

**Final Year Project to Obtain the Diploma of
Engineering**

Field:

Telecommunications

Speciality:

Telecommunication Systems and Networks

Subject:

**Study and Deployment of a Software Defined Networking
Solution on a Public Cloud (OpenNebula)**

Realized by:

LECHEHEB Manel

Members of The Jury:

Mrs BOUTARFA Souhila	Chair	M.C.B	ENST
Mrs ABBAD Leila	Supervisor	M.A.A	ENST
Mr SEFTA Mahfoud	Co-supervisor	Head of Cloud Infrastructure	Icosnet
Mr BELAHCENE Abdelkader	Examinator	M.A.A	ENST
Mrs CHIALI Imane	Examinator	M.C.B	ENST

Algiers, 06/07/2023

Dedecations

“

To my dearest mother, the one that always believes in me, supports me, guides me and the source of my happiness.

To my dear father, for his encouragement, support, and especially for the sacrifices he made.

To my life inspiration, my brother and sisters, who always encourage me to go beyond my dreams.

To the faculty and staff of National Higher School of Technology. Above all, to our God Almighty who showered us His blessings in our everyday lives, especially for the strength, courage, patience, wisdom,time, and guidance in the realization of this work.

”

Acknowledgments

“

First and foremost, I want to thank Allah for giving me strength, patience, desire, bravery, and health to complete this project.

I'd also want to thank my parents, my brother and my sisters for their support and encouragement during my long academic career.

I would like to express my gratitude and appreciation to my supervisor, Mrs. ABBAD Leila for assisting me through every difficult obstacle in my way, for supporting me and guiding me to finish my project.

I would also address my thanks to Mr. Sefta Mahfoud and all the infrastructure cloud's team who shared their expertise with me, guided me and answered my questions during the realisation of this project

And without forgetting the jury members for the honour they have given me by taking the time to read and evaluate this work.

Finally, thanks to everyone in my promotion team, the whole department of electrical engineering and industrial computing, and everyone who has helped me both morally and materially.

”

Contents

List of Figures	I
List of Tables	IV
List of Acronyms	V
General Introduction	1
I Generalities within Icosnet Cloud Architecture	4
I.1 Introduction	5
I.2 Virtualization	5
I.2.1 Definition	5
I.2.2 Why Virtualization ?	5
I.2.3 Type of Virtualization	6
I.2.4 Important Concepts in Virtualization	7
I.2.4.1 Virtual Machine Monitor (Hypervisor)	7
I.2.4.2 Host and Guest	7
I.2.4.3 Virtual Machine	8
I.2.4.4 Containers	8
I.2.4.5 Cluster	8
I.3 Cloud Computing	10
I.3.1 Definition	10
I.3.2 Types of Cloud Computing	11
I.3.3 Types of Cloud Services	11
I.3.4 Benefits of Cloud Computing	11
I.3.5 Relation between Virtualization and Cloud Computing	12
I.4 The Cloud Architecture of Icosnet	12
I.4.1 KVM Hypervisor	13
I.4.1.1 Installation	15
I.4.1.2 Networking Characteristics	17
I.4.2 OpenNebula Cloud	18
I.4.2.1 Installation	19
I.4.2.2 Networking Characteristics	21
I.5 Conclusion	22

II Needs Analysis and Choice of the SDN Solution	23
II.1 Introduction	24
II.2 Overview about SDN	24
II.2.1 Definition	24
II.2.2 OpenFlow	25
II.2.3 SDN on Cloud Computing	26
II.2.4 Why SDN ?	26
II.3 Requirements of the Solution	27
II.4 Choice of the Solution	27
II.4.1 OpenDaylight	28
II.4.2 OVS-DPDK	29
II.4.3 ONOS	31
II.5 Comparaision between the Solutions	32
II.6 Conclusion	34
III Implementation of ONOS SDN Controller	35
III.1 Introduction	36
III.2 The Architecture	36
III.3 The integration	37
III.3.1 SDN-Virtualization Part	37
III.3.1.1 Open Network Operating System (ONOS)	38
III.3.1.2 Open vSwitch	43
III.3.1.3 OVS-ONOS	48
III.3.2 SDN-Cloud Part	48
III.4 Conclusion	51
IV Results and Discussion	53
IV.1 Introduction	54
IV.2 Tests of the Solution	54
IV.2.1 Virtual Machines Isolation	54
IV.2.1.1 Reactive Forwarding	54
IV.2.1.2 Intent Framework	55
IV.2.1.3 Implementation	55
IV.2.2 Necessary Commands	58
IV.2.3 ONOS REST API	61
IV.3 Problems Experienced and Solutions	62
IV.3.1 Problems	62
IV.3.2 Solutions	63
IV.4 Propositions for a Real Implementation	64
IV.5 Conclusion	65
General Conclusion	66

Contents

Appendices	68
Bibliographies	75
Webographies	76
Abstract	

List of Figures

I.1	Comparison between Traditional and Virtual Computer Architecture [3]	5
I.2	Comparison between the Two Types of Hypervisors[5]	7
I.3	Concepts of Host and Guest within Hypervisor Based Virtualization [6]	8
I.4	Virtual Machine Architecture within Hardware Host	9
I.5	Comparison between VM and Container Architectures[8]	9
I.6	Migration of VMs from Overloaded Host to Host with Free Ressources on the Same Cluster[10]	10
I.7	Cloud Computing Service Providers	10
I.8	Icosnet Cloud Architecture	13
I.9	KVM Logo	13
I.10	Exemples of Libvirt API Commands	14
I.11	Virt-manager	15
I.12	Verify the VT and the KVM's Support	15
I.13	Adding Users and Verifying API Installation	16
I.14	Verification of libvirtd Status	16
I.15	Illustration of a Virtual Network Architecture Generated by KVM	17
I.16	Screenshot of (ip address) Command Displaying Network Interfaces Informations	17
I.17	KVM Networking Modes	18
I.18	OpenNebula Logo	18
I.19	Key Features Offered by OpenNebula[13]	19
I.20	OpenNebula Daemon Properly Started	19
I.21	OpenNebula Sunstone GUI	20
I.22	OpenNebula KVM Host Node	20
I.23	Opennebula Network[13]	21
I.24	Creating a new VM	21
II.1	Comparison between Traditional and SDN Architecture	24
II.2	Software defined Networking Architecture	25
II.3	SDN within Cloud Computing	26
II.4	OpenDaylight Logo	28
II.5	ODL Environment	28
II.6	ODL's Command Line Program	29
II.7	ODL's Problems	29
II.8	OVS Logo	30

II.9	Open vSwitch's Components	30
II.10	OVS-DPDK's Problem	31
II.11	ONOS Logo	32
III.1	The Suggested Architecture by Integrating SDN	36
III.2	Projection of the SDN Architecture on the Infrastructure	37
III.3	SDN-Virtualization Part	37
III.4	Adding SDN User	38
III.5	Java Version	38
III.6	Maven Version	39
III.7	ONOS Service Running	40
III.8	ONOS CLI	41
III.9	ONOS GUI	42
III.10	ONOS Activated Applications	43
III.11	OVS Service	44
III.12	Screenshot of "ip a" Command Displaying OVS Switch	44
III.13	OVS Configuration	44
III.14	Creating an OVS VNET	45
III.15	The Virtual Network Interface of the New VM Attached to OVS Virtual Network	47
III.16	Attachment the OVS Bridge to the ONOS Controller	48
III.17	SDN-Cloud Part	49
III.18	Opennebula Front-end Configuration with OVS	50
III.19	VNET Open vSwitch	50
III.20	Managing the Controlled Network via OpenNebula Sunstone	51
IV.1	Reactive Forwarding within ONOS	54
IV.2	The Types of ONOS Intents	55
IV.3	Displaying the Creation of the VMs Using the Command "ovs-vsctl show"	55
IV.4	Displaying the Two VMs on ONOS GUI	56
IV.5	Connection Establishment between the 2 VMs by activating the FWD	56
IV.6	Disconnection between the VMs because of the FWD Deactivation	57
IV.7	Intent's Flow	57
IV.8	VMs Connected through Intent	58
IV.9	Network View: Connectivity Requests Cause Flow [19]	58
IV.10	GUI ONOS Network Overview	59
IV.11	Cluster's Informations	59
IV.12	Routing and Security Configurations	60
IV.13	Managing Intents	60
IV.14	ONOS API	61
IV.15	ONOS Flows	62
IV.16	Issue of Incompatibility between the Versions	62
IV.17	Effect of the Nested Virtualization	63

IV.18	ONOS Cluster[18]	64
IV.19	Implementation of an OpenNebula Virtual Router	64
IV.20	Implementation of the Proxy[20]	65
A.1	VirtualBox's Snapshot	69
A.2	Creation of a Virtual machine Using virt-manager	70
A.3	Switching OS to Desktop	71
A.4	OS Desktop	71
A.5	MariaDB Back-end	72
A.6	OpenNebula Sunstone	73

List of Tables

II.1 Comparison between the SDN Solutions	33
---	----

List of Acronyms

API	<i>Application Programming Interface</i>
CLI	<i>Command Line Interface</i>
CPU	<i>Central Processing Unit</i>
CSP	<i>Cloud Service Provider</i>
FWD	<i>Forwarding</i>
GUI	<i>Graphical User Interface</i>
IaaS	<i>Infrastructure as a Service</i>
IBM	<i>International Business Machines Corporation</i>
IP	<i>Internet Protocol</i>
ISP	<i>Internet Service Provider</i>
IT	<i>Information Technology</i>
JSON	<i>JavaScript Object Notation</i>
KVM	<i>Kernel-based Virtual Machine</i>
LTS	<i>Long-Term Support</i>
MAC	<i>Media Access Control</i>
NAT	<i>Network Address Translation</i>
NETCONF	<i>Network management protocol</i>
NIC	<i>Network Interface Card</i>
ODL	<i>OpenDaylight</i>
OF	<i>OpenFlow</i>
ONF	<i>The Open Networking Forum</i>
ONOS	<i>Open Network Operating System</i>
OS	<i>Operating System</i>

List of Acronyms

OVS	<i>Open vSwitch</i>
PaaS	<i>Platform as a Service</i>
QEMU	<i>Quick Emulator</i>
QoS	<i>Quality Of Service</i>
RAM	<i>Random Access Memory</i>
REST	<i>Representational state transfer</i>
SaaS	<i>Software as a Service</i>
SAN	<i>Storage Area Network</i>
SDN	<i>Software Defined Networking</i>
SSH	<i>Secure Socket Shell</i>
SNMP	<i>Simple Network Management Protocol</i>
TCP	<i>Transmission Control Protocol</i>
VM	<i>Virtual Machine</i>
VMM	<i>Virtual Machine Monitor</i>
VMs	<i>Virtual Machines</i>
VNC	<i>Virtual Network Computing</i>
VNET	<i>Virtual Network</i>
VNIC	<i>Virtual Network Interface Card</i>
VT	<i>Virtualization Technology</i>
VXLAN	<i>Virtual eXtensible Local Area Network</i>
XML	<i>Extensible Markup Language</i>

General Introduction

During the few years, the world has completely changed, in view of the information technology (IT) landscape, according to the appearance of many emergence of revolutionary technologies such as virtualisation and cloud computing. These lasts are two interconnected state-of-the-art technology that has innovated the IT industry.

Virtualisation is a process that enables more powerful use of physical computer hardware. It allows the division of the computing resources (processors, memory, storage, etc.) of a single physical infrastructure into several virtual computers, commonly known as virtual machines, micro virtual machines or containers.

While cloud computing consists of hosting and using data on remote servers, that are managed by the cloud service providers (CSP), via the Internet. It offers access, orchestration and management of the shared computing resources using a software, rather than owning a local host device. Whereas the CSP sells such accessible services.

The combination of these two aspects has created a new beneficial business for the internet service provider (ISP) and especially for the CSP around the world. In Algeria, ICOSNET was one of them. It offers cloud services designed to suit client infrastructure's needs and size, through its data centers located in Algeria. The only private ISP in Algeria has a technical team responsible of the infrastructure cloud, which is constantly developing and keeping up to date with the latest technology in this field. That is why they suggest including software defined networking (SDN) on their own cloud.

The software defined networking is a cutting-edge technology that serves to centralize the management and the control of the network by separating the data forwarding and the network control, then using a software to administrate rather a virtual or a physical network. In our case, SDN is used to control a virtual network formed of a cluster running on three separated servers.

Context and Problematic

Considering that during this project, Icosnet uses a public cloud named OpenNebula, in the cloudy side. And in the second side, the Kernel-based Virtual Machine (KVM) hypervisor as a tool of virtualisation. But the conjunction of these two software creates

some limits. And the SDN controller is supposed to solve it. Among these limits:

- The decentralisation of the network configurations on the hardware, sometimes the same configurations repeat on many devices.
- Human contribution to network management increased the possibility of fatal errors. Especially, that Icosnet hosts data of many big companies.
- Absence of a graphical user interface (GUI) to clearly observe the architecture of the virtual network and all the changes on its configurations.
- The inability to isolate virtual instances on the same virtual network (VNET) from each other, and the obligation of creating a new VNET on OpenNebula for each instance in order to isolate it. Which is an exhausting and impractical action. Knowing that, the isolation of a virtual machine from the internal network is a major aspect of security.

Goals and Contributions

In the end of our dissertation, we are supposed to:

- Firstly, having a basic virtual infrastructure composed of virtual machines created using KVM hypervisor and attached to its bridge, through the public cloud OpenNebula Sunstone.
- Secondly, carry out the study of needs and the appropriate SDN solution for this architecture.
- Thirdly, the integration of the solution on the basic virtual infrastructure and applying the necessary modifications.
- Finally, testing the functioning of the SDN controller, applying the network configuration, especially isolation of the virtual machines (VMs), and suggesting requirement for the real implementation.

Dissertation Organization

It is structured into 4 chapters, such that:

- **Chapter I:** *"Generalities within Icosnet Cloud Architecture"*
Involved an overview about both virtualisation within the Kernel-based Virtual Machine and cloud computing within the public cloud OpenNebula. Also, review the current cloud infrastructure architecture used by Icosnet and its limitations.
- **Chapter II:** *"Needs Analysis and Choice of the SDN Solution"*
Discuss the concepts of software defined networking in the cloud environment, and study the suitable solution on such environment, according to the needs analysis.

- **Chapter III:** *"Implementation of ONOS SDN Controller"*
Detail the implementation and the configuration of whole open network operating system (ONOS) SDN solution and open vSwitch (OVS) between KVM and Opennebula.
- **Chapter IV:** *"Results and Discussion"*
Examine the capabilities of the new realizable architecture, its issues, and practical solutions by performing some tests.
- We conclude with a general summary of all the achieved work, valorisation of efforts and drawn attention to the difficulties faced. In the end, proposing some perspectives to ameliorate the solution.

Chapter I

Generalities within Icosnet Cloud Architecture

I.1 Introduction

In this chapter, we will briefly discuss the process of both virtualisation and cloud computing technologies, their general concepts and tools. Then, we will highlight the KVM hypervisor and the OpenNebula cloud, its installations, configurations and uses. Furthermore, we will review in detail Icosnet cloud architecture.

