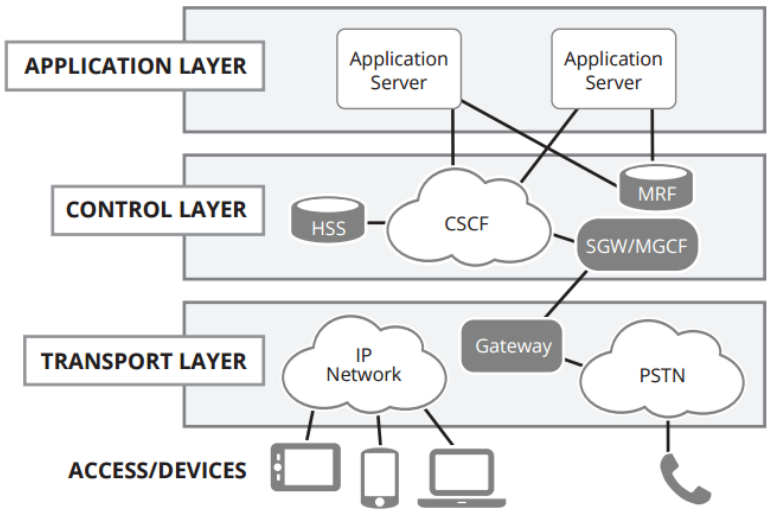

Figure 3.1: An oversimplified IP Multimedia Subsystem architecture *[22]*

IMS enables reliable multimedia communications between several devices across different networks. The modular architecture provides the same unified infrastructure and common mechanisms for controlling, routing and managing sessions, as well as implementing authentication, and accounting controls. The IMS standard covers the widely used Internet Engineering Task Force (IETF) recommendations such as SIP for session control signalling.

The control layer, which is considered the IMS core, is the essential part of the architecture responsible for controlling communications flows. The most functions of the control layer include:

- **Call Session Control Function (CSCF)**: the core IMS architecture, its mission is to control sessions between clients and applications,
- **Home Subscriber Server (HSS)** – The HSS is the main database for all users. It stores user profiles such as their identities, registration information, access settings, authorized services, user location and assigned S-CSCF,
- **Breakout Gateway Control Function (BGCF):** This entity is responsible for selecting the Media Gateway Control Function **(MGCF)** who will route the call to the PSTN,
- **Media Resource Functions (MRF)**: broken down into two elements: MRFC and MRFP which manage SIP communications from and to the S-CSCF and provide the user with the requested resources.
- **Application Server (AS**): provides several multimedia services to the IMS system such as telephony, video conference, third-party registration,…

Many IMS functions can be deconstructed into distinct functional elements. the CSCF function can play many discrete roles: Serving-CSCF (S-CSCF), Interrogating-CSCF (I-CSCF), or Proxy-CSCF (P-CSCF).

3.2.1.2 IP Multimedia Services Advantages and Benefits

IMS provides greater flexibility and extensibility compared to traditional provider network architectures. IMS Services advantages include:

- **Easy to manage**: IMS reduce costs on network administration, because of IP networks are easier to manage (OPEX / CAPEX),
- **Standards-based solutions:** IMS is the only reference architecture for fixed / mobile convergence currently standardized,
- **Service diversity:** allow operators to offer, on mobile phones, services operating over IP.
- **Convergence:** IMS is suitable for both wired and mobile networks. This will, therefore, promote fixed / mobile convergenc,
- **Quality of service:** easier network management and better quality of service.

3.3 Clearwater

Project Clearwater is an open-source IMS core, developed by **Metaswitch** Networks and released under the GNU GPLv3. It follows IMS architectural principles and supports all of the vital standardized interfaces expected of an IMS core network. Unlike traditional IMS implementations, Clearwater IMS was designed from the ground up to be optimized for deployment in virtualized environments and the Cloud. [23]

3.3.1 Clearwater Architecture

The Clearwater IMS framework is based on open-source software component and design patterns for deploying scalable web applications and using these design patterns to fit the constraints of SIP and IMS. The architecture of Clearwater, therefore, has some similarities to the traditional IMS architecture. [24] The following Figure 3.2 illustrates the Clearwater architecture and its components:

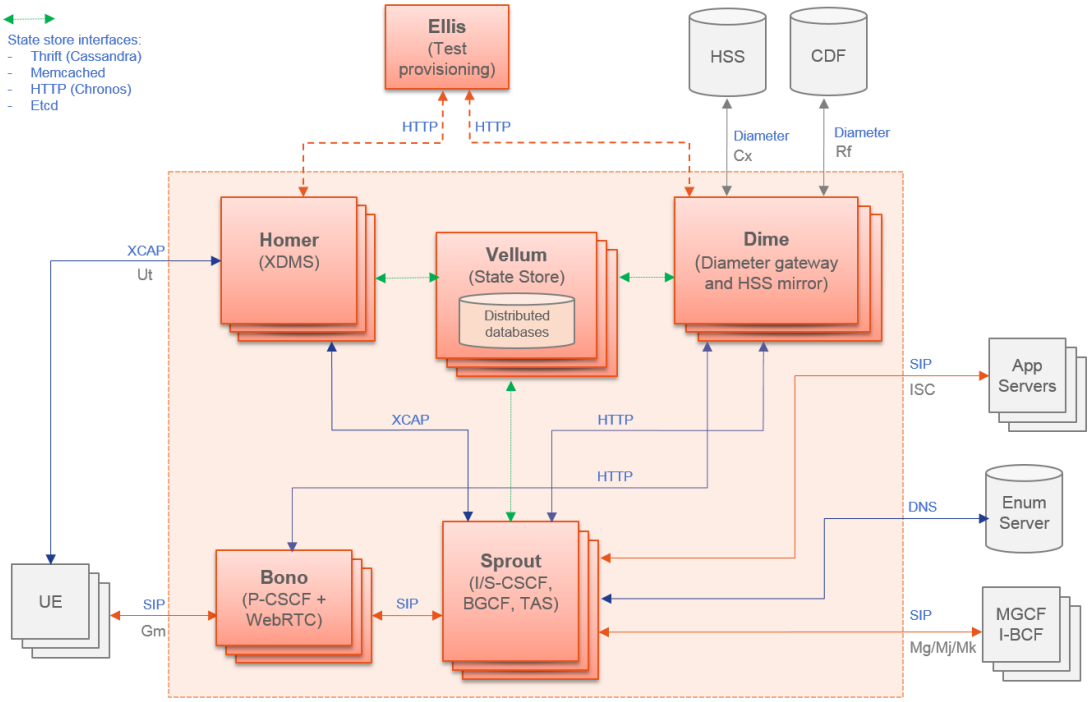

Figure 3.2: Clearwater architecture and its components. *[24]*

3.3.2 Clearwater Component

3.3.2.1 Bono (Edge Proxy)

The Bono node forms a horizontally scalable SIP proxy providing both a SIP IMS P-CSCF compatible interface and a WebRTC interface to clients. Client requests are shared across nodes using load balancing. This node is the client's first point of contact with the Clearwater system,

including support for the various Network Address Translation (NAT) address translation mechanisms for the multiple nodes traversed. A client is therefore attached to a particular Bono node for the duration of registration, but it can move to another Bono node if the connection or the client fails. Clients can connect to Bono using the SIP / UDP or SIP / TCP protocol. Bono supports any WebRTC client that performs call setup signalling using SIP over WebSocket. Bono nodes are not required if Clearwater is deployed with a third-party P-CSCF or SBC (Session Border Controller) which is the implementation of P-CSCF. [24]

3.3.2.2 Sprout (SIP Router)

Sprout nodes act as a horizontally scalable SIP registrar that manages client authentication and the ISC interface for application servers as well as interfacing with the rest of the servers such as Homestead and Homer. Sprout's nodes also contain an option to build an MMTel application server. The Sprout cluster includes a redundant Memcached cluster to store customer registration data with long-term status. SIP Transactions are balanced across the Sprout cluster, and there is no long-term association between a customer and a particular Sprout node. Sprout uses web service interfaces provided by Homestead and Homer to retrieve HSS configuration information such as authentication and user profile data as well as MMTel service settings. [24]

3.3.2.3 Homestead (HSS Cache)

Homestead provides a web services interface for Sprout to retrieve user profile information such as credentials. This component manages data directly through the web interface or extracts it from an HSS through the IMS compliant Cx interface. Homestead nodes function as a cluster using Cassandra to store data.

In the IMS architecture, the HSS mirror function is considered as part of the I-CSCF and S-CSCF components, however, in Clearwater IMS the roles of both the I-CSCF and S-CSCF components are implemented with a combination of a cluster from Sprout and Homestead. [24]

3.3.2.4 Homer (XDMS)

Homer is a standard XML document management server (XDMS) used to store MMtel service parameters in XML documents for each user of clearwater. Documents are managed using a standard XML Configuration Access Protocol (XCAP) interface. Like Homestead, Homer nodes are implemented as a cluster using Cassandra.

3.3.2.5 Ellis

Ellis is a simple provisioning portal that provides user self-registration, password management and control of MMTel service parameters. Ellis is not intended to be in the production part of Clearwater deployments. And horizontally scaling is not easy for this entity due to the use of MySQL. [24]

3.4 Openstack Deployment methods

Openstack can be deployed on a single-node or multi-node, a node is nothing more than a grouping of OpenStack services that run on bare metal, in a container or virtual machine. The

purpose of a node is to provide horizontal scaling, Openstack has four types of node: controller, compute, network and storage. There are several Openstack Deployment methods out there each one for training labs and testing purpose, some of these are:

3.4.1 DevStack

DevStack is a sequence of scripts used to quickly deploy a complete OpenStack environment based on the latest version. It is used as a development environment and integrations of new entities and project on the Openstack as the basis for much of the OpenStack project's functional testing [25].

3.4.2 RDO Packstack

Packstack is a tool that uses Puppet modules to deploy several components of OpenStack on servers over SSH automatically. Currently, only Red Hat Enterprise Linux (RHEL), CentOS, and their derivatives are supported. [26]

3.4.3 TripleO

TripleO also is known by (Openstack on Openstack) is a project intended to install, upgrade and operate OpenStack clouds using OpenStack's cloud facilities as the foundation - building on Nova, Ironic, Neutron and Heat to automate cloud management at datacenter scale. [27]

3.5 Test Architecture and Design

In our study, we use the Packstack RDO deployment method to deploy OpenStack on a single machine. This method has some software and hardware prerequisites [28]:

Software:

- Red Hat Enterprise Linux (RHEL) 7and CentOS 7 are the minimum recommended version with x86_64 architecture.

Hardware:

- A machine with 16GB RAM at least, CPU with nested virtualization enabled, and at least one network adapter.

Network

A static IP address with an internet connection to install the packages.