Drawing your attention to the fact that this work is a real cloud infrastructure project at Icosnet, in addition to being my end of studies project. I worked on it during my internship in coordination with the supervisors there, who provided for me the necessary tools to succeed in such an innovative project.

I.2 Virtualization

I.2.1 Definition

Virtualization was an inevitable result of the growing capability of datacenter technology and the continuing pressure to reduce technology costs[1]. To remove ambiguity, virtualization defined as a framework or methodology of dividing the resources of a computer hardware: Central Processing Unit (CPU), Random Access Memory(RAM), storage, Network Interface Card (NIC), etc, into multiple execution environments, by applying one or more concepts or technologies such as hardware and software partitioning, time-sharing, partial or complete machine simulation, emulation, quality of service, and many others. Furthermore, virtualization is a layer of abstraction that breaks the standard paradigm of computer architecture, decoupling the operating system from the physical hardware platform and the applications that run on it[2]. (Figure A.6)

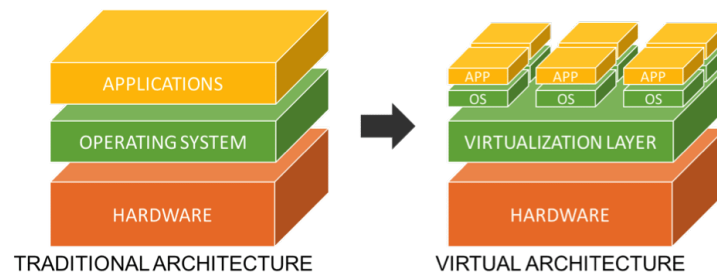


Figure I.1: Comparison between Traditional and Virtual Computer Architecture [3]

I.2.2 Why Virtualization ?

Virtualization provides notable benefits, here some of its key advantages:

- **Cost Savings:** By combining several physical servers into one virtual environment, virtualization can lower hardware expenses. This lessens the need for additional gear and the related expenses of electricity, cooling, and upkeep. Thing that makes virtualization an eco-friendly approach to IT.

- **Improved Efficiency:** By enabling the use of different operating systems and applications on a single physical server, virtualization can increase the effectiveness of IT operations. As a result, less new hardware and related expenses are needed.
- **Increased Availability:** By enabling redundancy and failover capabilities, virtualization can improve the availability of applications and services. This makes sure that even if one of the physical servers breaks down, the applications and services are still accessible.
- **Improved Security:** By isolating applications and services from one another, virtualization can enhance security. This lowers the chance that one application or service will negatively impact another.
- **Improved Mobility:** By enabling the migration of services and applications from one physical server to another and from a state to another, virtualization can enhance mobility. This makes it possible to relocate software and services to new data centers or locations and to an old state. Through a variety of functionalities, including snapshot (see Appendix A.1), copy, clone, and back up of the virtual system.

I.2.3 Type of Virtualization

Virtualization appears in a variety of kinds that come together to build the infrastructure that supports the virtual environment:

- **Server Virtualization:** This type of virtualization permits multiple operating systems and applications to run on the same hardware, by partitions a physical server into various virtual servers[4].
- **Desktop Virtualization:** Through this kind of virtualization, many users can access their own virtual desktops from any device, wherever they may be.
- **Application Virtualization:** This type of virtualization allows the running of applications on any hardware, irrespective of the underlying operating system (for example, running a Linux application in a Windows environment).
- **Storage Virtualization:** This kind of virtualization allows multiple physical storage devices to be managed as a single virtual entity.
- **Network Virtualization:** This type of virtualization permits the creation and the management of multiple virtual networks with its own addressing as a single entity on top of a physical network infrastructure. It covers virtual network interface card (VNIC), virtual switches and virtual routers, etc. Often, it uses SDN to control traffic.
- **Security Virtualization:** Multiple security policies may be applied to various virtual network segments using this kind of virtualization, and specially firewalling.

I.2.4 Important Concepts in Virtualization

I.2.4.1 Virtual Machine Monitor (Hypervisor)

The virtual machine monitor (VMM) is the major key tool for virtualization, it is a software that provides a layer of abstraction between the physical hardware and the virtual system, allowing the creation and the management of several virtual instances with isolated operating systems from each other, simultaneously and on a single physical machine. There exists two types of hypervisor:

a) Hypervisor type I

Also known by bare metal, it operates directly on the hardware, turning it into an instrument for controlling the operating system. Then, on the top of this hypervisor, the guest OSes operate. This kind has direct access to the hardware resources, which allows for efficient performance and resource management. For example : ESXi from VMware , KVM the free hypervisor for Linux .

b) Hypervisor type II

Also known by host metal, it works inside another operating system. For example: VirtualBox, Open Source software published by Oracle.

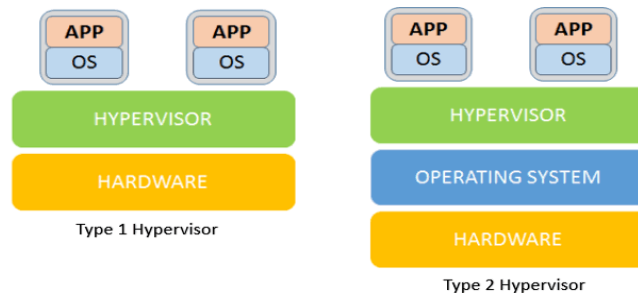


Figure I.2: Comparison between the Two Types of Hypervisors[5]

I.2.4.2 Host and Guest

The host is the physical machine host server; the underlying hardware that provides virtualized resources, such as CPU, RAM, storage and network I/O, and so on. While the guest is a completely separate and independent instance of an operating system and application software, that emulates the functionality of a physical computer system. The guest can take different forms, such as a virtual machine (VM) when its OS runs on the top of hosts infrastructures or a container when it shares the host's OS kernel. Guest can exist on a single physical machine but is usually distributed across multiple hosts for load balancing¹.

¹Distributing the computing workload within hosts

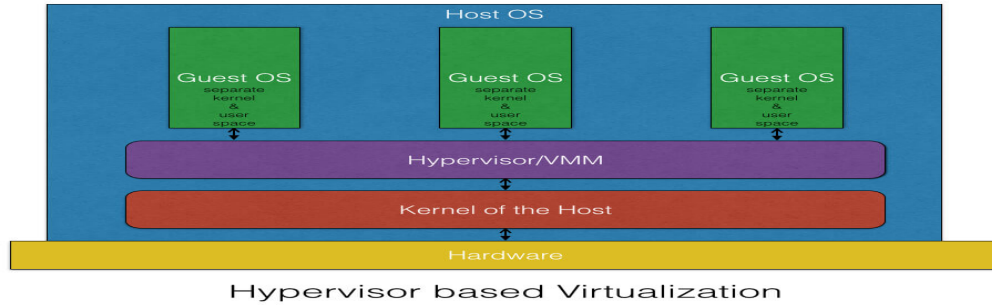


Figure I.3: Concepts of Host and Guest within Hypervisor Based Virtualization [6]

I.2.4.3 Virtual Machine

Virtual machine (VM) is a type of guests, i.e. it is an instance of an operating system created by a Virtual Machine Monitor (VMM) on the top of host infrastructure and enables a computer to behave like another computer. Every virtual machine contains a unique set of virtual hardware (CPU, memory, network interfaces, and disk storage) that is used to run an operating system and other programs. This enables customers to execute many applications and operating systems on the same machine, frequently with heterogeneous operating systems, without having to invest in additional physical hosts.

Its main properties within a datacenter consist on[7]:

- **Partitioning:** Distribute system resources among virtual machines while running different operating systems on a single physical host machine.
- **Insulation:** Isolate fault and security management at the hardware level.
- **Encapsulation:** Saving a virtual machine's whole state to files will enable easy moving and copying of the VM.
- **Hardware independence:** Any VM can be created or moved to any physical server.

I.2.4.4 Containers

Virtualization by containerization consists of partitioning directly at the level of the operating system. Each container so runs in its own environment while utilizing the same host OS. Because of this, containers are typically used to virtualize a program rather than a whole server².

I.2.4.5 Cluster

A cluster is a logical grouping of hosts (often three hosts: master, standby and worker³)

²During this thesis, we will focus especially on the hypervisor virtualization. So, we won't place much emphasis on virtualization by containerization

³Worker host is the responsible for balancing between master and standby host in case the master

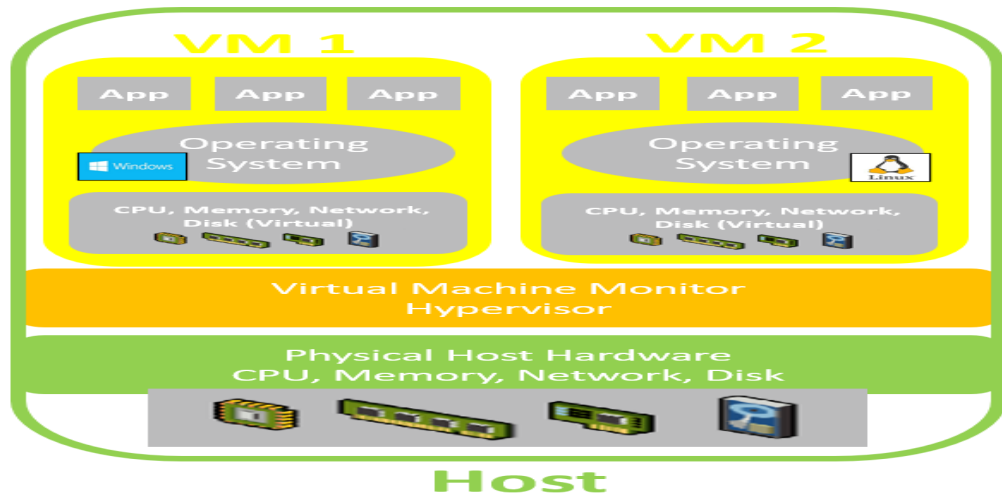


Figure I.4: Virtual Machine Architecture within Hardware Host

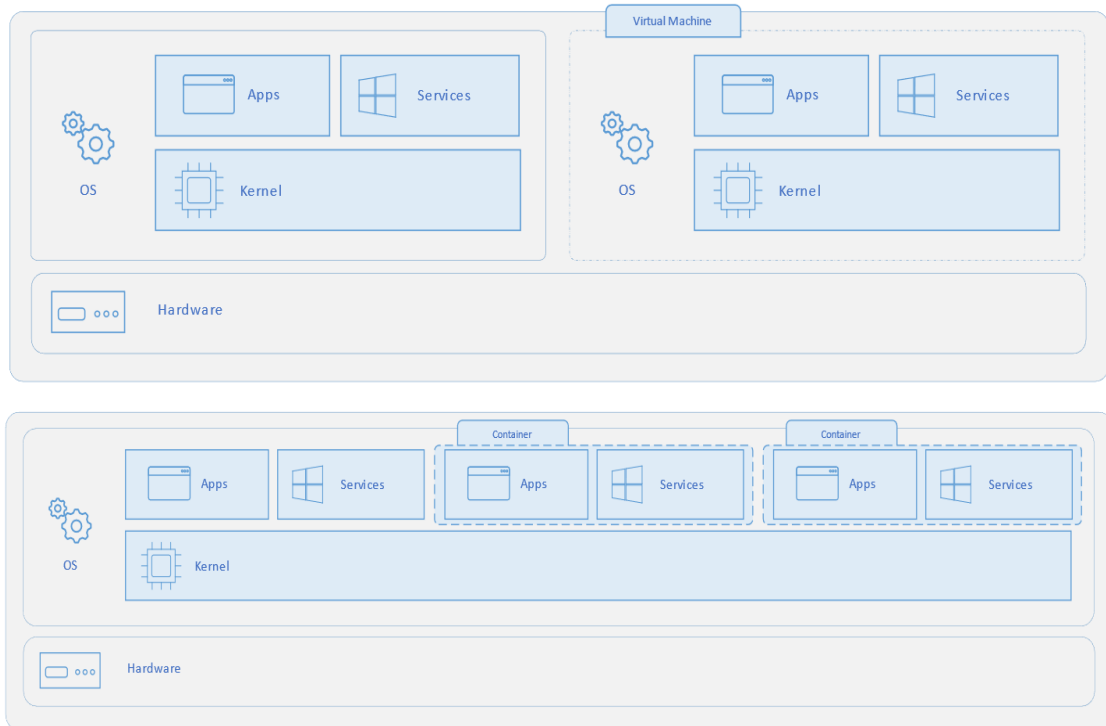


Figure I.5: Comparison between VM and Container Architectures[8]

with the same kind of CPU and shared a common storage domains. According to regulations set forth on the cluster and settings on the virtual machines, virtual machines are dynamically assigned to any host in a cluster and may be moved between them. The cluster is the greatest level at which power and load-sharing regulations can be established. It is an efficient method that safeguards against hardware and software malfunctions and guarantees high availability of servers and the network[9].

server fails or overloads.

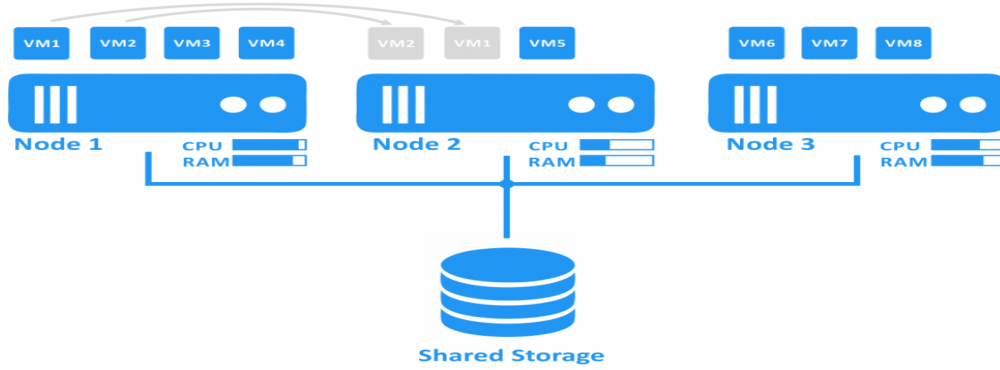


Figure I.6: Migration of VMs from Overloaded Host to Host with Free Ressources on the Same Cluster[10]

I.3 Cloud Computing

I.3.1 Definition

Cloud Computing consists of hosting and exploiting data on distant servers through the Internet. Rather than employing personal computers or local servers to run programs, cloud computing allows the use of shared computing resources. In most cases, it involves the utilization of virtualized resources, such as servers, storage, networks, software, and services, that are provided through the Internet ("the cloud"). Cloud computing eliminates the need for hardware and software by enabling users to access data and apps from any device with an Internet connection. It eliminates, also, the need for clients to configure or manage resources themselves, so they only pay the CSP for what they use.

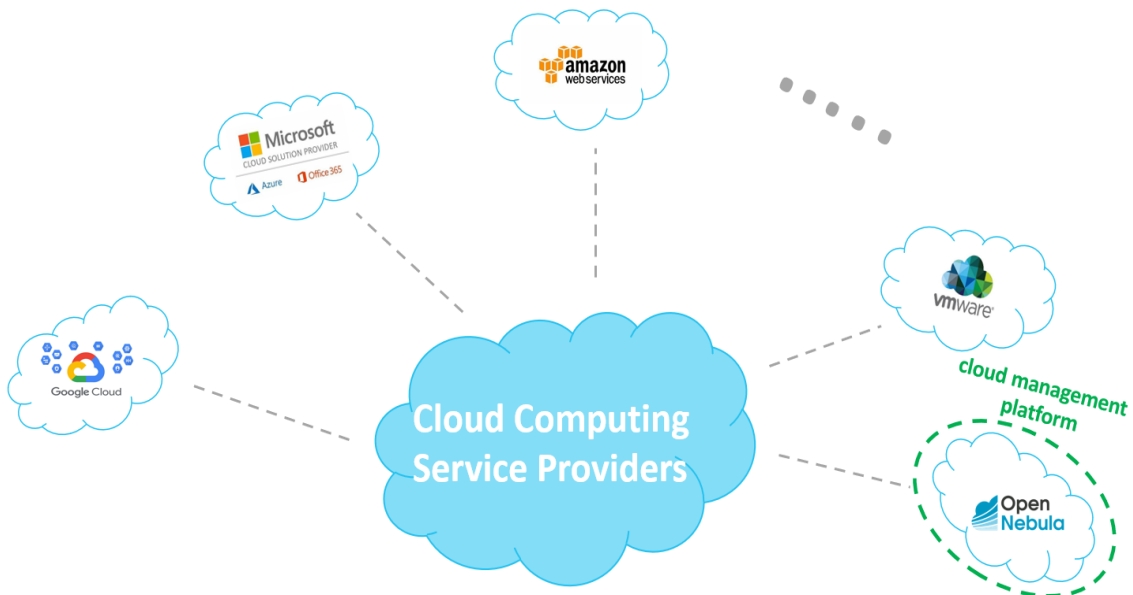


Figure I.7: Cloud Computing Service Providers⁴

⁴Google,MS, AMZ, VMware, OpenNebula are all Cloud Computing Service Providers each in his own way. But it should be noted that OpenNebula is a cloud management platform, i.e. a tool for cloud computing. And it hasn't its own data center.

I.3.2 Types of Cloud Computing

There are three principal types of cloud computing[11]:

- **Public Cloud:** Public clouds are shared infrastructure systems offered by a third-party cloud service provider and made available online to several customers or organizations
- **Private Cloud:** Private clouds offer additional control, customisation, and security because they are specialized infrastructure that is exclusively used by one company.
- **Hybrid Cloud** Hybrid Cloud is a combining between public and private cloud. It aims to handle the needs or limits of an organization while maximizing the advantages of both cloud deployment strategies.

I.3.3 Types of Cloud Services

There are three principal types of cloud services[11]:

- **Infrastructure as a service (IaaS):** It consists on offering infrastructure servers and virtual machines (VMs), storage, networks, operating systems from a cloud provider. while client should manage both OS and applications.
- **Platform as a service (PaaS):** It offers a platform and environment for clients to build, deploy, and manage applications without the complexity of infrastructure management. It is dedicated especially to developers and organizations looking to develop their own platform, through its flexibility.
- **Software as a service (SaaS):** It provides software applications over the internet. While the provider is responsible on providing the whole application, and take care of any upkeep, such as software updates and security patches.

I.3.4 Benefits of Cloud Computing

Here are only the tip of cloud computing's iceberg of potential:

- Cost Efficiency
- High flexibility
- Availability
- Environment friendly
- Easy backup and restore
- High performance
- Excellent accessibility

- More Secure

I.3.5 Relation between Virtualization and Cloud Computing

Virtualization is the key component of an improved cloud computing. Despite the fact that both two technologies could exist one far from the other, but the combination of them is the crucial key for CSP to enhance the services offered on their data centers. Admitting that virtualization is the physical foundation layer of services, while cloud computing offers access, orchestration and management of these services.

I.4 The Cloud Architecture of Icosnet

In keeping up to date with modern technologies, and whitening the creation and exploitation of its data center, Icosnet follows the approach of applying a combination between hypervisor virtualization and cloud computing, in order to offer a high quality of services as a CSP to its clients. So on the one hand, virtualisation is ensured by Kernel-based Virtual Machine (KVM) hypervisor. On the other hand, cloud computing is guaranteed by the use of the public cloud OpenNebula.

Technically, the data center of Icosnet uses a cluster of three physical servers linked to a storage bay (SAN, Storage Area Network)⁵ and managed by OpenNebula Sunstone. So as to create a KVM cluster managed by openNebula, KVM hypervisor is properly installed on each host server with an openNebula KVM node driver. And, in another separated virtual machine created on another platform, the OpenNebula front end is deployed. Also, an ssh link is established between the OpenNebula public cloud and the KVM cluster, to enable passwordless login on the system. For the purpose of that, all nodes and front-end can interact to one another through SSH without any manual intervention.

Then, all the servers are connected to a physical router that offers routing and NAT capabilities. Followed by a physical firewall to filter the inbound and outbound traffic. Knowing that, they don't need to use a physical switch because of the presence of a virtual switch on each server. This virtual bridge is automatically created while installing and configuring KVM.

To create a service or a virtual machine, the network administrator uses OpenNebula Sunstone to send orders to the KVM hypervisor, which create the instance on a specified host server. Through the cloud, a necessary configuration must be done, including assigning private internet protocol address (IP address) and VNET⁶, computing resources and determine an OS template, according to the clients requirements.

⁵Generally, servers are used to store its operating systems, necessary applications such KVM and temporary data. While SAN store virtual machines files and data that needs to be shared by multiple servers (snapshots, clone, and backup)

⁶Virtual Network

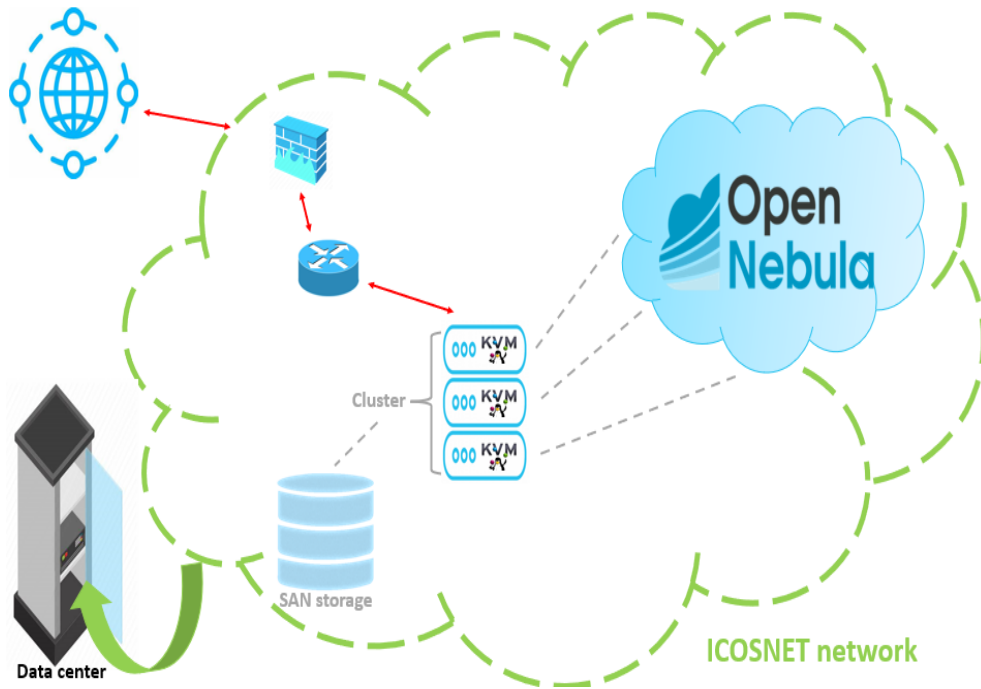


Figure I.8: Icosnet Cloud Architecture

In order to conduct a needs study, we try to approach this architecture by creating a test environment using two client-server virtual machines. One machine is considered as a host server including the KVM hypervisor, and the second machine is considered as the machine which contains the public cloud OpenNebula. Then an SSH linked is established between the two machines as required. Both of them use an Ubuntu 22.04 Long-Term Support (LTS) server OS, which is the latest LTS version of Ubuntu, because of its stability, long-term support, flexibility and its large integration with the cloud. Knowing that, Icosnet gives VM access to two VMs, I do the disk extension and the switching from server OS to desktop OS before using them. (For more information about the two process, please refer to Appendices A.3)

I.4.1 KVM Hypervisor

KVM is the acronym of "Kernel-based Virtual Machine" which is the Linux open-source virtualization module. It is considered as an hypervisor type I, because it enables the Linux machine to function as a host running a number of separated VMs known as guests. It is merged into the mainline Linux kernel since Linux2.6.20 version. Each VM is built as a typical Linux process, scheduled by the default Linux scheduler, and equipped with specific virtual hardware such a network card, graphics adapter, CPU(s), memory, and disks[12].

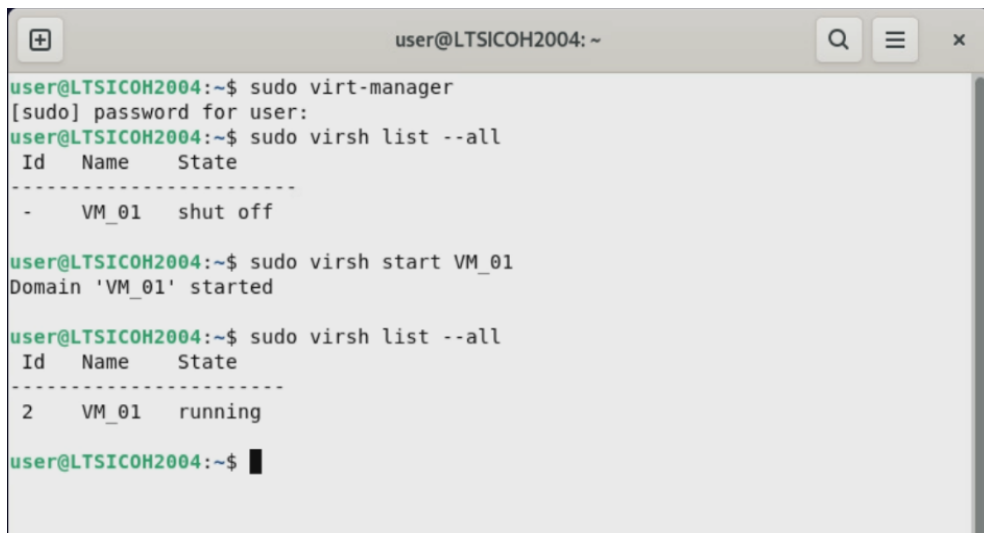


Figure I.9: KVM Logo

KVM offers many features including:

- **Full virtualization:** The full virtualization offered by KVM enables the operation of numerous virtual machines on a single physical server.
- **High performance:** KVM offers near-native performance, enabling high-performance virtual machines.
- **Scalability:** A single physical server can accommodate up to thousands of virtual computers using KVM.
- **Security:** Each virtual machine is protected from the others due to the robust isolation that KVM provides.
- **Open source:** KVM can be customized and integrated with other open source projects because it is open source.
- **Support for multiple operating systems:** Many operating systems such as Linux, Windows, and Mac OS X are supported by KVM.
- **Easy to use:** By integrating a simple graphical user interface and command line tools, KVM is simple to operate.

KVM needs two special package: the emulator QEMU and th API Libvirt. Qemu⁷ is an hypervisor type I used with KVM through "kvm-qemu" to emulate various hardware components, such as CPUs and I/O devices. While Libvirt API is used to unify the KVM management, it provides CLI utilities like virsh, virt-manager, and virt-install commands, which provides hundreds of options to manage every aspect of the virtual machines communicate with the virtualization infrastructure.



```
user@LTSICOH2004: ~
user@LTSICOH2004:~$ sudo virt-manager
[sudo] password for user:
user@LTSICOH2004:~$ sudo virsh list --all
 Id   Name   State
-----
 -    VM_01  shut off

user@LTSICOH2004:~$ sudo virsh start VM_01
Domain 'VM_01' started

user@LTSICOH2004:~$ sudo virsh list --all
 Id   Name   State
-----
 2    VM_01  running

user@LTSICOH2004:~$ █
```

Figure I.10: Exemples of Libvirt API Commands

A GUI utility called virt-manager is used to manage virtual machines using KVM:

⁷QEMU is an hypervisor but can be used as an emulator with KVM to improve virtualization capabilities, and because KVM is a bare-metal VMM that provides hardware virtualization capabilities and doesn't offer a software to manage it

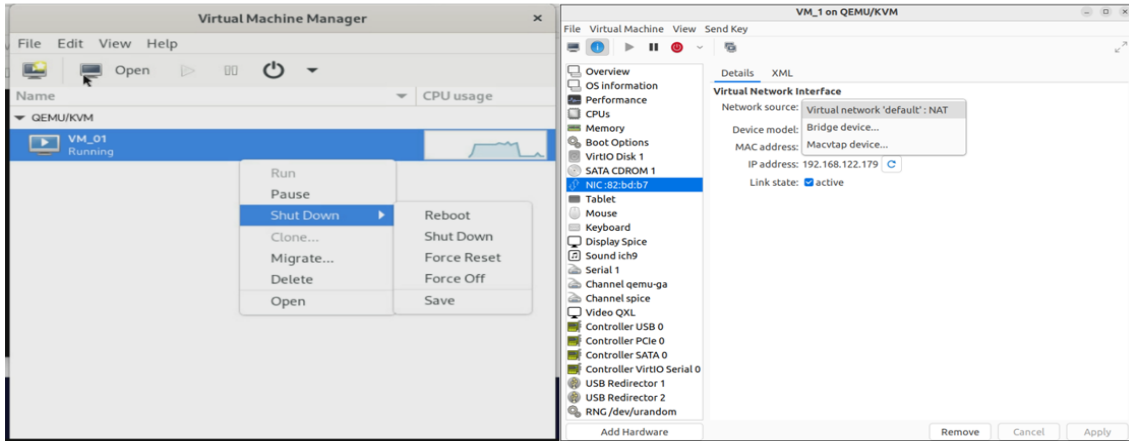


Figure I.11: Virt-manager

I.4.1.1 Installation

To install KVM on Ubuntu 22.04 LTS, we need to follow these steps:

1. Check if the CPU supports Virtualization technology(VT), using:

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

If we get number $\neq 0$, the CPU supports VT. Otherwise, it does not support VT.