To test the functionality of the IMS architecture in a cloud computing platform, we deployed a test environment at the Algeria Telecom company. The study is based on the deployment of Openstack as infrastructure IaaS for the free solution Clearwater IMS.

3.5.1 Hardware Configuration

	Operating System	CPU	Memory
Physical server	Centos 7.8	i7	24 GB

3.5.2 Openstack Installation Steps

1- disabling **NetworkManager** and firewall services and enable the network service

```
$ sudo systemctl disable firewalld
$ sudo systemctl stop firewalld
$ sudo systemctl disable NetworkManager
$ sudo systemctl stop NetworkManager
$ sudo systemctl enable network
$ sudo systemctl start network
```

2- checking the list of the Openstack version available on centos-release

```
$ sudo yum list centos-release-openstack*
```

- Installing the appropriate version

```
$ sudo yum install -y centos-release-openstack-*
```

- Updating the list of packages:

```
$ sudo yum update -y
```

3- Installing Packstack Installer

```
$ sudo yum install -y openstack-packstack
```

4- get the interface name linked with IP address

```
$ ip addr show
```

- Customizing and generate the answer-file

```
$ sudo packstack --allinone --default-password admin
      --os-neutron-ovs-bridge-interfaces=br-ex:interface-name
      --os-neutron-ml2-tenant-network-types=vxlan
      --os-neutron-ml2-mechanism-drivers=openvswitch
      --os-neutron-ml2-type-drivers=vxlan,flat
      --os-neutron-l2-agent=openvswitch
      --os-heat-install=y
      --os-heat-cfn-install=y
      --gen-answer-file=demo.txt
```

It generates an answer file with several options enabled, this lets us customize the installation of OpenStack as enabling and disabling some services, the supported network types, the password for the admin, the size of the backend storage, etc.

- Installing the Openstack from the answer-file

```
$ sudo packstack --answer-file=./demo.txt
```

- Now the installation begins this takes about 30-45 mins depending on the internet speed

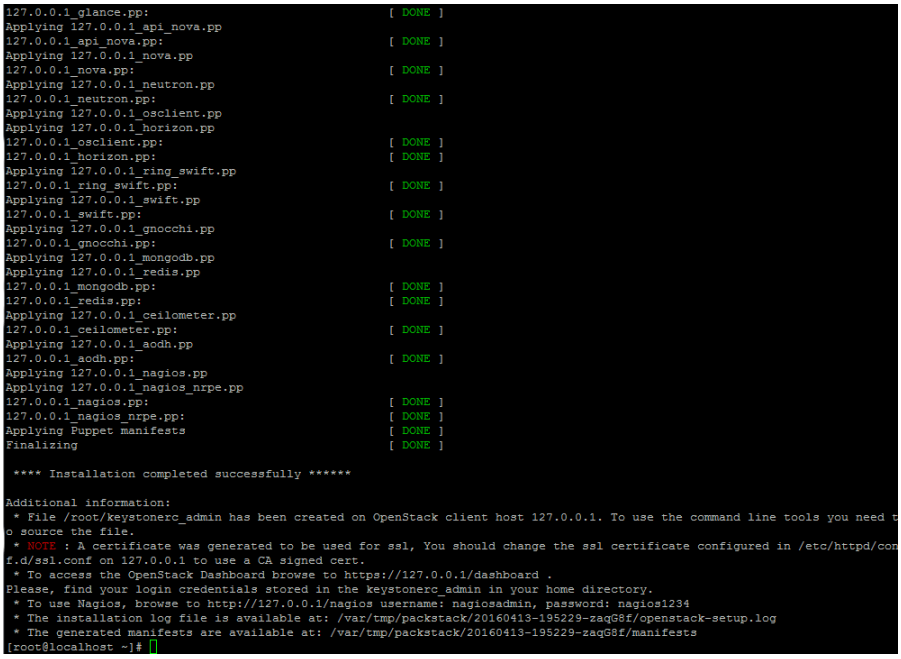

Figure 3.3: OpenStack Installation Completed *[33]*

Once the process is complete, we can log in to the OpenStack web interface Horizon by going to https://$IPaddress/dashboard. The user name is admin. The password is in the **keystonerc_admin** file in the /root directory of the control node. Figure 3.4 shows the login page, and Figure 3.5 shows the horizon dashboard

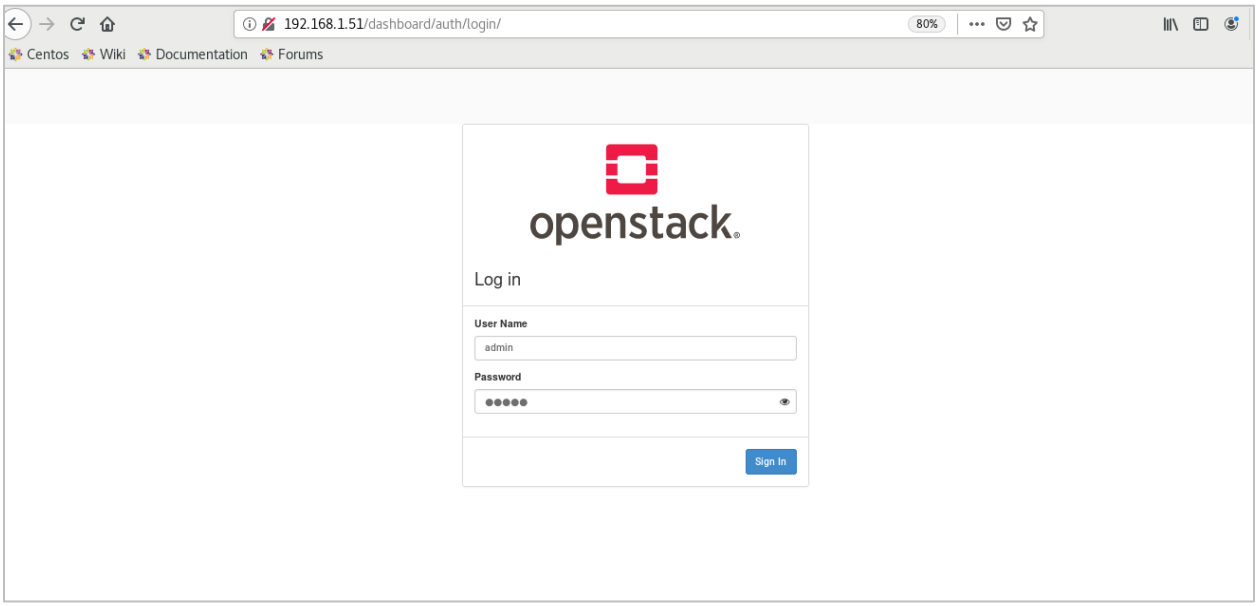

Figure 3.4: Openstack login page

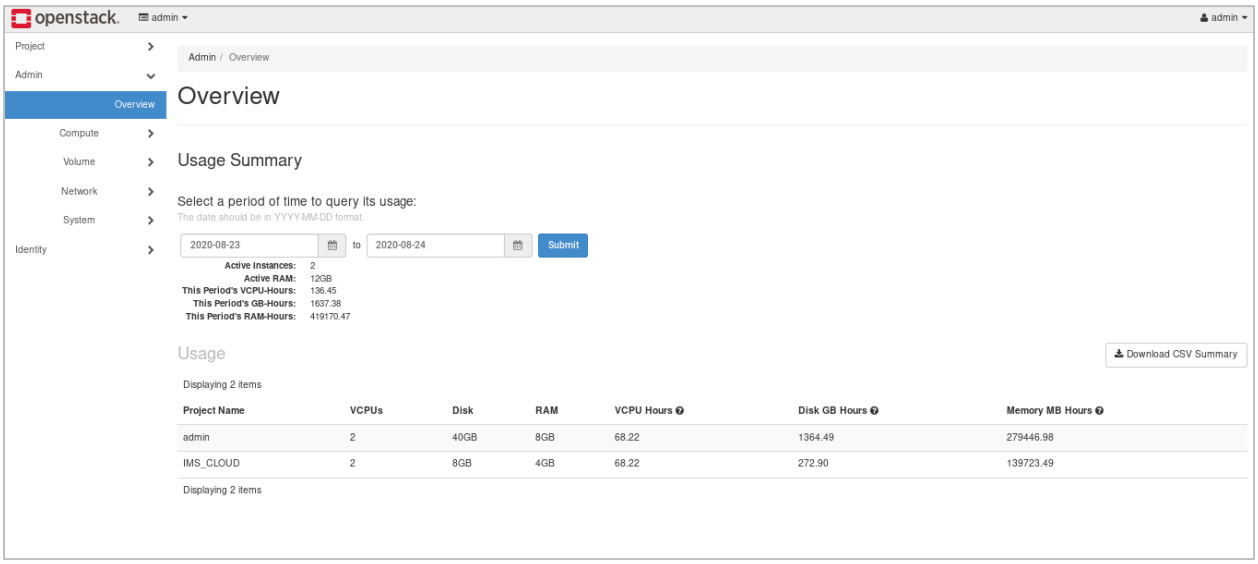

Figure 3.5: Openstack dashboard

3.5.3 Configuration of Openstack

Once the installation is complete, we configure and check the network configuration:

- Checking that interface is related to the **br-ex** bridge based on our installation. there must be two files one is related to the interface and the other to the bridge:

  In the path ***/etc/sysconfig/network-script/ifcfg-interface-name*** (ex: ens192) this file looks like:

```
TYPE=OVSPort
NAME=interface-name
DEVICE= interface-name
DEVICETYPE=ovs
OVS_BRIDGE=br-ex
ONBOOT=yes
```

- The external bridge settings look like below, In the path ***/etc/sysconfig/network-script/ifcfg-br-ex***

```
DEVICE=br-ex
NAME = br-ex
DEVICETYPE=ovs
TYPE=OVSBridge
BOOTPROTO=static
IPADDR=IP@
NETMASK=mask
GATEWAY=gateway_IP
IPV4_FAILURE_FATAL=no
IPV6INIT=no
ONBOOT=yes
```

- The configure ***/etc/neutron/plugins/ml2/ml2_conf.init*** *file* by adding the missed drivers list (**vxlan,flat,vlan,gre,local,geneve**) in type_drivers attribute

```
[ml2]
type_drivers=vxlan,flat,vlan,gre,local,geneve
tenant_network_types=vxlan
mechanism_drivers=openvswitch
path_mtu=0
extension_drivers=port_security,qos
[securitygroup]
enable_security_group=True
firewall_driver=neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
[ml2_type_vxlan]
vxlan_group=224.0.0.1
vni_ranges=10:100
[ml2_type_flat]
flat_networks=*
```

- The configuration of ***/etc/neutron/plugins/ml2/openvswitch_agent.conf*** *file* looks like this by changing the default value of l2_population=False to l2_population=True

```
[ovs]
bridge_mappings=extnet:br-ex
integration_bridge=br-int
tunnel_bridge=br-tun
local_ip=IP-address

[agent]
l2_population=True
drop_flows_on_start=False
tunnel_types=vxlan
vxlan_udp_port=4789

[securitygroup]
firewall_driver=neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

- Now our network is configured we can start by creating our provider network, we can do that by Horizon dashboard or by CLI client
- First, we source the **Keystonerc** file to get the Openstack admin privileges

```
# source keystonerc_admin
```

- We can now use the Openstack commands with the admin privileges.
- We can test that by running this command that lists the networks available in the Openstack.

```
# openstack network list
```

Create the provider network named ***external_network*** with a network type flat and map it with the physical_network named ***extent*** (in ***openvswitch_agent.conf*** file **bridge_mappings=extnet:br-ex**)

```
# neutron net-create external_network --provider:network_type flat -
-provider:physical_network extnet  --router:external
```

- We need to create a public subnet with an allocation range outside of the external DHCP range and set the gateway to the default gateway of the external network.

- 192.168.1.1/24 is the router and CIDR we defined in ***/etc/sysconfig/network-scripts/ifcfg-br-ex*** for external connectivity.

```
# neutron subnet-create --name public_subnet --
enable_dhcp=False --allocation-pool=start=192.168.1.100,
end=192.168.1.130 \
```

- The provider network is configured and active and is ready to be attached to instances

We can also use the graphical user interface Horizon to manage our networks

- In Horizon dashboard, using the admin credentials, under the admin tab go to network tab, we create a new network

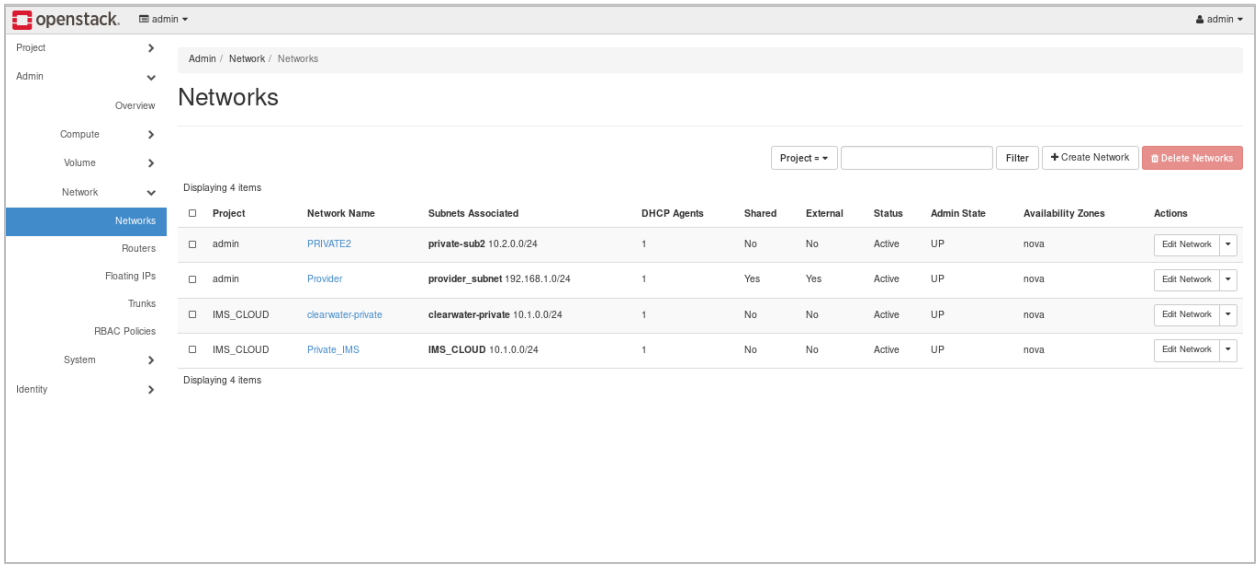

Figure 3.6: Network tab management

- Fill the field with our network configuration in the network, subnet, and details tabs

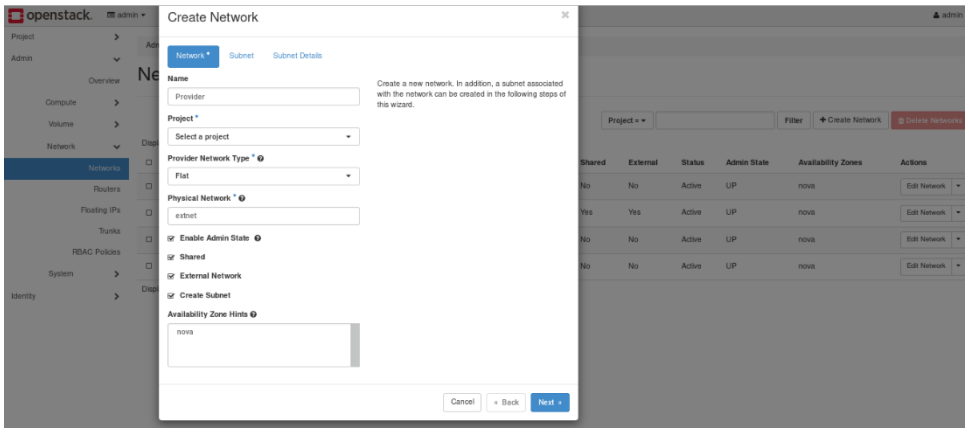

Figure 3.7: Provider Network Creation

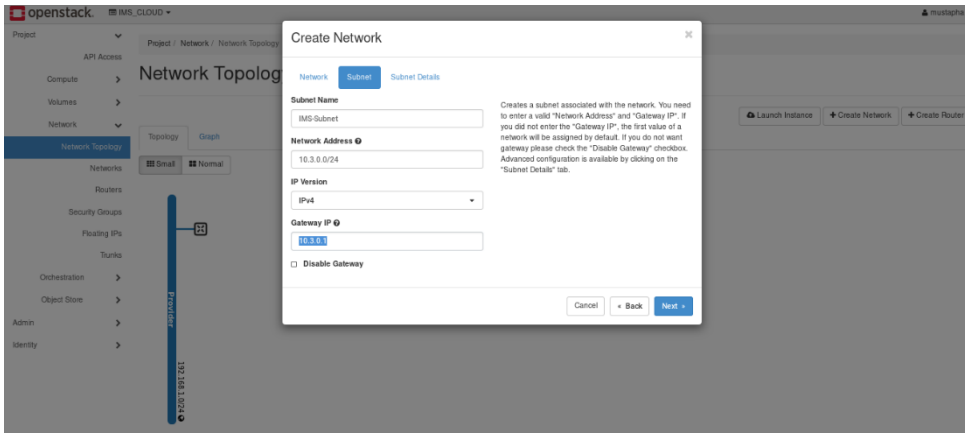

Figure 3.8: Subnet Setting for the Provider Network

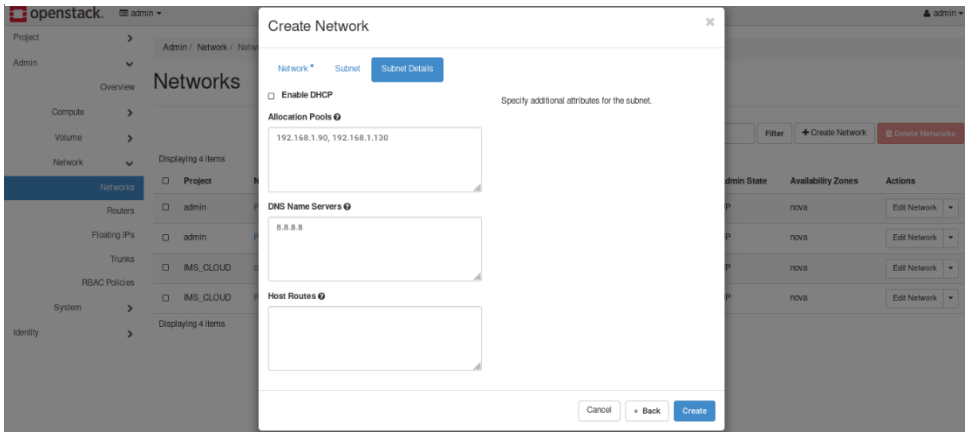

Figure 3.9 Network DHCP configuration

- Now the provider network is configured and we can now use it.

3.5.4 Clearwater Deployment

There are several methods out there for deploying Clearwater:

- **All-in-one Images:** can be installed on EC2 or the used virtualization platform, as long as it supports OVF (Open Virtualization Format).
- **An automated install using the Chef orchestration system**: This is the recommended install for spinning up large scale deployments since it can handle creating a sized deployment in a single command.
- **A manual install:** using Debian packages and hand configuring each machine. This is the recommended method if **Chef** is not supported on the virtualization platform.
- **Installing from source:** If running on a non-Ubuntu-based OS, or need to test a code fix or enhancement we made, we can also install Clearwater from source. Per-component instructions are provided that cover the route from source code to running services. Familiarity with performing a manual install on ubuntu will help with configuring the network correctly after the software is installed.

We use the All-In-one OVF deployment image of clearwater for our study.

3.5.4.1 All-in-one Image

While Clearwater is designed to be massively horizontally scalable, it is also possible to install all Clearwater components on a single node. This makes installation much simpler and is useful for familiarizing ourself with Clearwater before moving up to a larger-scale deployment using one of the other installation methods.

All-in-one images consist of:

- Ubuntu 14.04, configured to use DHCP
- Bono, Sprout, Homestead, Homer and Ellis
- Clearwater auto-configuration scripts.

On boot, the image retrieves its IP configuration over DHCP, and the auto-configuration scripts then configure the Bono, sprout, homestead, homer and Ellis software to match.

The image is designed to run on a virtual machine with a single-core, 2GB RAM and 8GB of disk space.

3.5.4.2 All-in-one OVF Installation Process

This process should work on any virtualization platform that supports OVFs using x86-64 CPUs. We only tested it on

- VMware Player
- VirtualBox
- VMware ESXi.
- KVM

- To install the OVF, we must first download it from and save it to disk.