2. Check the host system's CPU specifications and capabilities, using:

```
$ apt install cpu-checker
```

3. Check if the system supports KVM, using:

```
$ kvm-ok
```

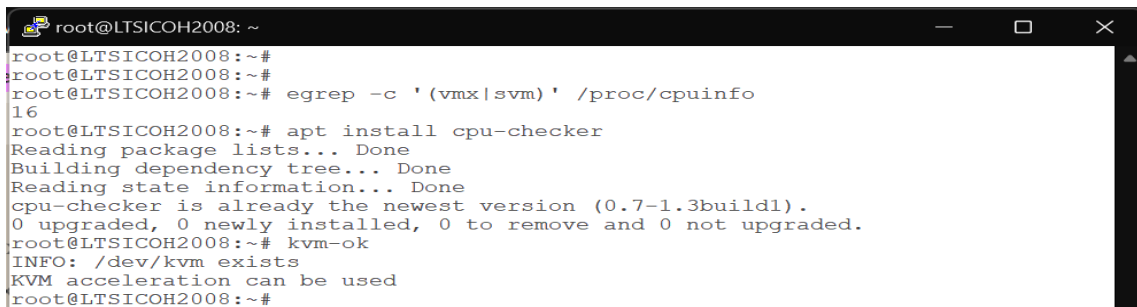


Figure I.12: Verify the VT and the KVM's Support

4. Download the necessary packages, using:

```
$ sudo apt install qemu-kvm libvirt-daemon-system
```

```
$ sudo apt install libvirt-clients bridge-utils
```

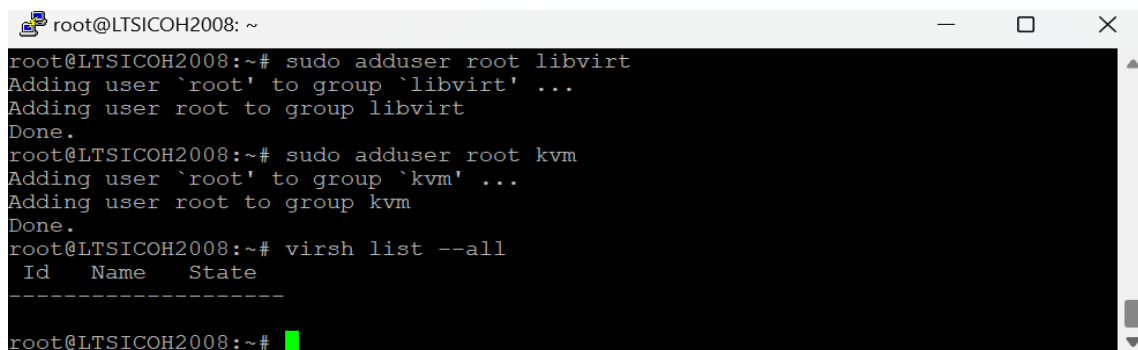
5. Add user to both Libvirt and KVM groups, using:

```
$ sudo adduser root libvirt
```

```
$ sudo adduser root kvm
```

6. Verify the installation of libvirt API, using:

```
$ sudo virsh list-all
```

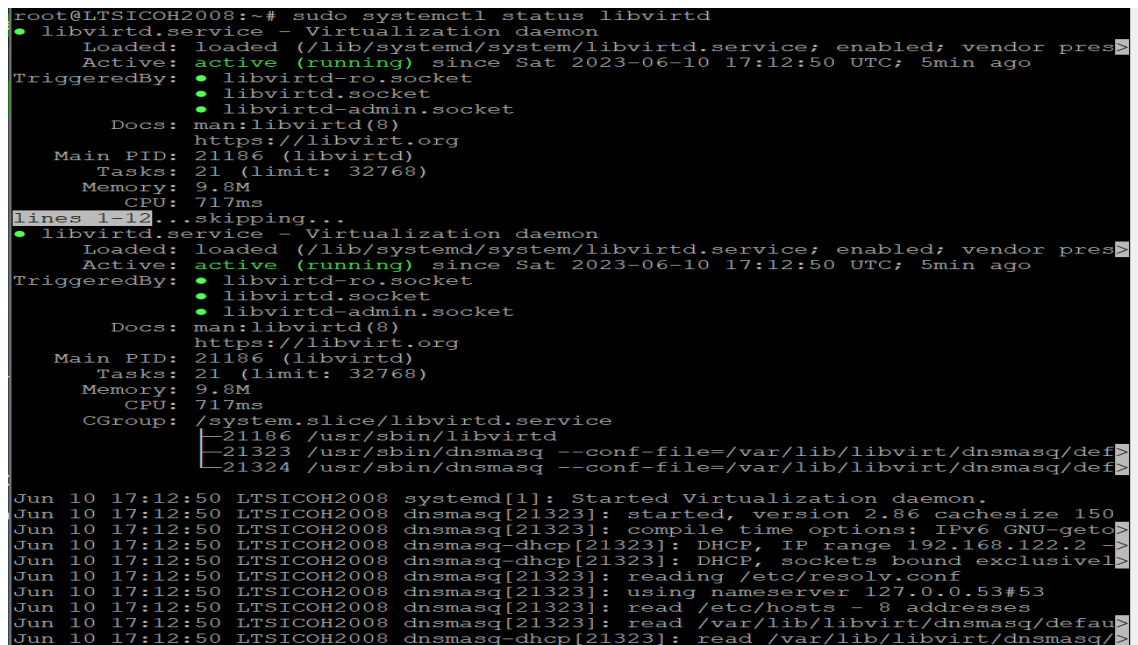


```
root@LTSICOH2008: ~  
root@LTSICOH2008:~# sudo adduser root libvirt  
Adding user `root' to group `libvirt' ...  
Adding user root to group libvirt  
Done.  
root@LTSICOH2008:~# sudo adduser root kvm  
Adding user `root' to group `kvm' ...  
Adding user root to group kvm  
Done.  
root@LTSICOH2008:~# virsh list --all  
Id Name State  
-----  
root@LTSICOH2008:~#
```

Figure I.13: Adding Users and Verifying API Installation

7. Check the status of the libvirtd⁸ service, using:

```
$ sudo systemctl status libvirtd
```



```
root@LTSICOH2008:~# sudo systemctl status libvirtd  
● libvirtd.service - Virtualization daemon  
   Loaded: loaded (/lib/systemd/system/libvirtd.service; enabled; vendor pres  
   Active: active (running) since Sat 2023-06-10 17:12:50 UTC; 5min ago  
   TriggeredBy: ● libvirtd-ro.socket  
                 ● libvirtd.socket  
                 ● libvirtd-admin.socket  
   Docs: man:libvirtd(8)  
          https://libvirt.org  
   Main PID: 21186 (libvirtd)  
   Tasks: 21 (limit: 32768)  
   Memory: 9.8M  
   CPU: 717ms  
   lines 1-12...skipping...  
● libvirtd.service - Virtualization daemon  
   Loaded: loaded (/lib/systemd/system/libvirtd.service; enabled; vendor pres  
   Active: active (running) since Sat 2023-06-10 17:12:50 UTC; 5min ago  
   TriggeredBy: ● libvirtd-ro.socket  
                 ● libvirtd.socket  
                 ● libvirtd-admin.socket  
   Docs: man:libvirtd(8)  
          https://libvirt.org  
   Main PID: 21186 (libvirtd)  
   Tasks: 21 (limit: 32768)  
   Memory: 9.8M  
   CPU: 717ms  
   CGroup: /system.slice/libvirtd.service  
           └─21186 /usr/sbin/libvirtd  
             └─21323 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/def  
               └─21324 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/def  
Jun 10 17:12:50 LTSICOH2008 systemd[1]: Started Virtualization daemon.  
Jun 10 17:12:50 LTSICOH2008 dnsmasq[21323]: started, version 2.86 cachesize 150  
Jun 10 17:12:50 LTSICOH2008 dnsmasq[21323]: compile time options: IPv6 GNU-geto  
Jun 10 17:12:50 LTSICOH2008 dnsmasq-dhcp[21323]: DHCP, IP range 192.168.122.2 -  
Jun 10 17:12:50 LTSICOH2008 dnsmasq-dhcp[21323]: DHCP, sockets bound exclusivel  
Jun 10 17:12:50 LTSICOH2008 dnsmasq[21323]: reading /etc/resolv.conf  
Jun 10 17:12:50 LTSICOH2008 dnsmasq[21323]: using nameserver 127.0.0.53#53  
Jun 10 17:12:50 LTSICOH2008 dnsmasq[21323]: read /etc/hosts - 8 addresses  
Jun 10 17:12:50 LTSICOH2008 dnsmasq[21323]: read /var/lib/libvirt/dnsmasq/defau  
Jun 10 17:12:50 LTSICOH2008 dnsmasq-dhcp[21323]: read /var/lib/libvirt/dnsmasq/
```

Figure I.14: Verification of libvirtd Status

⁸The "libvirtd" service is a daemon that manages the communication between virtualization solutions KVM and the libvirt API

I.4.1.2 Networking Characteristics

When libvirt and KVM are properly installed, a new virtual device is added simultaneously which is a virtual bridge. And while creating an new VM (To delve deeper into the process of creating a VM, please refer to Appendix A.2) , a new VNET which corresponds to the virtual network interface card (VNIC) with a new IP address according to the networking mode used are added .

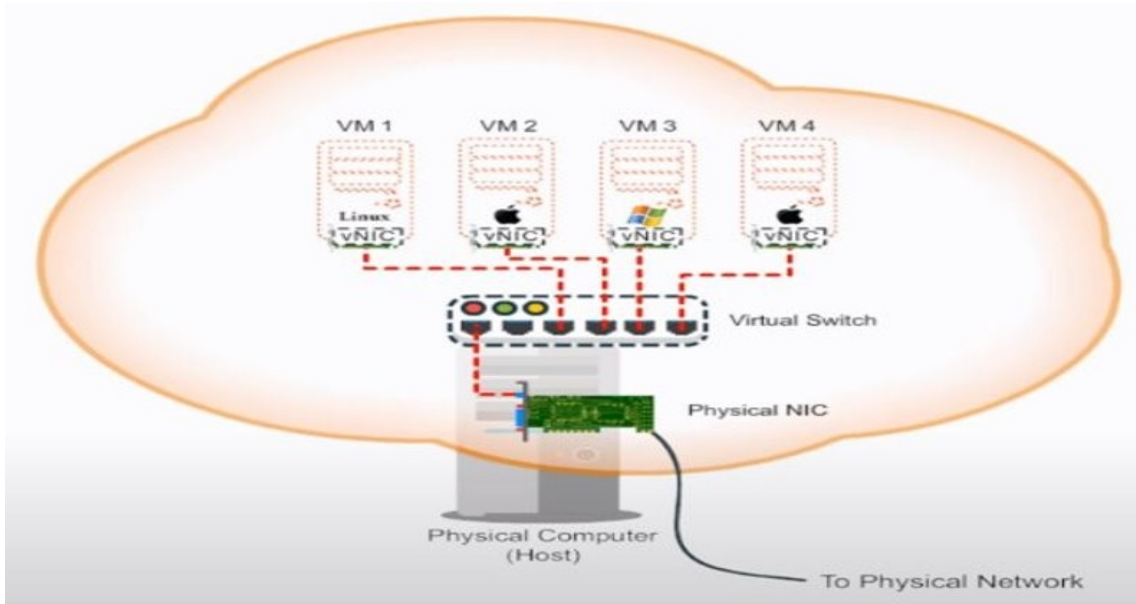


Figure I.15: Illustration of a Virtual Network Architecture Generated by KVM

```

3: virbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP gro
up default qlen 1000
   link/ether 52:54:00:de:6b:30 brd ff:ff:ff:ff:ff:ff
   inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
      valid_lft forever preferred_lft forever
7: vnet3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master virbr0
state UNKNOWN group default qlen 1000
   link/ether fe:54:00:97:55:e7 brd ff:ff:ff:ff:ff:ff
   inet6 fe80::fc54:ff:fe97:55e7/64 scope link
      valid_lft forever preferred_lft forever
8: vnet4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master virbr0
state UNKNOWN group default qlen 1000
   link/ether fe:54:00:82:bd:b7 brd ff:ff:ff:ff:ff:ff
   inet6 fe80::fc54:ff:fe82:bdb7/64 scope link
      valid_lft forever preferred_lft forever
manel@manel-Inspiron-14-5425:~$
    
```

Interfaces of the 2 virtual machines

Virtual Bridge (Switch)

Figure I.16: Screenshot of (**ip address**) Command Displaying Network Interfaces Informations

There exist three principal networking modes for communication between the guests and the KVM host:

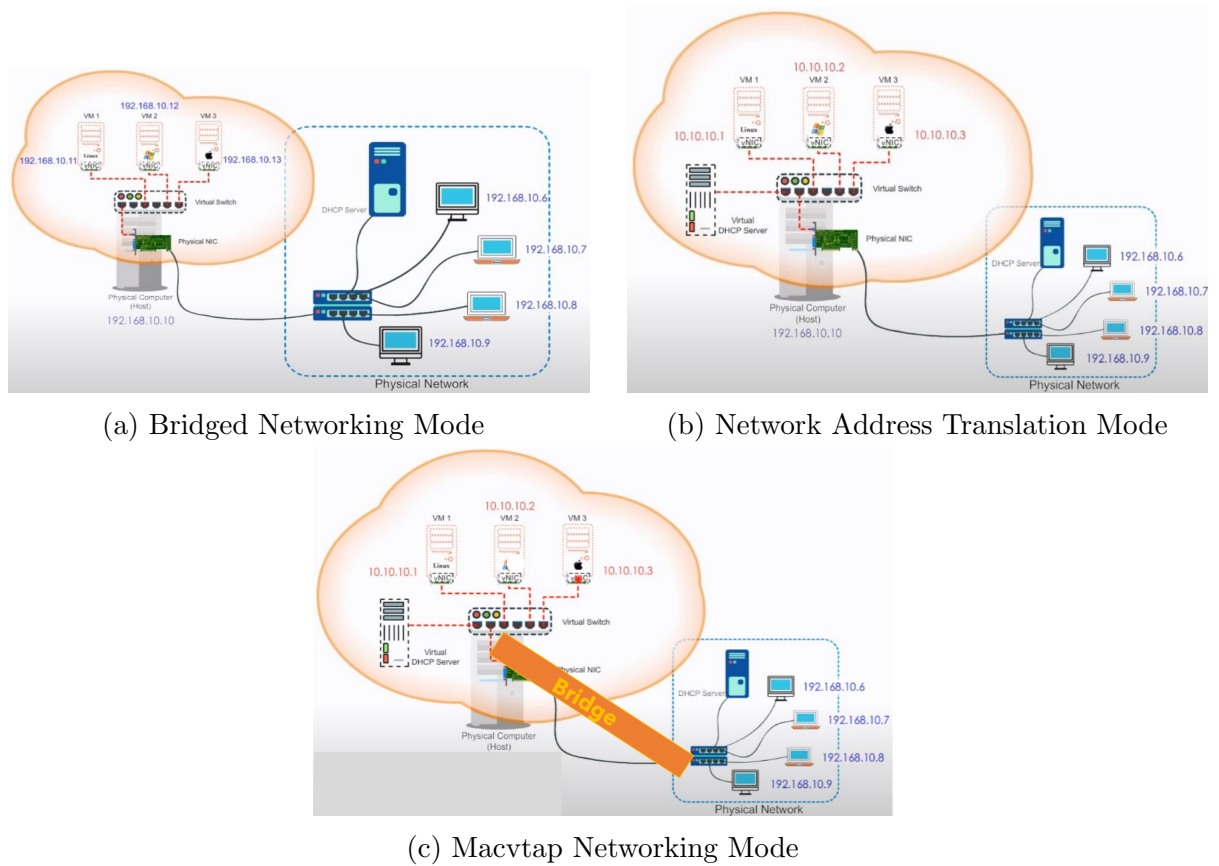


Figure I.17: KVM Networking Modes

I.4.2 OpenNebula Cloud

OpenNebula is an open source cloud and edge computing platform. It is a cloud management platform for building and managing private, public and hybrid clouds. It combines hypervisor virtualization and container technologies with multi-tenancy, auto-provision and elasticity to offer applications and services on-demand[13]. It provides a centralized interface for deploying, monitoring and managing virtualized computing resources, including virtual machines, storage, and networking[14].