```
$ curl http://vm-images.cw-ngv.com/cw-aio.ova
```

Then, we must import it into our OpenStack platform. To do that we must convert the **cw-aio.ova** file into **qcow2** format. To do that we need to extract the ova file, This outputs three files

```
$ tar -xvf cw-aio.ova
cw-aio-disk1.vmdk
cw-aio.ovf
cw-aio.mf
```

Now we can convert the **vmdk** file using the **qemu-img convert** command to a qcow2 image file which is supported by OpenStack

```
$ qemu-img convert -f vmdk -O qcow2 cw-aio-disk1.vmdk cw-aio.qcow2
```

Upload the image into Openstack, in the dashboard under the image tab of the Compute section create an image give it a name, and a disk type of qcow2

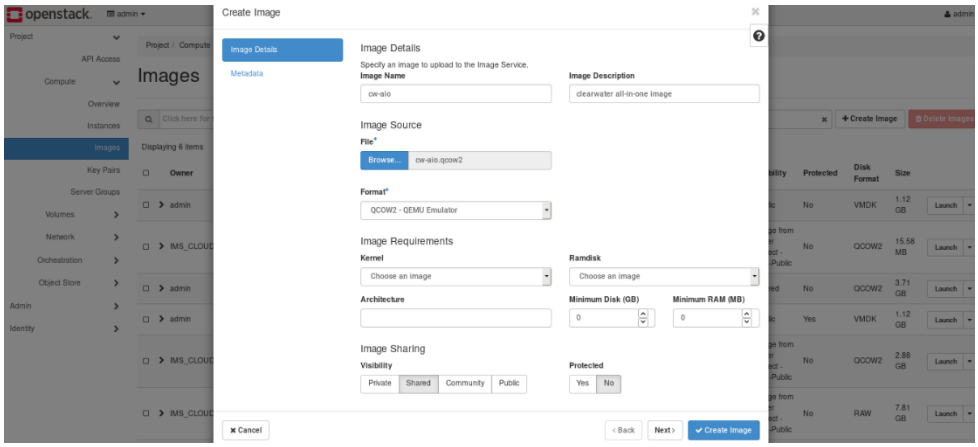

Figure 3.10 Create an image for cw-aio

Before launching an instance using the image created, we must create an SSH key for our instance, so we can access it.

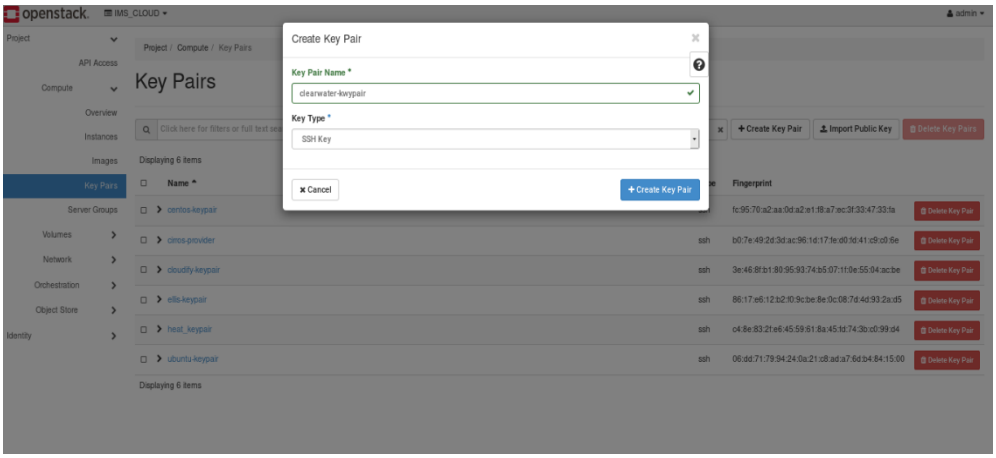

Figure 3.11: Create an SSH keypair

Now we launch a new instance, we give it a name like `clearwater-all-in-one`.

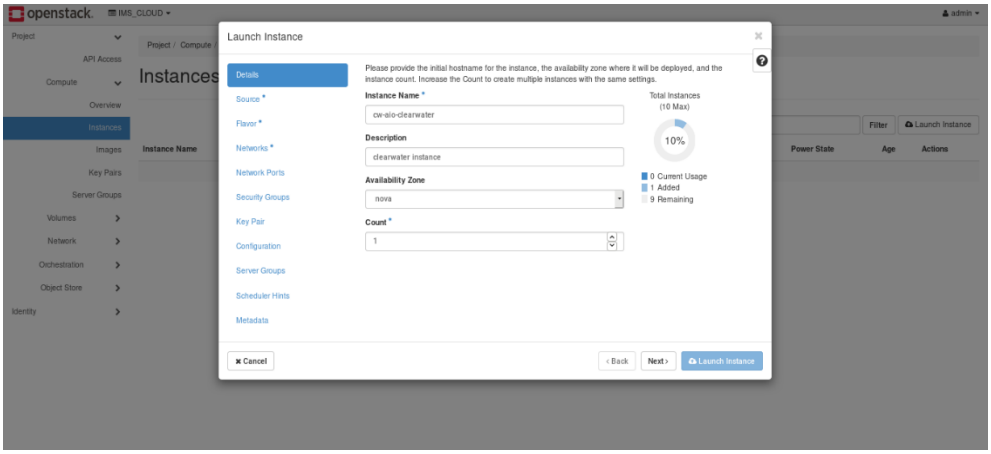

Figure 3.12: Launching a New Instance

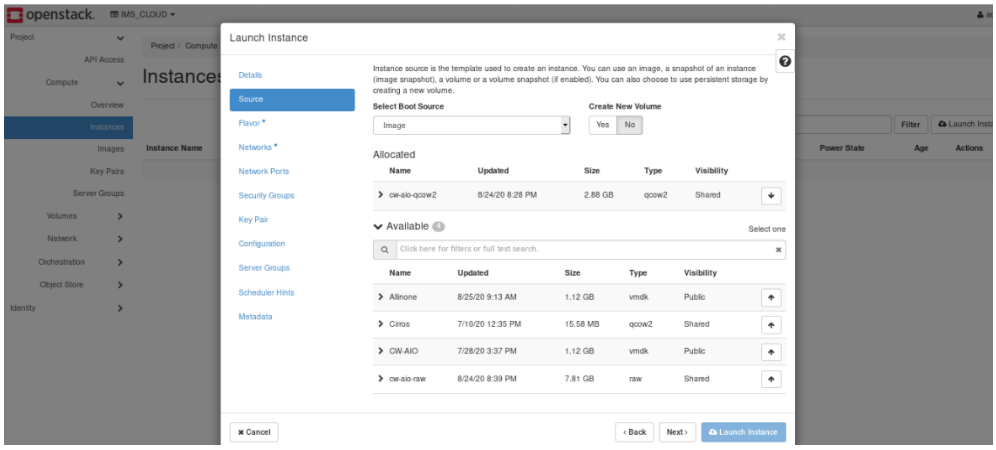

Figure 3.13: Choose an Image

A flavor with (1vCPU and 2GB of RAM and 8 GB of disk) will be sufficient.

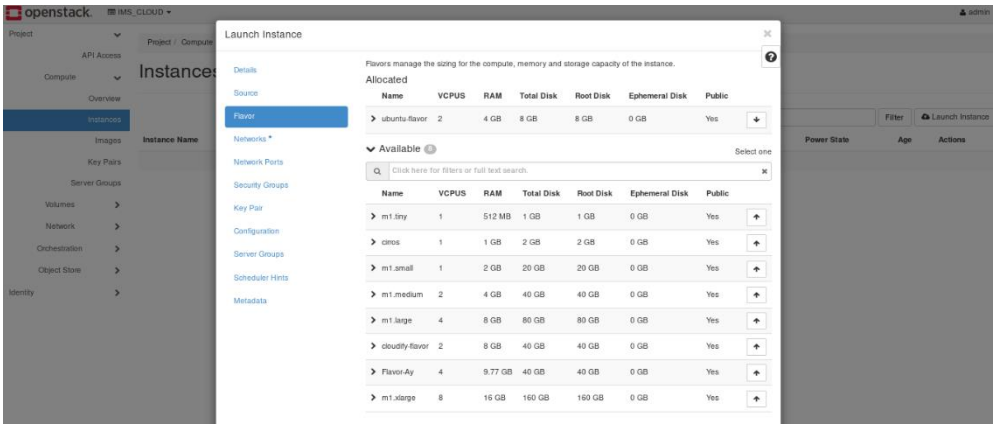

Figure 3.14: Choose a Flavor

Choosing a network to be attached to

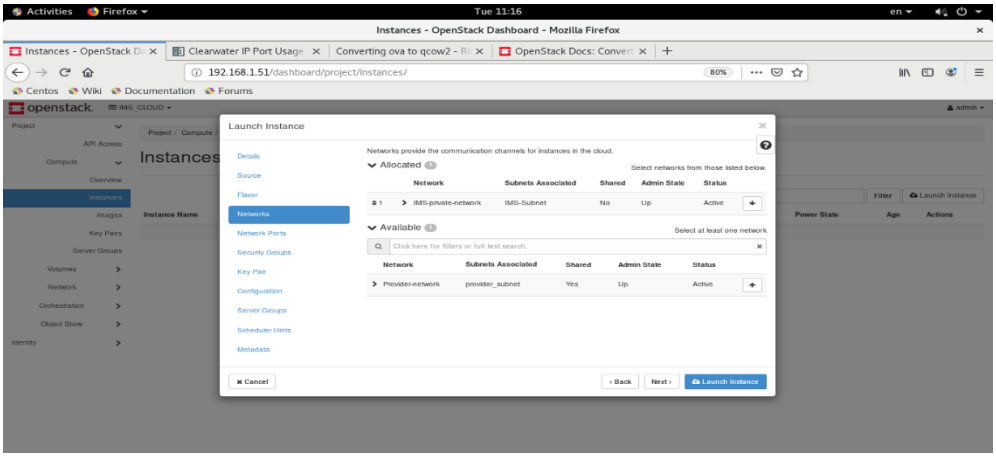

Figure 3.15: Select a Network

Selecting a security Group: which is a set of rules that will be applied to the instance to control the traffic going in and out of the instance like allowing the ssh connection, the ICMP protocol...

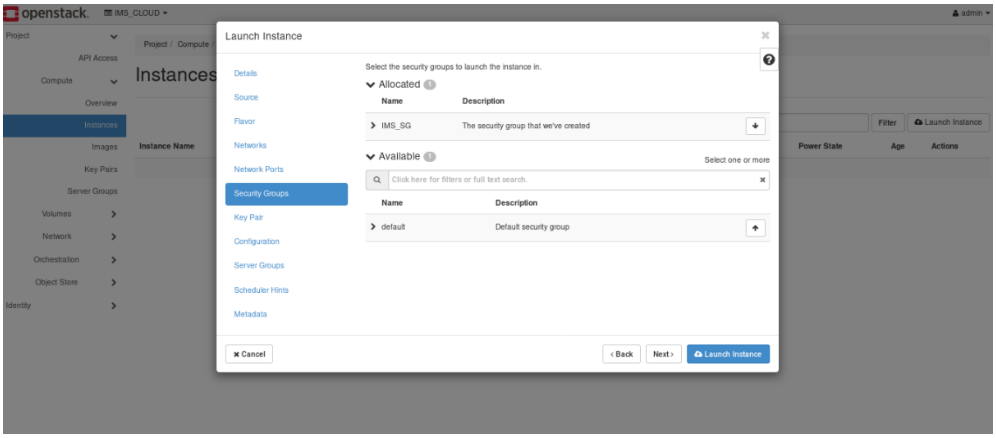

Figure 3.16: Select a Security Group

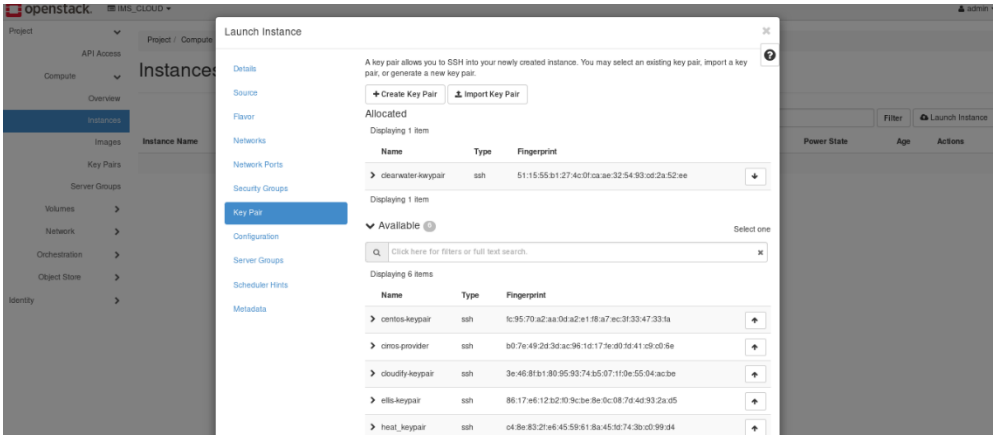

Figure 3.17: Select a keypair

Now we can launch the machine this takes a minute to be ready

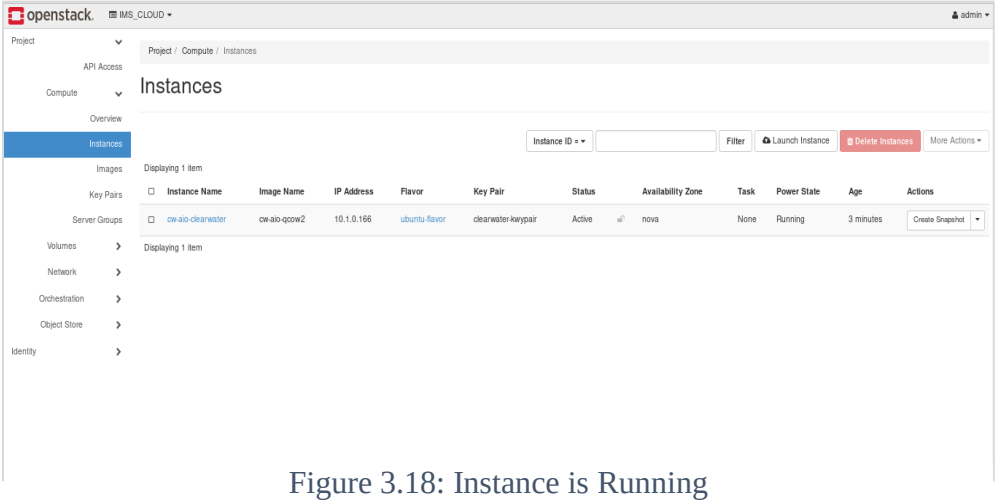

Figure 3.18: Instance is Running

We can see that an IP address was assigned to the instance from the ***ims-private*** network pool, but this address is not accessible from the outside, for that we need another type of IP address named **floating IP** which is like a public IP accessible from the outside this address is in the same network as the external network, so we can access the instance from the outside. A NAT mechanism is implemented on the Openstack virtual router to forward the traffic to the specific instance.

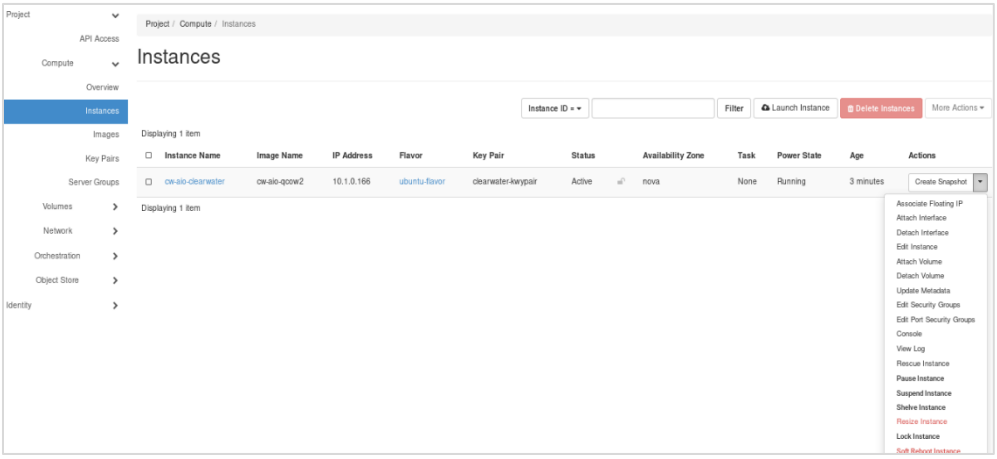

Figure 3.19: Associate a Floating IP

Now the machine is ready to be connected to, let's test the connectivity of the instance we can do that by:

- We can ping the floating IP of the instance.
- SSH to the instance using the floating IP, the username is **ubuntu** and password is **cw-aio**
- Access the Ellis user interface by pointing the browser at https://floating-ip

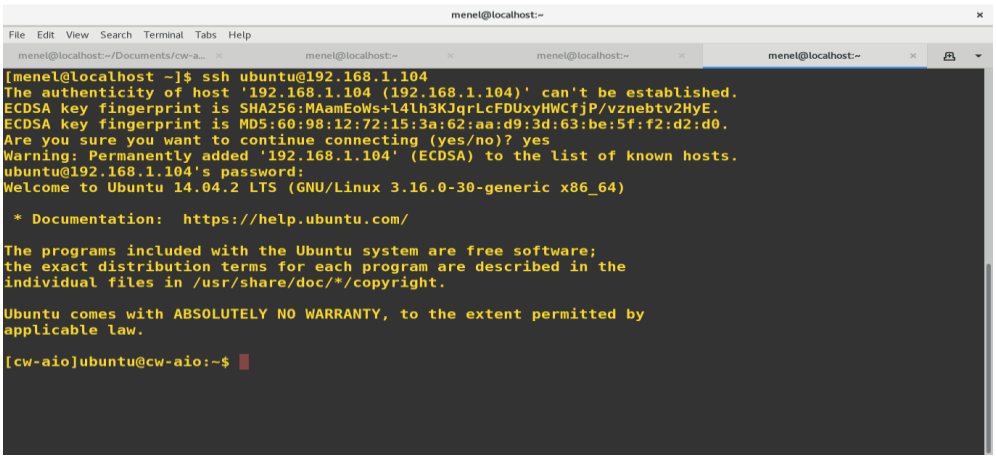

Figure 3.20: SSH to the instance

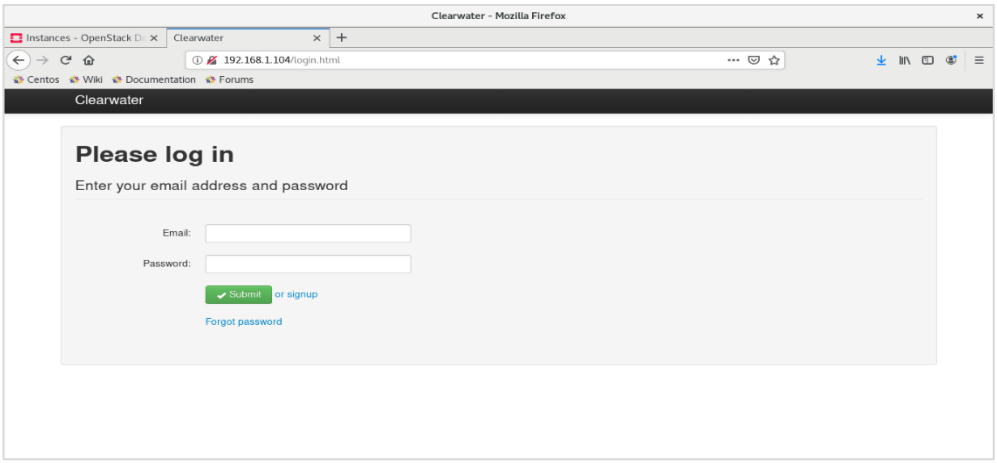

Figure 3.22: Ellis Web UI

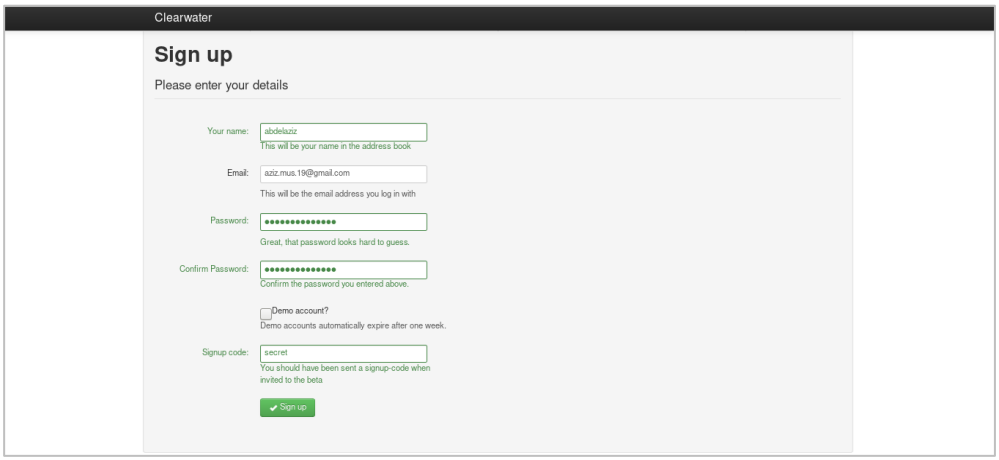

Figure 3.21: Ellis Sign up page

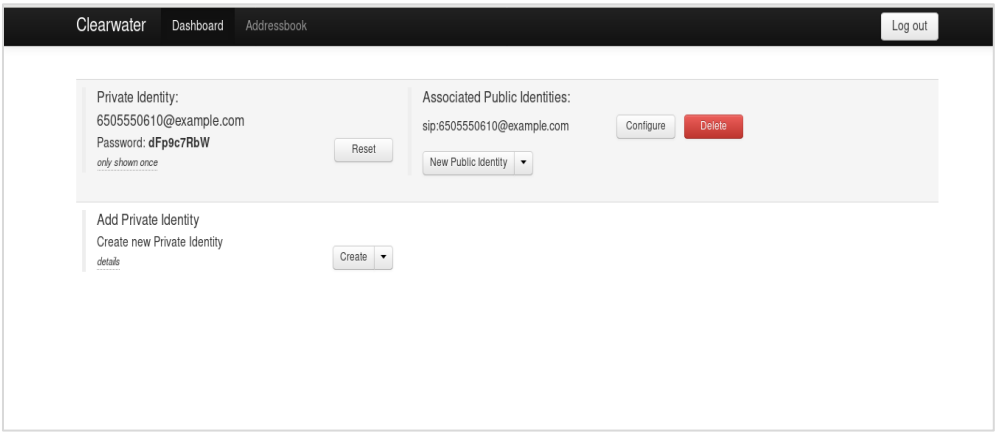

Figure 3.23: Ellis portal

3.5.5 Making calls through Clearwater

To make calls through Clearwater, we need to register at least two SIP clients. In our study, we used four SIP client, two Android phones and two laptops running Windows 10 using the Zoiper softphone.

Launch the Zoiper client and enter the username, e.g. **`6505550352@example.com`** and password, e.g. **`HVQENwddx`** provided by Ellis, the password is displayed once, and we have to hit the reset button to regenerate a new one, the domain name is **`example.com`**

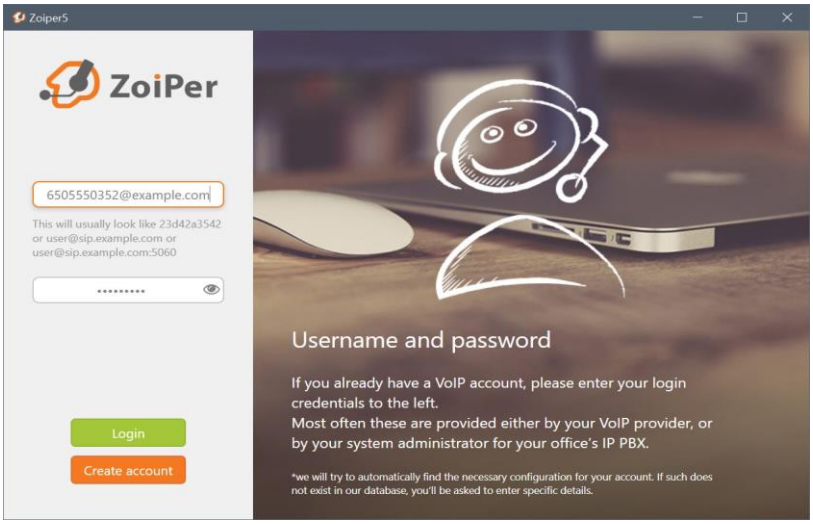

Figure 3.24: The Client Username and Password

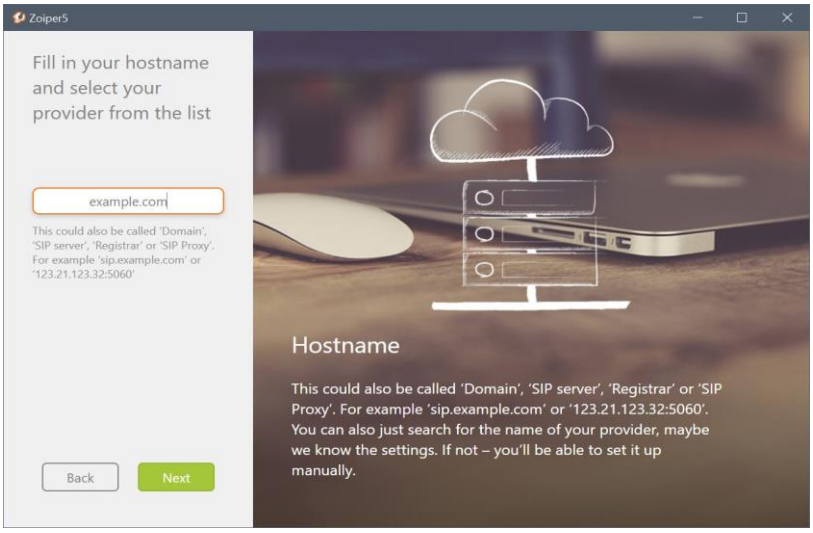

Figure 3.25: domain name

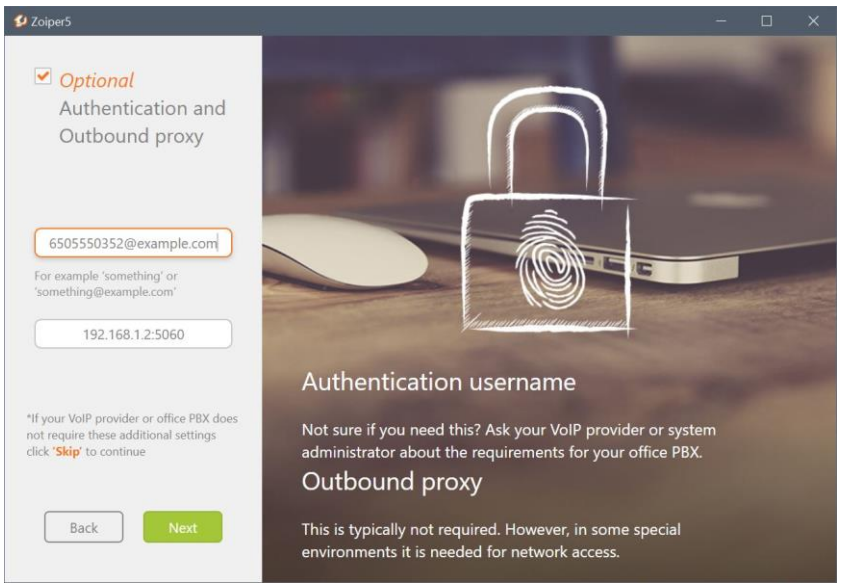

Figure 3.26: Authentication and Outbound proxy

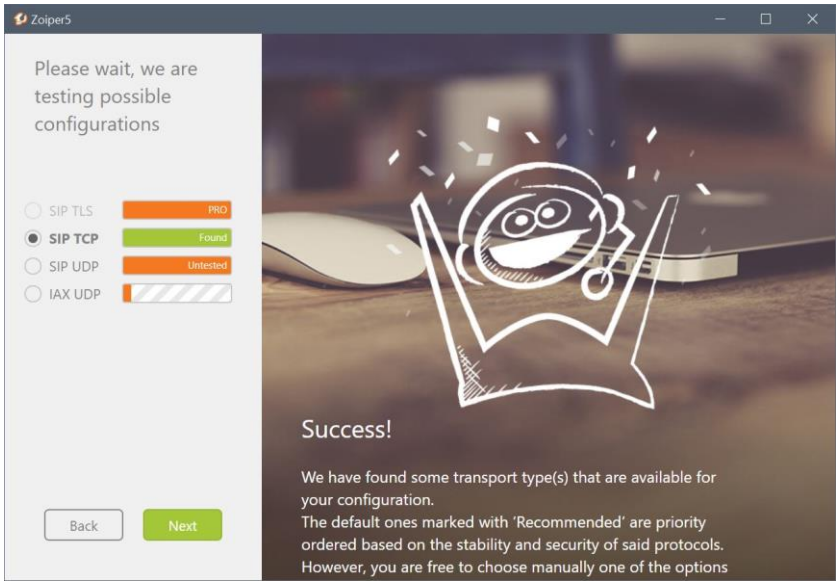

Figure 3.27: Registration succeeded

One the registration succeeded we can now configure a second SIP client and make calls

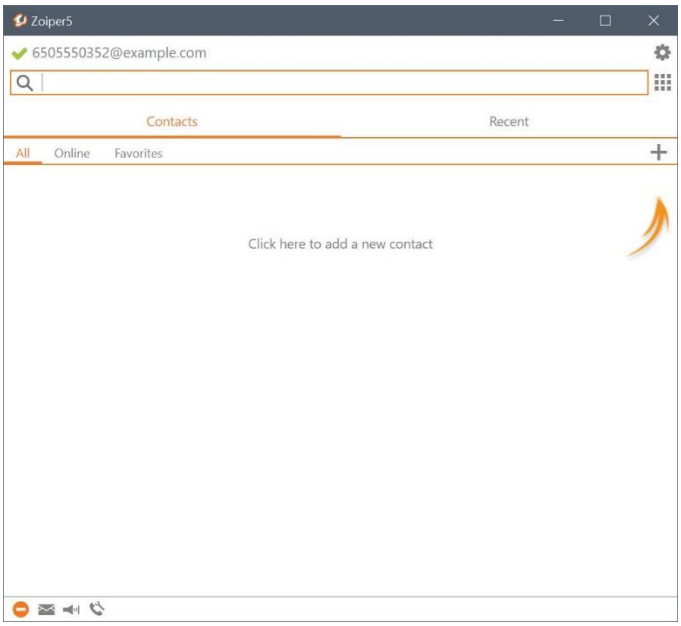

Figure 3.28: Zoiper main window

3.5.6 SIP Flow Messages

3.5.6.1 REGISTER Method

In order To connect to Clearwater, the end-user must first connect to the network: this process is called "registration" which is mainly based on the exchange of SIP signalling messages:

- UE (User Equipment) sends a first registration request to the P-CSCF proxy which resides on the Bono Node using a "REGISTER" message.
- The P-CSCF must first verify the identity of the end-user against their profile stored in the database for that it needs the I-SCSF and S-SCSF functions which reside on the Sprout node.
- Sprout needs authentication data of the user stored on the homestead node, so it sends an HTTP GET request to homestead to retrieve authentication data.
- Homestead sends a multimedia-auth-request to the HSS database to get the subscriber authentication data, and HSS sends a multimedia-auth-response.
- Homestead send an HTTP 200 OK response to sprout with the authentication data of the user.
- Sprout finds no authorization included in the request of the user, and store the digest received from the homestead in his cache, so it doesn't need to re-query homestead again.
- Sprout sends a SIP 401 unauthorized message to the UE.

- After this verification, the UE sends a second "REGISTER" registration message with authentication digest.
- The Sprout finds the authorization acceptable on the request using the cached digest.
- Sprout accepts the registration request and retrieves the subscriber profile, and sends a 200 OK to UE.

Figure 3.29 below shows a packet capture during the registration process using Wireshark (Version 3.2.6).

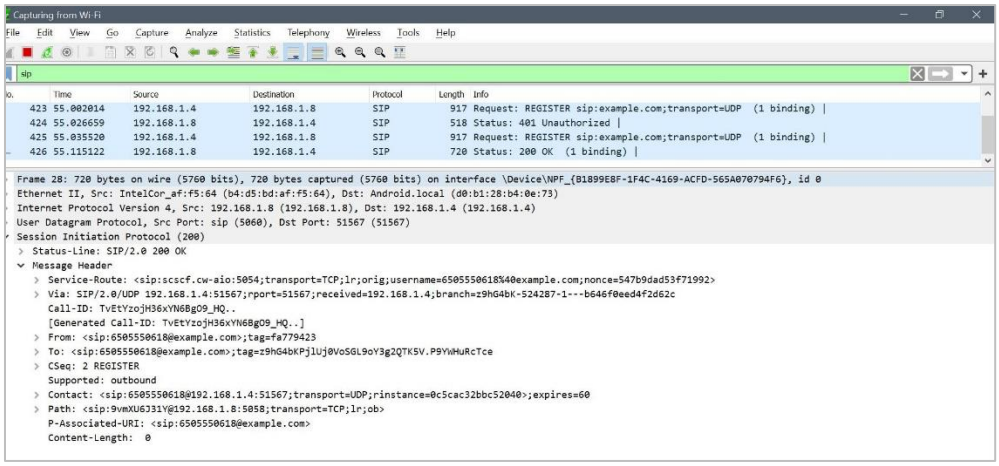

Figure 3.29: Registration flow in Wireshark

3.5.6.2 INVITE Method

Now that the user <sip: 6505550618@example.com> has been registered in Clearwater, the user could perform a voice session. Finally, the client must initiate a session by sending the "INVITE" request, which allows a client to request a new session.

This SIP method is used both to initiate a voice session for an already registered client. An example of an "INVITE" request sent by UE1 to initiate a phone call with UE2 is shown in Figure 3.30. The parameters of the session are negotiated using the SDP protocol, the message of which will be encapsulated in the SIP "INVITE" request.

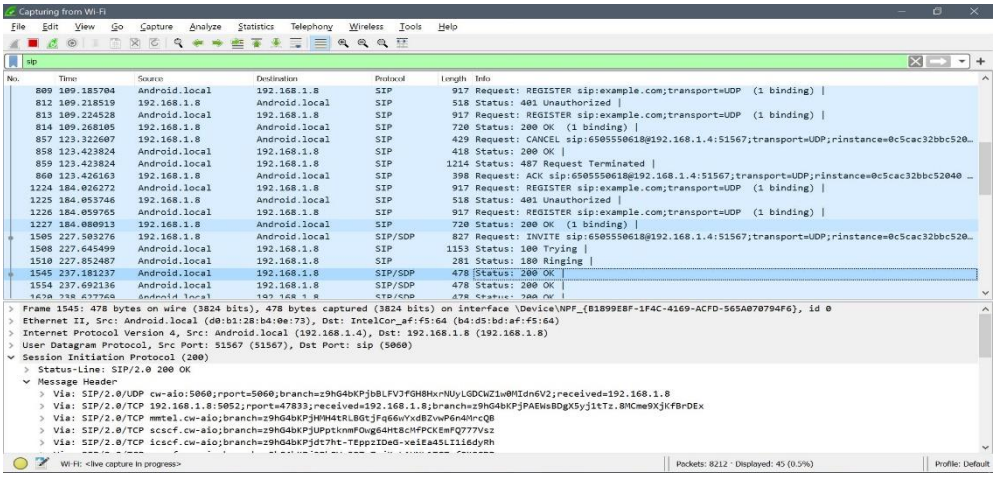

Figure 3.30: INVITE method captured in Wireshark

The audio session is an example of establishing a session. The two participants (UE1 and UE2) negotiate the parameters of the session (quality, media encoding and capabilities of the terminal to be used), as well as the reservation of resources by "SIP / SDP" messages. Sample SDP message is shown in Figure 3.31 When all the parameters of the session are negotiated, and the two UEs exchange a "200 OK" message, then they exchange an ACK message which indicates that the creation of the voice session was successful.

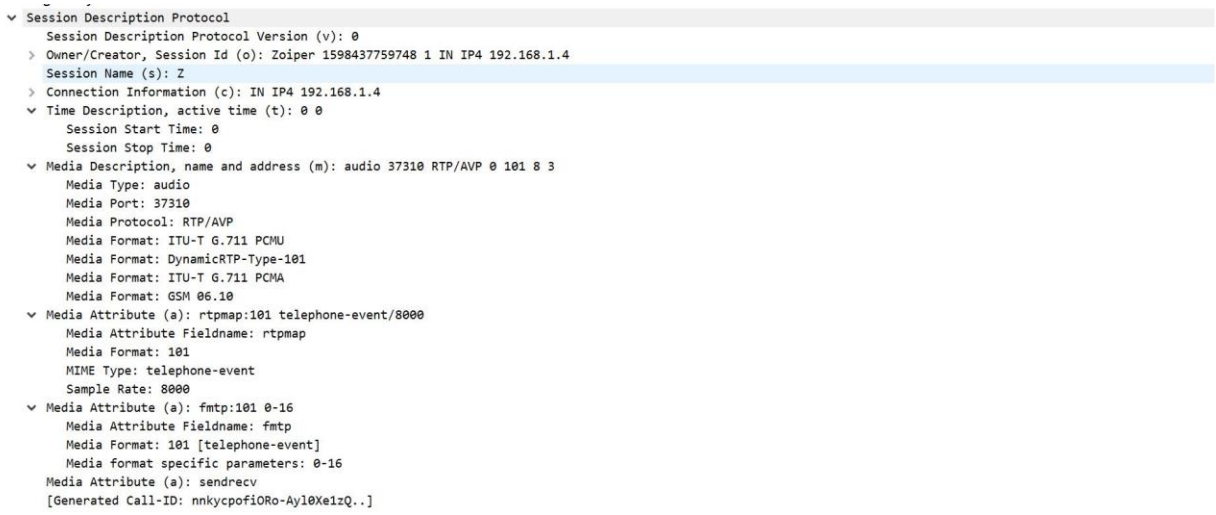

Figure 3.31: Session Description Protocol SDP

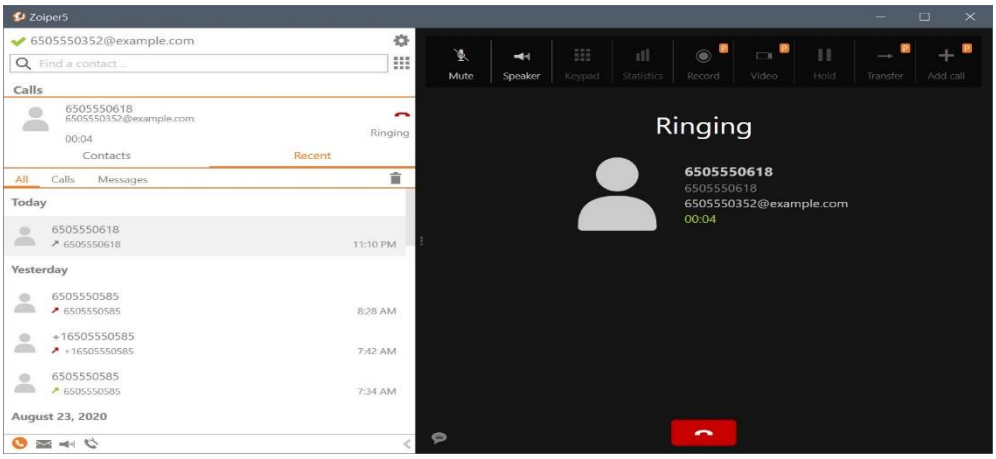

Figure 3.32: Call Ringing

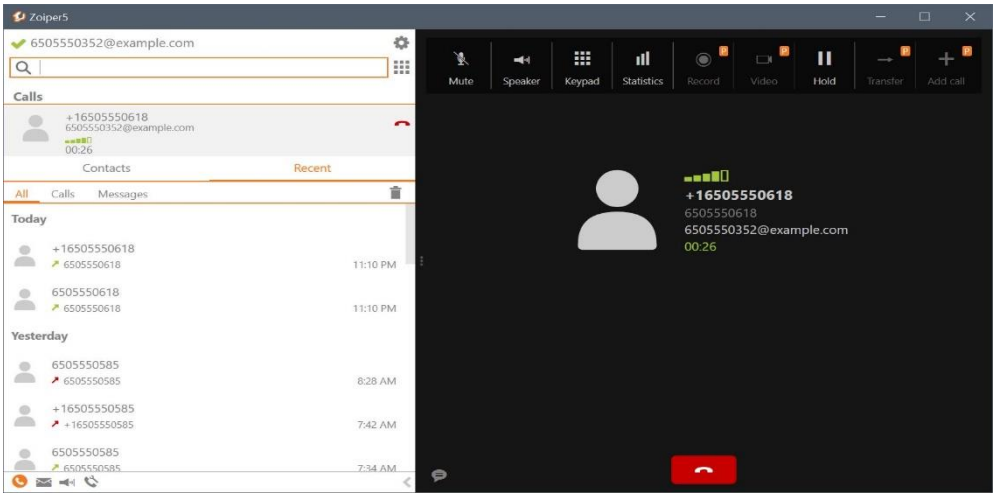

Figure 3.33: Client Answered

3.6 Conclusion

In this chapter, we defined a general idea about the IP Multimedia Subsystem IMS which has been defined in the context of next-generation networks to provide IP based multimedia services and its architecture, exploring its different components and the benefits of such a system. We also discovered An Implementation of the IMS, which is the opensource IMS Clearwater and its components and architecture. We successfully deployed and configured the infrastructure as a Service Openstack by using the Packstack installation method, and we prepared the provider and private network for our clearwater. For clearwater deployment, we choose the all-in-one OVF image to deploy clearwater on Openstack, we successfully registered several clients using Zoiper softphone and made some calls provisioning the registration and call flow in Wireshark.