OpenNebula provides a single, feature-rich and flexible platform that unifies management of IT infrastructure and applications, preventing vendor lock-in and reducing complexity, resource consumption and operational costs. It uses a modular architecture, allowing users to add custom functionality through plugins and APIs. It supports various hypervisors, such as KVM[13].

Throughout our project, we use the stable version 6.6 of OpenNebula as a public cloud with the KVM hypervisor. OpenNebula uses a module called "vmm_exec" to



Figure I.18: OpenNebula Logo

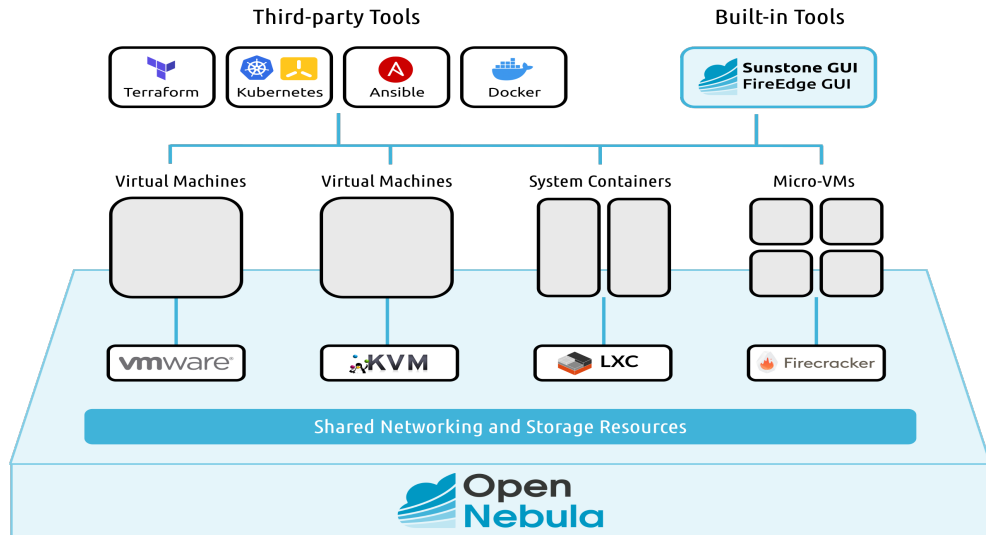


Figure I.19: Key Features Offered by OpenNebula[13]

communicate with KVM using the libvirt communication protocol. Exactly, it uses the VMM(Verification Methodology Manual) protocol. It resembles a driver. The vmm_exec sends commands to KVM to start, stop, monitor and manage virtual machines.

I.4.2.1 Installation

The installation of OpenNebula consists on:

- **Cloud single front end installation:**

Involving MySQL/MariaDB database installation, OpenNebula community edition software downloading, OpenNebula Daemon configuration, managing the oneadmin user, starting services and opening firewall's ports on the first machine.(To delve deeper on the complete installation, please refer to Annexe A.4)

To verify installtion, we run command: `$ oneuser show`

```

root@LTSICOH2004: ~
root@LTSICOH2004:~# oneuser show
USER_0 INFORMATION
ID : 0
NAME : oneadmin
GROUP : oneadmin
PASSWORD : a560f64750a7b82e630e56e560744e20f23b7adcbe4d4
4b186bb1189cf4e04cd
AUTH_DRIVER : core
ENABLED : Yes

TOKENS

USER TEMPLATE
TOKEN_PASSWORD="5a67217563bb30eab93ec1e3e17d9da564534c93ed336bf
d2d72da6c4551c1a6"

VMS USAGE & QUOTAS

VMS USAGE & QUOTAS - RUNNING

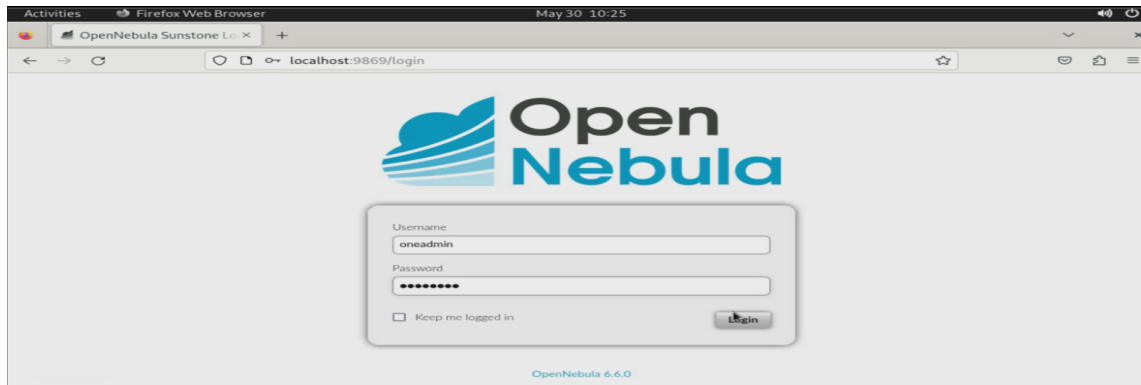
DATASTORE USAGE & QUOTAS

NETWORK USAGE & QUOTAS

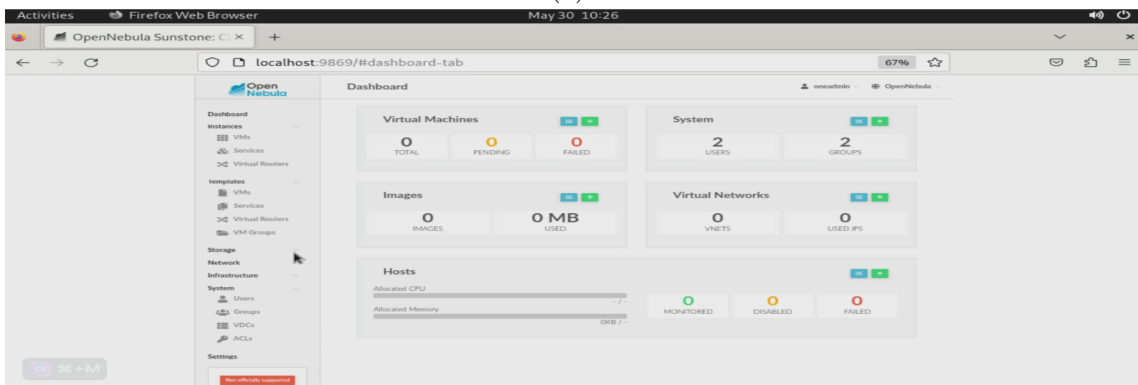
IMAGE USAGE & QUOTAS
    
```

Figure I.20: OpenNebula Daemon Properly Started

We login in through the Sunstone GUI via the link : http://<frontend_address>:9869



(a)



(b)

Figure I.21: OpenNebula Sunstone GUI

- KVM node installation:** Involving OpenNebula software and KVM Node Package installation on the second machine, Passwordless SSH configuration from Front-end to hypervisor Node an Adding the Host with Sunstone.(To delve deeper on the complete installation, please refer to Annexe A.4)

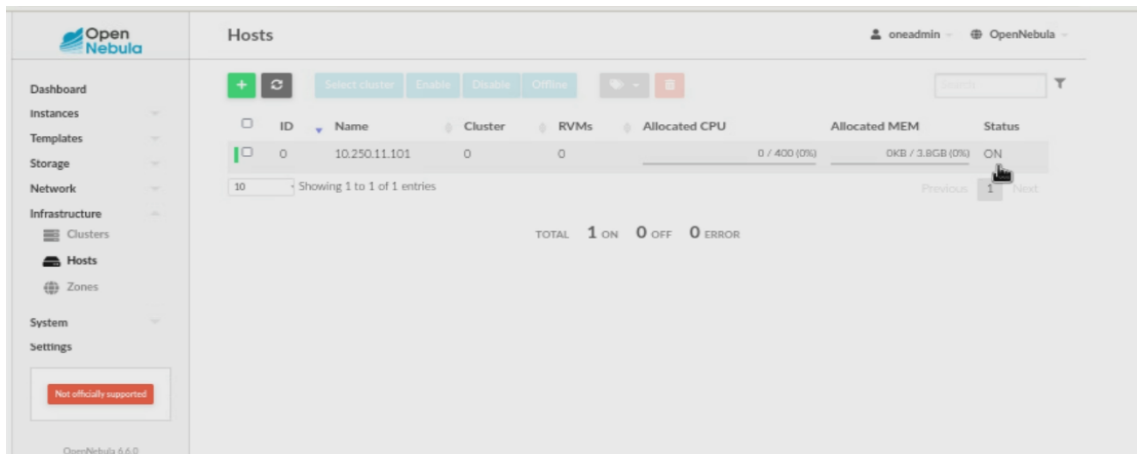


Figure I.22: OpenNebula KVM Host Node

I.4.2.2 Networking Characteristics

To manage and monitor the host, and to transfer the Image files. For our tests, We use the simple virtual network mode which is bridged mode.

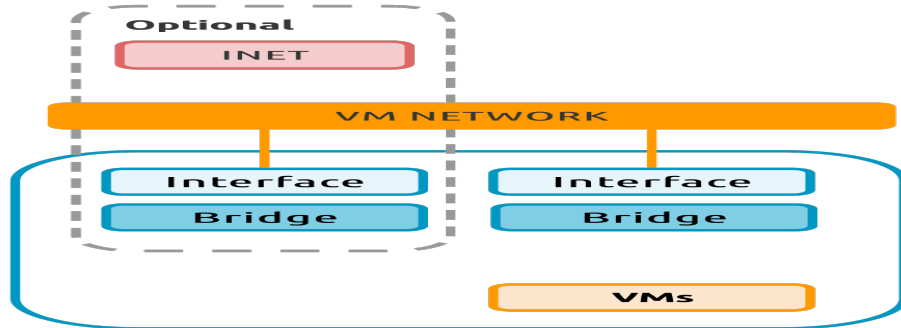
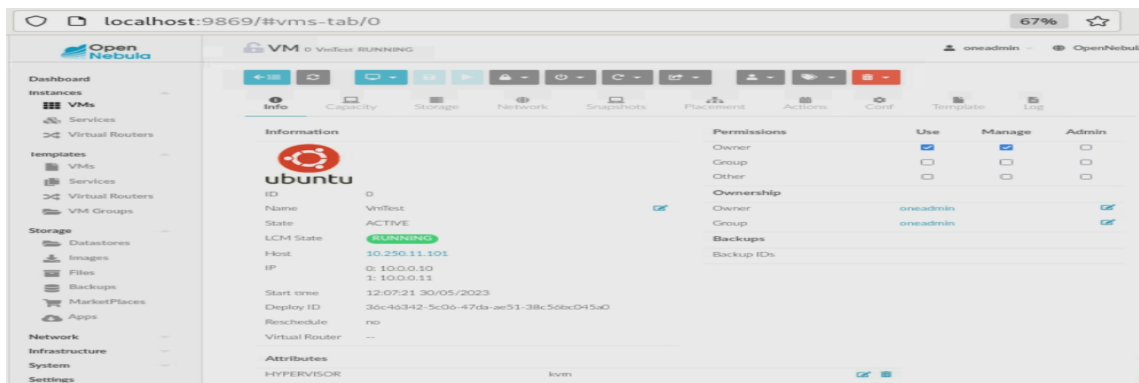
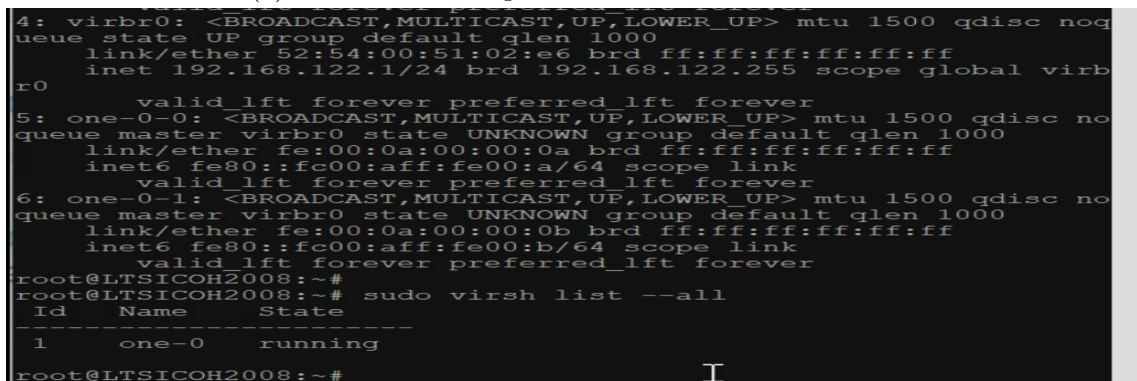


Figure I.23: Opennebula Network[13]

So after creating a VNET using bridged mode, we create template for the VM, then the VM instance itself:



(a) Screenshot on OpenNebula Sunstone Front end



(b) Screenshot on KVM Node

Figure I.24: Creating a new VM

I.5 Conclusion

In conclusion, this chapter delves into various aspects of virtualization and cloud computing, with a specific emphasis on the KVM hypervisor and the OpenNebula public cloud. The chapter explores the implementation details, networking specifications, and the application of these technologies through the cloud architecture of Icosnet. By successfully creating a prototype of the basic architecture of Icosnet, we have laid the foundation for delivering a comprehensive needs analysis and selecting the appropriate SDN solution, in the next chapter. This enables us to optimize the network infrastructure and enhance overall performance.

Chapter II

Needs Analysis and Choice of the SDN Solution

II.1 Introduction

Through this chapter, we aim to provide a comprehensive understanding of the concept of software defined networking (SDN) in general, with a specific focus on its application in cloud computing. We will conduct a needs analysis and conclude by selecting the appropriate solution for Icosnet’s cloud architecture.

II.2 Overview about SDN

Switches on physical networks route information in the form of packets based on the combined knowledge of different elements. Routing tables are built up by exchanging topology and state information across different network devices. Network problems cause temporary disruption as devices try to discover new paths, which can lead to data loss and routing delays. In light of the fact that every new technology is developed to address a specific need or solve an issue. The SDN technology emerges to solve specially such a problem and others.