General Conclusion

Cloud computing is changing the way we deliver and create services through the Internet. By giving organizations a flexible and extensible approach to build and manage its IT infrastructure as well as its IT services, that can increase the productivity of companies, reduce hardware and deployment costs. Which add a new value to the company, improve the QoS profits and increase revenues.

One of the main problems in telecom operators was the resource management problem, which relies on how to estimate hardware needed to offer the best QoS for the system. Which is known as overuse and underuse of resource, with Cloud Computing characteristics this problem has become easier to handle in a programmatical way and at a lower cost.

The move to cloud services offers various benefits not only for operators but also for end users. Thus, through the cloudification of services and equipment, telecom operators will better manage network resources and quickly integrates new services with less human intervention and at a competitive cost.

At the beginning of our graduation project, we prepared an Infrastructure as a Service for the private Cloud of Algeria telecom using one of the big opensource projects in the world. Openstack is supported and developed by big companies which make it a powerful and extensible tool for managing infrastructures.

Then we took an opensource IMS cloud solution Clearwater IMS and deployed it in Openstack platform. It's an IMS core who follows the IMS architectural principles, it was designed from the ground up to be optimized for deployment in virtualized environments and the Cloud.

In our study, we have tested the deployment of Openstack and clearwater on a dedicated server at Algeria Telecom company and provision voice call using free softphones to test the calls between two users.

We finished our project, but IMS deployment on the cloud is a long-term project. It would be interesting to extend the test environment to:

- Deploy Openstack on multi nodes using one of the available methods and integrate other Openstack components to improve our environment,
- Deploy Clearwater using the multi-nodes mode to improve the performance, fault-tolerance, scalability, and QoS,
- It will be interesting to add other multimedia services such as chat, video calls, video conference and IP TV etc.
- Add more functionalities to calls such as billing, and call forwarding etc.
- Deploy multiple IMS solutions and test media services between those IMS core.

References

[1] P. Mell, T. Grance, “The NIST Definition of Cloud Computing,” NIST, 09 2011. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-145/final. [Accessed 25 03 2020].

[2] S. Maddie, Beginning Serverless Computing, Richmond, Virginia, USA: Library of Congress, 2018.

[3] Redhat, “Understanding virtualization,” Redhat, [Online]. Available: https://www.redhat.com/en/topics/virtualization. [Accessed 14 05 2020].

[4] K. Scarfone M. Souppaya P. Hoffman, “Guide to Security for Full Virtualization Technologies,” 01 2011. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-125/final. [Accessed 03 04 2020].

[5] Ciena, “What is SDN,” Ciena , [Online]. Available: https://www.ciena.com/insights/what-is/What-Is-SDN.html. [Accessed 12 08 2020].

[6] Redhat, “What is NFV,” Redhat, [Online]. Available: https://www.redhat.com/en/topics/virtualization/what-is-nfv. [Accessed 02 07 2020].

[7] IBM, “Data Centers,” IBM Cloud Education, 24 01 2020. [Online]. Available: https://www.ibm.com/cloud/learn/data-centers. [Accessed 02 06 2020].