II.2.1 Definition

The Open Networking Forum (ONF)¹ defines SDN as follows:

“Software-Defined Networking (SDN) is an emerging architecture that is dynamic,manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today’s applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. The OpenFlow protocol is a foundational element for building SDN solutions[15].”

Simply put, SDN is a networking architecture that permits the centralized management and control of a network into a single entity ”controller” that is driven by application programming interfaces (APIs). As a result, it separates the control plane, which determines how the network should behave, from the data plane, which implements those decisions. Using several protocols and technologies.

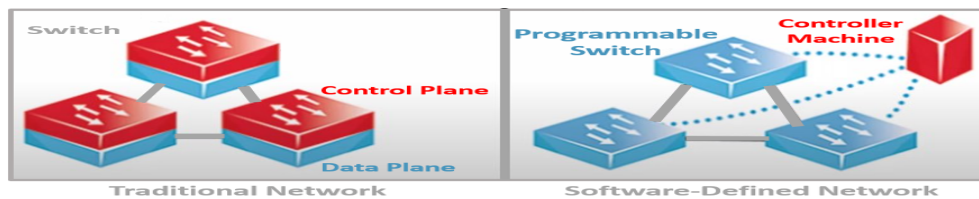


Figure II.1: Comparison between Traditional and SDN Architecture

¹A working groups that aim to accelerate the use of SDN and OpenFlow technologies through cooperation, standardization activities, and the creation of open-source solutions

In general, software defined networking functionalities can be categorized into three planes. These are as follows:

- **Data Plane:** It involves the various network devices (such as switches, access points, routers, and firewalls), which send and receive information to and from the controller across a southbound APIs.
- **Control Plane:** It includes the controller, which is the central component of an SDN architecture and which allows for centralized management and control, automation, and policy enforcement across the network environments.
- **Application Plane:** The controller and the applications and policy engines communicate with one another via northbound APIs, making an SDN appear to be a single logical network device.

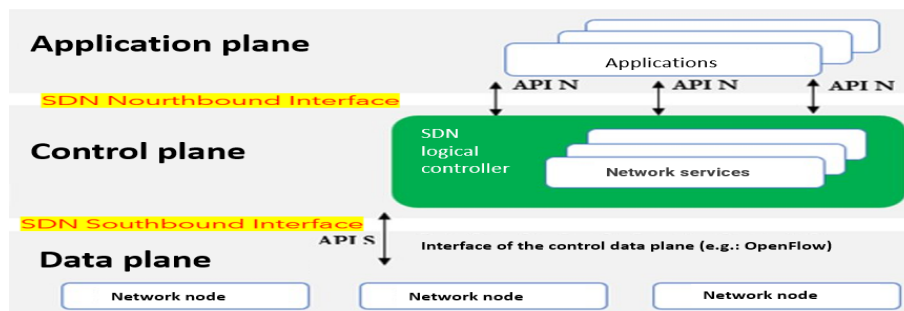


Figure II.2: Software defined Networking Architecture

II.2.2 OpenFlow

OpenFlow protocol (OF) is the language used by the central controller to communicate changes to network equipment, switches². It works on the transmission control protocol (TCP). The standard protocol is TCP 6633 for OF V1.0 and 6653 for OF V1.3+, while the latest version used in the industry is V1.5. whereas, OF channel between the switch and the controller is successfully established only after a successful TCP 3-way handshake.

Furthermore, the essential part of the OF protocol is the flow table which is similar to the media access control (MAC) table of a traditional switch. It saves flow coming from the controller that instructs the SDN switch what to do with a packet when it arrives at an incoming port. The switch will compare certain factors, such as an IP address, port number, MAC address, VLAN ID, etc., and choose the best matching flow entry from the table, carrying out the action linked to that entry. The corresponding action is carried out if a match is discovered. If a match cannot be made “TABLE_MISS”, the switch forwards the packet to the SDN controller, which will then decide on the best course of action and update the flow table.

²In the context of SDN, a switch refers to any network device that can use the OF protocol, not just layer 2 devices in the OSI model

II.2.3 SDN on Cloud Computing

Besides that sdn has emerged as a powerful approach for implementing network architectures in physical networks. It penetrated both virtualization and cloud computing environment. Whether on the physical network infrastructure or on the virtualized network infrastructure, the technical concept of SDN still the same. So, SDN controllers by using application programming interfaces (APIs) communicate with all the virtual appliances to manage and direct network traffic.

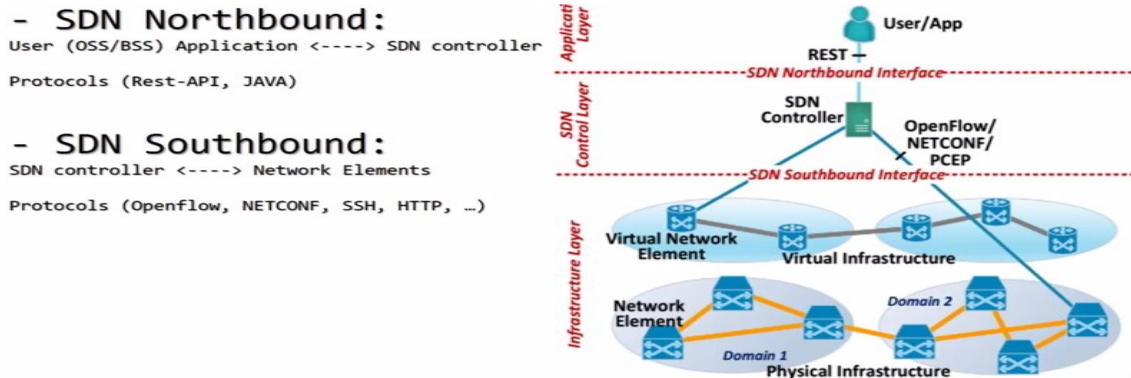


Figure II.3: SDN within Cloud Computing

II.2.4 Why SDN ?

The way we design and maintain networks is changing due to the use of software defined networking. SDN has been adopted by businesses more and more as a result of its many beneficial features, among its benefits:

- **Cost Reduction:** By obviating the need for expensive networking hardware. Since the intelligence is centralized at the SDN controller, it enables businesses to employ less expensive hardware for better effect. Knowing that, some SDN solutions are offered for free, and others come with a paid license such VMware NSX.
- **Ease of Management:** Network planning and setup are made easier with a single management panel that provides improved visibility into network resources. And, configuration's faults will be reduced progressively.
- **Centralized Network Processing:** It provides a single product for all networking demands and concentrates decision-making power at the controller, simplifying network components and reducing complexity. consequently, it provides services quicker and with greater agility.
- **Overhead Reduction:** By using the different methods of isolation that are accessible in the SDN, which is an important concept in the delivery of virtual machines, to provide separation for customer workloads. For example, setting up VLANs on various networking can be complex, but service providers can quickly isolate customer virtual machines with SDN.

- **Improved Network Security:** It make it simple to identify and respond to security threats by implementing security policies and monitoring network traffic from a central console. The problem of creating a single point of attack is also present, although this is resolved by applying clustering process on a virtual environment.
- **Traffic Control:** Managers of data centres can gain from utilising a single management solution to centralise networking control. Simultaneously, SDN offers a number of isolation options, such as establishing firewalls and ACLs at the NIC level of virtual machines. The SDN management panel also allows to set traffic rules, which aids in giving a complete control over network traffic.

II.3 Requirements of the Solution

For this project, Icosnet imposed a number of requirements to be met when choosing the right SDN solution. These requirements were based on the experience gained from the use of the NSX-t the paid SDN solution from VMware on another platform. They are numerated as follows:

1. The solution must be compatible with the existing architecture. In other words, with both KVM hypervisor and OpenNebula public cloud. SO, the SDN solution's testes must well work on the previous prototype prepared.
2. The solution must be capable of being integrated on the OpenNebula Sunstone.
3. The solution must be for free, open source and with a good documentation's availability.
4. The controller must be capable of providing a high performance isolation between different virtual instances. Either by using virtual local area network (VLAN), virtual extensible local area network (VXLAN) or another process. For guaranteed east-west disconnection.
5. In order to permit north-south connection, the solution must provide efficient network configuration control by providing both NAT and routing protocol control.
6. The solution must offer robust firewalling rule control in order to give security configuration control. So enabling the possibility of developing the solution in the future and include additional security features.

II.4 Choice of the Solution

There are several SDN solutions that satisfy these requirements and differ to some extent in certain criteria. In order to select the most suitable solution to be deployed in the data center, a comparison between the most efficient solutions has been made.

By performing extensive researches, we have adapted to these three controllers:

II.4.1 OpenDaylight

OpenDaylight (ODL) is an open source software defined networking platform that makes use of open protocols to offer network device monitoring and centralized programmatic control[16]. Prior to being hosted by the Linux Foundation, it was previously developed in 2013 as a joint effort between the International Business Machines corporation (IBM) and Cisco. It can be deployed on Linux-based distributions such as CentOS, Red Hat, and Ubuntu, as well as on Windows-based systems. ODL is a powerful and flexible SDN controller platform, that serves as a base for developing and automating networks of any size.



Figure II.4: OpenDaylight Logo

OpenDaylight includes an OpenFlow plugin, which enables communication with OpenFlow-enabled network devices. It includes a southbound plugin for routing protocols. Thus, ODL controller can be used on both virtual and physical infrastructure networks.

In addition to offering the requirements already mentioned, it offers a user interface for the control and the supervision of network components. Which makes it ideal to use in our project, this is why I try to install and manipulate Sulfur-SR3 version the 16th release(16.03) of ODL (it was the latest stable version). OpenDaylight is a Java³ program, so we need to install java and maven. I used java 11.0.17 and maven 3.9.0 (II.5).

```
vmopendaylight@vmopendaylight:~$ mvn -version
Apache Maven 3.9.0 (9b58d2bad23a66be161c4664ef21ce219c2c8584)
Maven home: /opt/maven
Java version: 11.0.17, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en_NG, platform encoding: UTF-8
OS name: "linux", version: "5.15.0-60-generic", arch: "amd64", family: "unix"
vmopendaylight@vmopendaylight:~$ java -version
openjdk version "11.0.17" 2022-10-18
OpenJDK Runtime Environment (build 11.0.17+8-post-Ubuntu-1ubuntu222.04)
OpenJDK 64-Bit Server VM (build 11.0.17+8-post-Ubuntu-1ubuntu222.04, mixed mode, sharing)
vmopendaylight@vmopendaylight:~$
```

Figure II.5: ODL Environment

Then I had properly installed it, following the instruction provided on the official web site:

But I still can't access to the OpenDaylight DLUX (OpenDaylight User Interface),

³Java is the programming language, and Java Runtime Environment (JRE) is the environment in which Java programs are running. While Apache Maven is a build automation tool for maintaining Java programs

exposing common control and visibility interfaces to the virtual networking layer. Open vSwitch supports many Linux-based virtualization technologies such: KVM and Virtual-Box. It supports both VLAN and VXLAN.



Figure II.8: OVS Logo

It has two modes of operation. The switching and forwarding functions are handled only by the first mode, known as normal mode. The second one, flow mode which uses the flow table to determine the packet forwarding rules. The OVS Controller is primarily in charge of managing this flow table, which enables great automation and abstraction when adding or removing control flows to accommodate network requirements[14].

The principal components of OVS are:

- **Openvswitch-switch:** The switch is implemented by the daemon `ovs-vswitchd` and a flow-based switching Linux kernel module.
- **ovsdb-server:** It is a small database server that `ovs-vswitchd` contacts to inquire more about its configuration.
- **Openvswitch-switch-dpdk:** It enables the utilization of the Data Plane Development Kit (DPDK), which consist on collection of libraries and drivers for quick user-space packet processing.
- **Openvswitch-testcontroller:** It is a simple useful SDN controller that can be used on tests though not for production, generally used with Mininet⁴. It can control a number of switches using openflow protocol.

```
root@LTSICOH2008:~# sudo apt search openvswitch
Sorting... Done
Full Text Search... Done
neutron-openvswitch-agent/jammy-updates,jammy-security 2:20.3.0-0ubuntu1.1 all
  Neutron is a virtual network service for Openstack - Open vSwitch plugin agent
openvswitch-common/jammy-updates,jammy-security 2.17.5-0ubuntu0.22.04.2 amd64
  Open vSwitch common components
openvswitch-dbg/jammy-updates,jammy-security 2.17.5-0ubuntu0.22.04.2 amd64
  Debug symbols for Open vSwitch packages
openvswitch-doc/jammy-updates,jammy-security 2.17.5-0ubuntu0.22.04.2 all
  Open vSwitch documentation
openvswitch-ipsec/jammy-updates,jammy-security 2.17.5-0ubuntu0.22.04.2 amd64
  Open vSwitch IPsec tunneling support
openvswitch-pki/jammy-updates,jammy-security 2.17.5-0ubuntu0.22.04.2 all
  Open vSwitch public key infrastructure dependency package
openvswitch-source/jammy-updates,jammy-security 2.17.5-0ubuntu0.22.04.2 all
  Open vSwitch source code
openvswitch-switch/jammy-updates,jammy-security 2.17.5-0ubuntu0.22.04.2 amd64
  Open vSwitch switch implementations
openvswitch-switch-dpdk/jammy-updates,jammy-security 2.17.5-0ubuntu0.22.04.2 amd64
  DPDK enabled Open vSwitch switch implementation
openvswitch-test/jammy-updates,jammy-security 2.17.5-0ubuntu0.22.04.2 all
  Open vSwitch test package
openvswitch-testcontroller/jammy-updates,jammy-security 2.17.5-0ubuntu0.22.04.2 amd64
  Simple controller for testing OpenFlow setups
openvswitch-vtep/jammy-updates,jammy-security 2.17.5-0ubuntu0.22.04.2 amd64
  Open vSwitch VTEP utilities
python3-openvswitch/jammy-updates,jammy-security 2.17.5-0ubuntu0.22.04.2 all
  Python 3 bindings for Open vSwitch
root@LTSICOH2008:~#
```

Figure II.9: Open vSwitch's Components

⁴It is an emulator that allows the creation of realistic virtual network instances for tests

Data Plane Development Kit (DPDK) devices can be combined with Open vSwitch, to increase capacities and optimise virtual switch performances. It is a software framework that offers a collection of libraries and drivers for quickening user space packet processing. So, DPDK lead to high performance of networking control, i.e. it boosts Sdn utilization.

Despite that Open vSwitch provides a power full virtual switch either for data centers, it doesn't offer its proper graphical user interface which complicates its use. Then, its proper controller doesn't offer a high performance of openflow control. So, we won't use all the SDN solution, but we appreciate the use of its virtual switch. Knowing that, we get a lot of problems while linking OVS with DPDK. So, we avoid its utilization, because we aim to use the resulting solution on a reeal datacenter that hosts virtual instances of important clients and we won't risk.

```
checking for struct tcf_t.firstuse... yes
checking whether dpdk is enabled... yes
checking for rte_config.h... yes
checking whether RTE_LIBRTE_VHOST_NUMA is declared... yes
checking for library containing get_mempolicy... -lnuma
checking whether RTE_EAL_NUMA_AWARE_HUGE_PAGES is declared... yes
checking for library containing get_mempolicy... (cached) -lnuma
checking whether RTE_NET_PCAP is declared... no
checking whether RTE_NET_AF_XDP is declared... no
checking whether RTE_LIBRTE_VHOST_NUMA is declared... (cached) yes
checking whether RTE_NET_MLX5 is declared... no
checking whether RTE_NET_MLX4 is declared... no
checking whether MAP_HUGE_SHIFT is declared... yes
checking for library containing dlopen... none required
checking whether linking with dpdk works... no
configure: error: Failed to link with DPDK, check the config.log for more details. If a working DPDK library was not found in the default search path, update PKG_CONFIG_PATH for pkg-config to find the .pc file in a non-standard location.
sdn@sdn01:~/usr/src/ovs$
```

Figure II.10: OVS-DPDK's Problem

II.4.3 ONOS

Open Network Operating System (ONOS) is an open source SDN controller. It is especially created to meet the needs of network service providers, it offers high levels of scalability, availability and performance. ONOS SDN serves as the control plane for both enterprise networks and service provider networks including campus LANs and data centre networks.

The ONOS is made to offer ease of support for new network services, and provision of SDN control for legacy OpenFlow-enabled devices. Service providers can grow their networks and add fresh components with the help of ONOS without impacting the rest of the system. Its distributed architecture decreases the risk of network failure, resulting in high network availability.

ONOS has a large community which has actively participated in its development. A new version of ONOS is published almost every three months, and its source code is written in Java.



Figure II.11: ONOS Logo

The open network operating system SDN solution provides several features[18], which include:

- **High Availability and Resiliency:** Which is a crucial key for CSP, stabilize the network connection using multiple mechanisms such clustering.
- **Performance at Scale:** It is buildings and architectures to offer an extreme efficiency. It is able to control and manage several devices and supports millions of applications intent search queries while maintaining less than 50 millisecond response time (or better) for network events, at its northbound interface.
- **Modular Software:** ONOS has been modularized to make software easier to read, test, maintain, and customize, with over 135 platform extensions available. It offers a lot of applications, and keep growing with each platform release.
- **Northbound Abstractions:** By using the API, ONOS simplifies the creation, deployment, and operation of configuration, management, and control applications.
- **Southbound Abstractions:** ONOS abstracts device characteristics for easy adaptation to legacy or new devices. ONOS southbound supports P4, OpenFlow, CLI, Network Management Protocol NETCONF, RESTCONF, Simple Network Management Protocol (SNMP), CLI, BGP and more,
- **GUI Framework and Base UI:** The ONOS GUI gives users access to a multi-layer network view and enables them to explore various network aspects such as connection, state, faults, and more.

In additions to Icosnet requirements, ONOS controller offers several other functionalities. Which make it the suitable controller to use.

II.5 Comparaison between the Solutions

We sum up the features of all the solutions on the following table: (table II.1)

	<i>OpenDaylight</i>	<i>Open vSwitch - DPDK</i>	<i>ONOS</i>
<i>Compatible with KVM</i>	Green	Green	Green
<i>Compatible with OpenNebula</i>	Green	Green	Green
<i>Open source</i>	Green	Green	Green
<i>For free</i>	Green	Green	Green
<i>With GUI</i>	Green	Red	Green
<i>Isolation between virtual instances</i>	Green	Green	Green
<i>Networking (routing protocols and NAT)</i>	Green	Red Doesn't allow routing protocols	Green
<i>Firewalling</i>	Green	Green	Green
<i>Robustness within a datacenter</i>	Green	Green	Green
<i>Availability of documentations</i>	Red	Green	Green

Table II.1: Comparison between the SDN Solutions

II.6 Conclusion

As seen in this chapter, software defined networking is the cutting-edge technology that resolve traditional networking's structural limitations. SDN separates the control plane from the data plane and allowing a centralised network devices control and management. In addition of its usage on physical infrastructure, SDN storm the cloud infrastructure and a lot of solutions has developed. ODL, OVS and ONOS are among them.

Moreover, during this chapter, we tried to highlight the specifications of each solution, by applying a comparison between them according to our structure requirements. As a result, the controller ONOS is the suitable controller while the OVS offers a high bridging performance with the advent of openflow protocol compared to Linux bridges which have a huge limitation with the configurations of openfLow protocol.

So we decide to combine the two solutions ONOS controller and OVS switch, and we eliminate the utilization of KVM bridge. In order to improve the cloud infrastructure and centralize the networking control. In the next chapter, we will focus on the implementations of these choices on the prototype of Icosnet cloud architecture.

Chapter III

Implementation of ONOS SDN Controller

III.1 Introduction

Across this chapter, we will implement the whole chosen solutions including both OVS bridge and ONOS controller on the basic cloud infrastructure. So, the rest of the chapter includes their installation, their configuration and their connection to the KVM hypervisor and the OpenNebula public cloud.

III.2 The Architecture

In order to improve and centralize the cloud networking, we have opted to the SDN technology. We discussed, in the previous chapter, the solutions that exist within a virtualized environment, and we choose the combination of ONOS SDN controller and OVS bridge to offer an enhanced quality. The utilisation of ONOS controller aims to centralize management and control of the network using an API. While the utilisation of OVS switch aims to replacing the Linux bridge which is generated automatically with the KVM, to resolve the issue that KVM bridge doesn't allow OpenFlow protocol and to improve the quality of service (QoS) because OVS switch works in two modes switching and forwarding mode and flow mode.

Here is an image (III.1) that illustrates the suggested architecture:

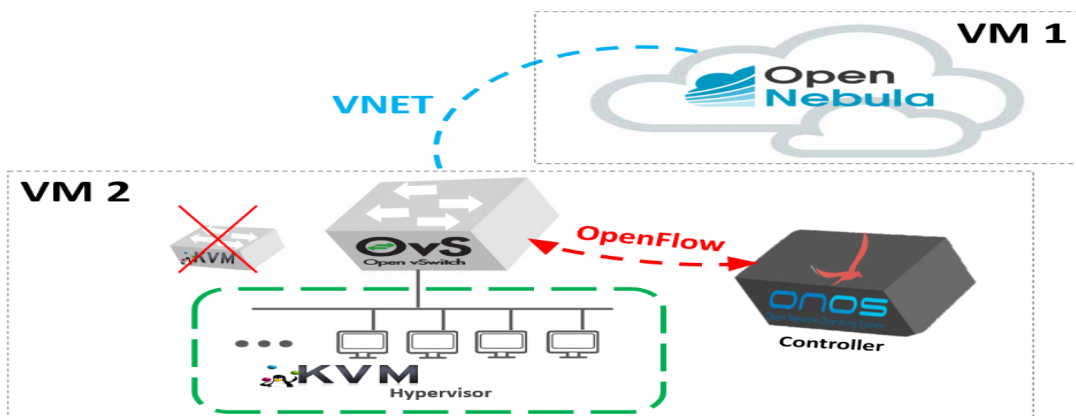


Figure III.1: The Suggested Architecture by Integrating SDN

Before implementing the solution on the real cluster, we will test it on the prototype prepared during the first chapter. So, the first Ubuntu 22.04 LTS machine that contains the public cloud OpenNebula will keep its initial status, i.e. no modification will be done on Opennebula front end's installation, and no new driver will be installed. We will just change some configuration thereafter. However, on the second Ubuntu 22.04 LTS machine, both OVS virtual switch and ONOS SDN controller are installed and configured, then the virtual machines already created and linked to the KVM bridge will be migrated to OVS switch and the KVM bridge will be ignored. We will be able, also, to link the new virtual machines created directly to Open vSwitch. After that, we will set up The ONOS SDN

controller to control all the virtual infrastructure, using OpenFlow protocol which works on TCP.

Our proposition is inspired by the standard architecture of software defined networking. Here is an illustration where we project the SDN architecture on the prototype:

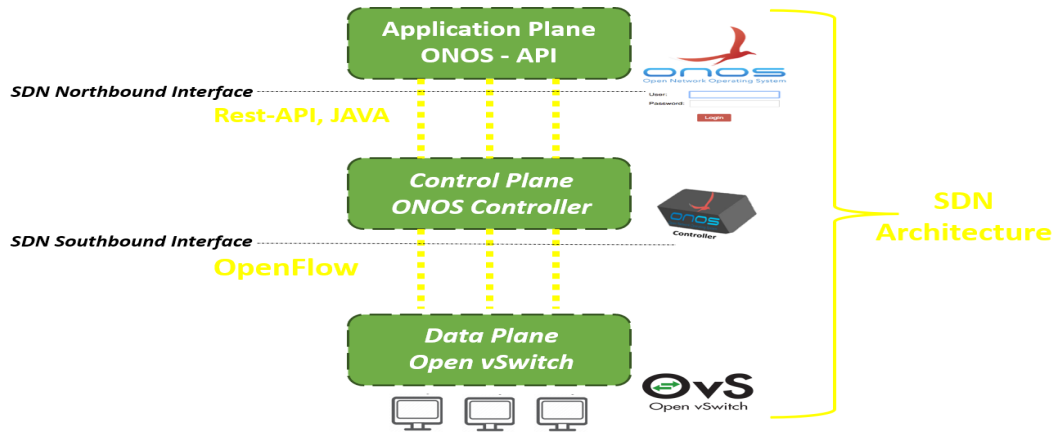


Figure III.2: Projection of the SDN Architecture on the Infrastructure

III.3 The integration

In order to simplify the work and easily detect configurations errors, While the integration, we work on two approaches: SDN-Virtualization and SDN-Cloud. The first one consists on configuring the whole bloc (controller and Ovs switch), and the second approach consists on configuring the bloc with OpenNebula public cloud.

III.3.1 SDN-Virtualization Part

Firstly, Open Network Operating System SDN controller will be installed. Secondly, Open vSwitch switch will be installed too. Thirdly, we configure OVS to take into consideration the ONOS as its controller. In this part, the presence of OpenNebula is totally ignored to simplify things, then the controller tests are applied using a manual management.

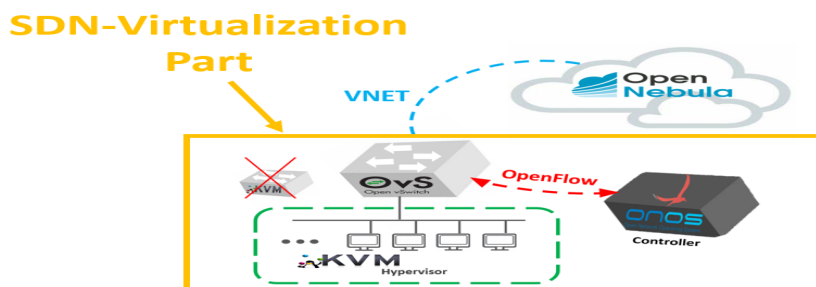


Figure III.3: SDN-Virtualization Part

III.3.1.1 Open Network Operating System (ONOS)

a) Requirement [18]

The following prerequisites should be satisfied in order to provide a basic execution environment:

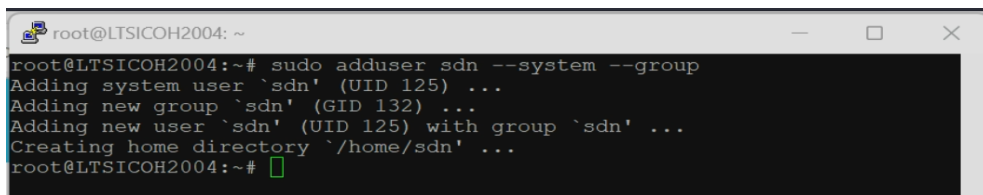
- 2 core CPU
- 2 GB RAM
- 10 GB hdd
- 1 NIC (any speed)

The following ports of the machine must be open in order for ONOS to provide the associated functionalities:

- **Port 8181:** For REST API and GUI
- **Port 8101:** To access the ONOS CLI
- **Port 6653:** For OpenFlow
- **Port 9876:** For intra-cluster communication (communication between target machines)

Running ONOS as root is not advised. Scripts used to operate ONOS as a service require a special unprivileged user (typically user "sdn") set up in the system. So, the sdn user is created and added to the sdn group using:

```
$ sudo adduser sdn --system --group
```



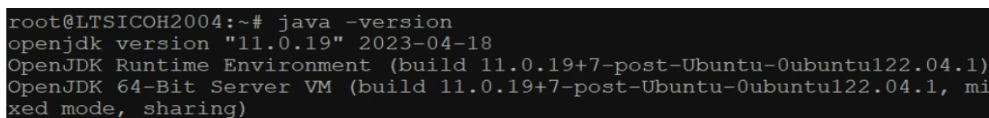
```
root@LTSICOH2004:~# sudo adduser sdn --system --group
Adding system user `sdn' (UID 125) ...
Adding new group `sdn' (GID 132) ...
Adding new user `sdn' (UID 125) with group `sdn' ...
Creating home directory `/home/sdn' ...
root@LTSICOH2004:~#
```

Figure III.4: Adding SDN User

ONOS is a platform built on Java. So, Java must be installed. These commands are used to install and verify its installation:

```
$ sudo apt update
```

```
$ sudo apt install default-jdk
```



```
root@LTSICOH2004:~# java -version
openjdk version "11.0.19" 2023-04-18
OpenJDK Runtime Environment (build 11.0.19+7-post-Ubuntu-0ubuntu122.04.1)
OpenJDK 64-Bit Server VM (build 11.0.19+7-post-Ubuntu-0ubuntu122.04.1, mixed mode, sharing)
```

Figure III.5: Java Version

We need to install the last version of Apache Maven 3.9.1 which serves as the build tool and manages dependencies of ONOS, using: `$ cd /opt/`

```
$ wget link_to_zip file_of_( apache-maven-3.9.1-bin.zip)
$ sudo unzip apache-maven-3.9.1-bin.zip
$ sudo mv apache-maven-3.9.1 maven
```

We set up environment variables for Maven¹, including necessary path to locate Java environment and the Maven installation directory for a better performance, using this command:

```
$ sudo nano /etc/profile.d/maven.sh

export JAVA_HOME=/usr/lib/jvm/default-java
export M2_HOME=/opt/maven
export MAVEN_HOME=/opt/maven
export PATH=${M2_HOME}/bin:${PATH}

$ source /etc/profile.d/maven.sh
```

```
root@LTSICOH2004:/opt# ls
apache-maven-3.9.1-bin.zip  maven
root@LTSICOH2004:/opt# sudo nano /etc/profile.d/maven.sh
root@LTSICOH2004:/opt# source /etc/profile.d/maven.sh
root@LTSICOH2004:/opt# mvn -version
Apache Maven 3.9.1 (2e178502fcd8bffc201671fb2537d0cb4b4cc58f8)
Maven home: /opt/maven
Java version: 11.0.19, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-open
jdk-amd64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.15.0-72-generic", arch: "amd64", family: "u
nix"
```

Figure III.6: Maven Version

Then, we install both curl² and git³ using:

```
$ sudo apt install git $ sudo apt-get install curl
```