[8] 03 07 2020. [Online]. Available: https://en.wikipedia.org/wiki/Service_provider. [Accessed 01 08 2020].

[9] VMware, “Virtual Machine,” Vmware, [Online]. Available: https://www.vmware.com/topics/glossary/content/virtual-machine. [Accessed 05 05 2020].

[10] VMWare, “Virtual Machine,” VMWare, [Online]. Available: https://www.vmware.com/topics/glossary/content/virtual-machine. [Accessed 3 08 2020].

[11] aws.amazon, “What is cloud computing,” aws.amazon, [Online]. Available: https://aws.amazon.com/what-is-cloud-computing/. [Accessed 25 April 2020].

[12] T. O. Foundation, “docs.openstack,” The Openstack Foundation, 2012. [Online]. Available: www.openstack.org. [Accessed 18 April 2020].

[13] M. Solberg and B. Silverman, OpenStack for Architects, Packt Publishing Ltd, 2017.

[14] V. K. Selvaraj, “Field Sketch,” in *OpenStack Bootcamp Build and operate a private cloud environment effectively*, Birmingham, Packt Publishing, 2017, p. 46.

[15] T. O. Foundation, “OpenStack Compute (nova),” The Openstack Foundation, 11 20 2020. [Online]. Available: https://docs.openstack.org/nova/latest/. [Accessed 15 04 2020].

[16] The University of Cambridge, “general_concepts,” The University of Cambridge, 2018. [Online]. Available: https://docs.hpc.cam.ac.uk/cloud/background/00-welcome.html. [Accessed 07 April 2020].

[17] T. O. Foudation, “Introduction to the Block Storage service,” The Openstack Foudation, 16 08 2019. [Online]. Available: https://docs.openstack.org/mitaka/config-reference/block-storage/block-storage-overview.html. [Accessed 12 04 2020].

[18] K. Jackson, C. Bunch, E. Sigler, “Swift – OpenStack Object Storage,” in *OpenStack Cloud Computing Cookbook Third Edition*, Birmingham, Packt Publishing Ltd, August 2015, p. 163.

[19] T. O. Foundation, “Storage concepts,” The Openstack Foundation, 29 11 2018. [Online]. Available: https://docs.openstack.org/arch-design/design-storage/design-storage-concepts.html. [Accessed 13 04 2020].

[20] Selvaraj, V. Kumar, “Day 5 - Networking Strategy,” in *OpenStack Bootcamp*, Birmingham, Packt Publishing Ltd, November 2017, p. 87.

[21] K. Jackson, C. Bunch, E. Sigler, “Keystone – OpenStack Identity Service,” in *OpenStack Cloud Computing Cookbook Third Edition*, Birmingham, Packt Publishing Ltd, 2015, p. 1.

[22] RibbonCommunications, “ip-multimedia-subsystem-ims,” RibbonCommunications, [Online]. Available: https://ribboncommunications.com/company/get-help/glossary/ip-multimedia-subsystem-ims. [Accessed 4 08 2020].

[23] Metaswitch, “clearwater-ims-core,” metaswitch, [Online]. Available: https://www.metaswitch.com/products/core-network/clearwater-ims-core. [Accessed 20 07 2020].

[24] Clearwater, “Clearwater Architecture,” Clearwater, 2016. [Online]. Available: https://clearwater.readthedocs.io/en/stable/Clearwater_Architecture.html. [Accessed 15 07 2020].

[25] T. O. Foundation, “DevStack,” Openstack Foundation, 29 05 2020. [Online]. Available: https://docs.openstack.org/devstack/latest/. [Accessed 03 08 2020].

[26] T. O. Community, “Packstack,” The Openstack Foundation, [Online]. Available: https://wiki.openstack.org/wiki/Packstack. [Accessed 1 06 2020].

[27] T. O. Foundation, “Welcome to TripleO documentation,” The Openstack Foundation, 16 08 2019. [Online]. Available: https://docs.openstack.org/tripleo-docs/latest/. [Accessed 15 07 2020].

[28] Rdoproject, “Packstack: Create a proof of concept cloud,” rdoproject.org, [Online]. Available: https://www.rdoproject.org/install/packstack/. [Accessed 1 07 2020].

[29] T. O. foundation, “Log in to the dashboard,” Openstack foundation, 09 August 2019. [Online]. Available: https://docs.openstack.org/horizon/stein/user/log-in.html. [Accessed 15 May 2020].

[30] T. O. Foundation, “Open vSwitch: Self-service networks,” Openstack, 28 July 2020. [Online]. Available: https://docs.openstack.org/neutron/pike/admin/deploy-ovs-selfservice.html. [Accessed 28 July 2020].

[31] R. E. GHOMARI, “Cloud computing an integral part of the modern day it industry,” The Supinfo University, 23 06 2019 . [Online]. Available: https://www.supinfo.com/articles/single/9396-cloud-computing-an-integral-part-of-the-modern-day-it-industry. [Accessed 02 05 2020].

[32] Wikipedia, “Hypervisor,” Wikipedia, 15 03 2020. [Online]. Available: https://en.wikipedia.org/wiki/Hypervisor. [Accessed 05 03 2020].

[33] M. Cezar, “How to Install Your Own Cloud Platform with OpenStack in RHEL/CentOS 7,” TecMint.com, 25 08 2016. [Online]. Available: https://www.tecmint.com/openstack-installation-guide-rhel-centos/. [Accessed 16 03 2020].

[34] Selvaraj, V. Kumar, “Chapter 3: Day 3 - Field Sketch,” in *OpenStack Bootcamp*, Birmingham, Packt, 2017, p. 30.

[35] A. Shrivastwa, S. Sarat, “An Introduction to OpenStack,” in *Learning OpenStack*, Birmingham, Packt Publishing Ltd, 2015, p. 8.

[36] N. Khan, T. Herawan, Z. Inayat, “International Journal of Cloud Applications and Computing · July 2012,” 19 05 2014. [Online]. Available: https://www.researchgate.net/figure/Simple-architecture-of-cloud-computing_fig2_262201862. [Accessed 5 06 2020].

Abstract

Cloudification of IP Multimedia Subsystem IMS is the new migration of the telecom operators, thus to its standardized architecture that provides a variety of multimedia services such as telephony, video, text, and Internet, …etc., which provides the services over a unified IP network. Operators will have more flexibility and ease to deploy new IT services or existing services to bring new added value to their networks with better performance and at a lower cost.
To cloudify the IMS, we used the open-source Openstack solution as infrastructure as a Service using a server of the Algerie Telecom of Tlemcen. Then we used the open-source IMS product: Clearwater IMS, to test the functionality of our solution, we used several free SIP based software clients.

Keywords: IMS, Cloud Computing, Multimedia services, Clearwater, Openstack.

Résumé

Cloudification du IP multimédia subsystem IMS est la nouvelle migration des opérateurs télécoms, car il est basé sur son architecture standardisée qui fournit une variété de services multimédias tels que la téléphonie, la vidéo, le texte, Internet, …etc. de fournir ces services sur un seul réseau IP. Les opérateurs auront plus de flexibilités et de facilité de déployer de nouveaux services informatiques ou des services existants dans le but de fournir une nouvelle valeur ajoutée à leurs réseaux avec les meilleures performances et à moindre coût.
Afin de valider la cloudification IMS nous avons utilisé la solution open source Openstack en tant qu'infrastructure en tant que service (IaaS) en utilisant un serveur de la société Algérie Telecom de Tlemcen. Ensuite, nous avons utilisé la solution IMS open source : Clearwater IMS, pour tester le fonctionnement de notre solution, nous avons utilisé plusieurs logiciels gratuits de base sur SIP comme des clients d'IMS.

Mots clés : IMS, Cloud Computing, Services multimédias, Clearwater, Openstack.

ملخص

إن الانتقال IP multimédia subsystem IMS الى السحابيات يعد قفزة وتحدياً جديد لشركات الاتصالات، بناءًا على بنيته الموحدة التي تتيح مجموعة من خدمات الوسائط المتعددة مثل المكالمات الهاتفية والدردشة النصية ومكالمات الفيديو وعدة خدمات اخرى عبر شبكة IP واحدة، سيتمتع المشغلون بمزيد من المرونة والسهولة لنشر خدمات جديدة أو تطوير الخدمات الحالية من أجل تحقيق قيمة مضافة جديدة لشبكاتهم مع أداء أفضل وبتكلفة أقل.

من اجل تحقيق هذا التحول نحو السحابيات، استعملنا برنامج Openstack مفتوح المصدر كبنية تحتية كخدمة (IaaS) باستخدام خادم من شركة إتصالات الجزائر بتلمسان. ثم استخدمنا برنامج IMS مفتوح المصدر: Clearwater IMS، لاختبار عمل البرنامج، استخدمنا العديد من البرامج المجانية الأساسية على SIP كعملاء IMS.

الكلمات المفتاحية: السحابيات، الخدمات المتعددة الوسائط، Openstack, IMS, Clearwater IMS