ONOS should be installed under (/opt) directory:

```
$ sudo mkdir /opt
$ cd /opt
```

b) Installation

¹A project management tool primarily used for developing Java applications

²It provides a command-line utility for sending or receiving data to or from a server. It supports many protocols such as HTTP, HTTPS, FTP, and more

³Git is a distributed version control system that programmers use to collaborate while improving source code

We opt installing the latest Long Term Supported (LTS) Release which is ONOS 2.7.0 version named X-Wing (LTS) version. We install the ONOS tar.gz format, untar the ONOS archive and rename the extracted directory, using:

```
$ sudo wget -c link_to_ONOS_2.7.0.tar.gz
```

```
$ sudo tar xzf onos-2.7.0.tar.gz
```

```
$ sudo mv onos-2.7.0 onos
```

ONOS is running using its start-stop script: `$ /opt/onos/bin/onos-service start`

In production environments, it is required to configure ONOS to start running as a real Linux service according to our Ubuntu 22.04 LTS version. In order that the OS can start it automatically as part of the boot process and can restart it in case of caching, using these commands:

```
$ sudo cp /opt/onos/init/onos.initd /etc/init.d/onos
```

```
$ sudo update-rc.d onos defaults
```

```
$ sudo cp /opt/onos/init/onos.conf /etc/init/onos.conf
```

```
$ sudo cp /opt/onos/init/onos.service /etc/systemd/system/
```

```
$ sudo systemctl daemon-reload
```

```
$ sudo systemctl enable onos
```

ONOS options are configured, by adding the default user "sdn" and activate some necessary applications using: `$ sudo nano /opt/onos/options`

```
ONOS_USER=sdn
ONOS_APPS=drivers , openflow , gui2
```

We check the status of the ONOS service and we start it using:

```
$ sudo systemctl status onos.service
```

```
$ sudo systemctl start onos.service
```

```
root@LTSICOH2004:/opt# sudo systemctl status onos.service
● onos.service - Open Network Operating System
   Loaded: loaded (/etc/systemd/system/onos.service; enabled; vendor p
   Active: activating (start) since Sun 2023-05-28 10:35:54 UTC; 422ms
   Control PID: 12000 (onos)
   Tasks: 14 (limit: 4530)
   Memory: 18.4M
   CPU: 398ms
   CGroup: /system.slice/onos.service
           └─12000 /bin/bash /etc/init.d/onos start
           └─12008 sudo -n -u sdn /opt/onos/karaf/bin/status
           └─12009 /bin/sh /opt/onos/karaf/bin/karaf status
           └─12111 /usr/bin/java -Xms128M -Xmx512M -XX:+UnlockDiagnost
May 28 10:35:54 LTSICOH2004 systemd[1]: onos.service: Scheduled restart
May 28 10:35:54 LTSICOH2004 systemd[1]: Stopped Open Network Operating S
May 28 10:35:54 LTSICOH2004 systemd[1]: onos.service: Consumed 1.850s CP
May 28 10:35:54 LTSICOH2004 systemd[1]: Starting Open Network Operating
May 28 10:35:54 LTSICOH2004 sudo[12008]: root : PWD=/ ; USER=sdn ; C
May 28 10:35:54 LTSICOH2004 sudo[12008]: pam_unix(sudo:session): session
lines 1-13 / 13 (END)
root@LTSICOH2004:/opt# sudo systemctl start onos.service
root@LTSICOH2004:/opt#
root@LTSICOH2004:/opt# sudo systemctl status onos.service
● onos.service - Open Network Operating System
   Loaded: loaded (/etc/systemd/system/onos.service; enabled; vendor p
   Active: active (running) since Sun 2023-05-28 10:36:03 UTC; 478ms a
   Process: 12746 ExecStart=/etc/init.d/onos start (code=exited, status
   Main PID: 12880 (karaf)
   Tasks: 18 (limit: 4530)
   Memory: 20.9M
   CPU: 1.425s
   CGroup: /system.slice/onos.service
           └─12880 /bin/sh /opt/onos/apache-karaf-4.2.9/bin/karaf serv
           └─12973 /usr/bin/java -XX:+UseG1GC -XX:MaxGCPUseMillis=200
```

Figure III.7: ONOS Service Running

Allowing the "ssh-rsa" algorithm to be used for host key authentication during the Secure Shell (SSH) handshake process, by running:

```
$ cd /opt/onos/bin/ $ sudo nano ~/.ssh/config
```

```
HostKeyAlgorithms +ssh-rsa
```


Finally, we start interacting with ONOS via:

- **Command Line Interface (CLI):**

while the default login credentials are onos / rocks.

```
$ cd /opt/onos/  
$ ./bin/onos start  
$ ./bin/onos-service list  
$ ./bin/onos -l onos
```

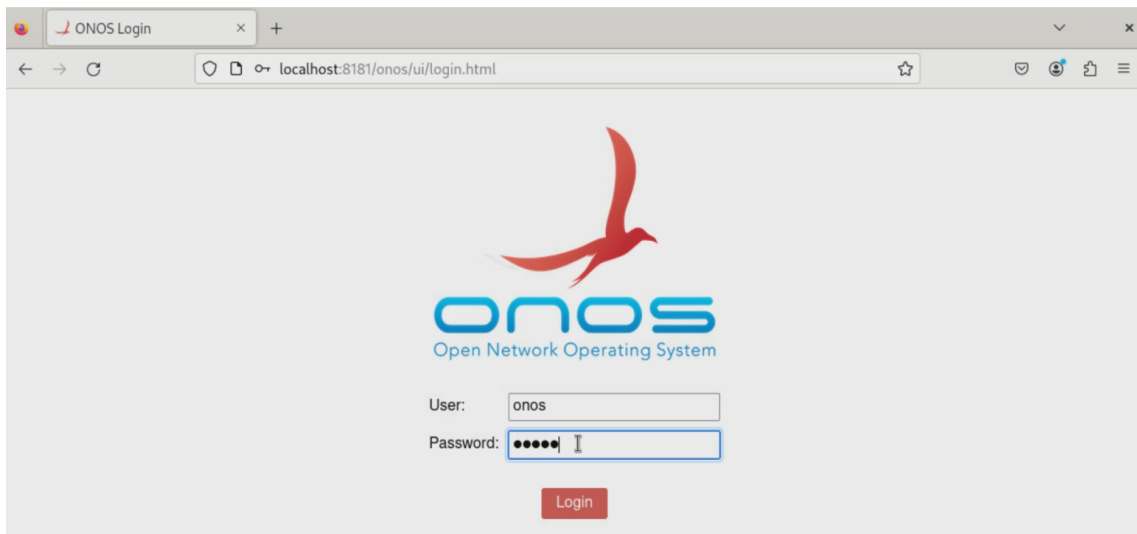
```
root@LTSICOH2004:~# cd /opt/onos/bin/  
root@LTSICOH2004:/opt/onos/bin#  
root@LTSICOH2004:/opt/onos/bin# ./onos -l onos  
Warning: Permanently added '[localhost]:8101' (RSA) to the list of known  
hosts.  
Password authentication  
(onos@localhost) Password:  
Welcome to Open Network Operating System (ONOS)!
```



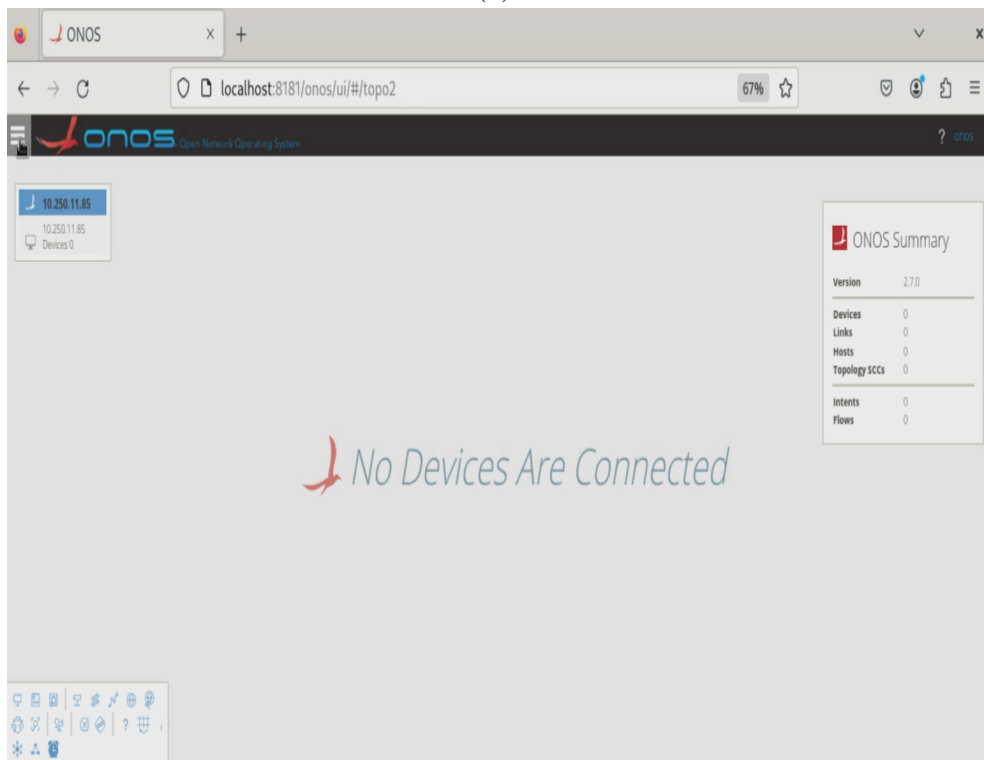
```
Documentation: wiki.onosproject.org  
Tutorials:     tutorials.onosproject.org  
Mailing lists: lists.onosproject.org  
  
Come help out! Find out how at: contribute.onosproject.org  
  
Hit '<tab>' for a list of available commands  
and '[cmd] --help' for help on a specific command.  
Hit '<ctrl-d>' or type 'logout' to exit ONOS session.  
  
onos@root >  
onos@root >  
onos@root > summary  
node=10.250.11.85, version=2.7.0 clusterId=default  
nodes=1, devices=0, links=0, hosts=0, SCC(s)=0, flows=0, intents=0  
onos@root >  
Connection to localhost closed.  
root@LTSICOH2004:/opt/onos/bin#
```

Figure III.8: ONOS CLI

- Graphical user interface (GUI):



(a)



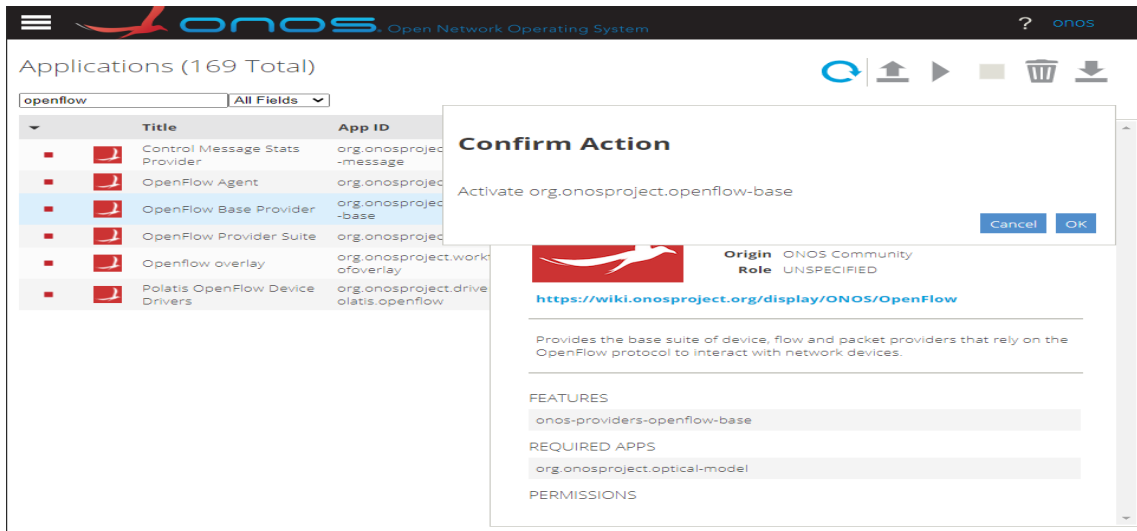
(b)

Figure III.9: ONOS GUI

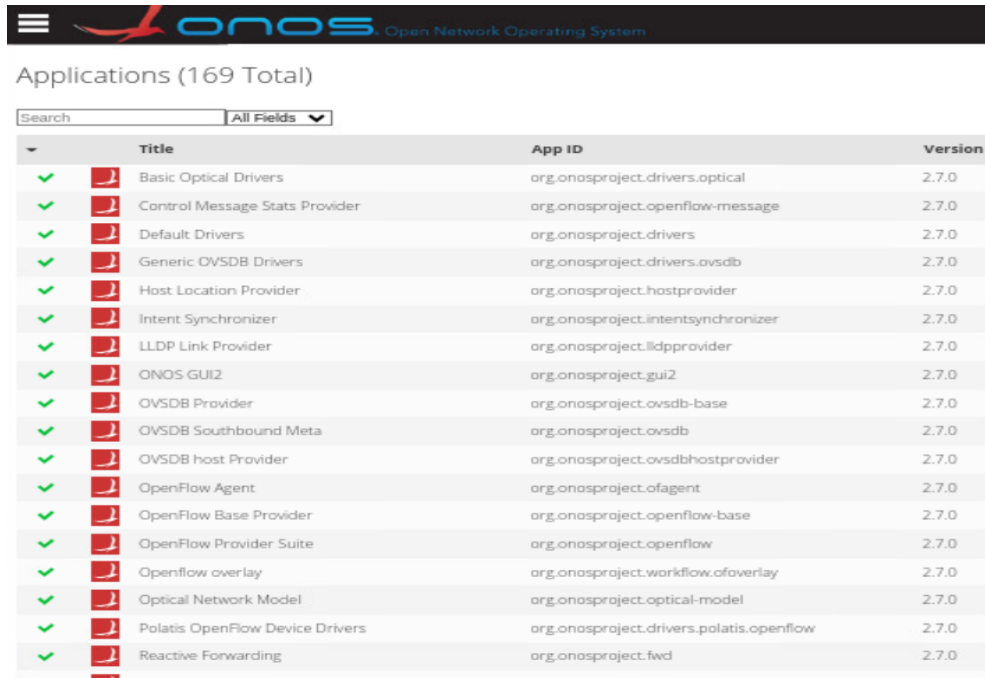
c) Configuration

In order to properly configure ONOS controller, some applications must be activated, initially, such as OpenFlow and Reactive Forwarding. (FigureIII.10)

Thereafter, the rest of the configuration of ONOS with Open vSwitch will be done after the complete installation of OVS.



(a)



(b)

Figure III.10: ONOS Activated Applications

III.3.1.2 Open vSwitch

a) Requirement

There is no need to install all the requirements individually while using "apt get install" which automatically manages package dependencies.

b) Installation

The installation consists of downloading both "openvswitch-switch" and "openvswitch-common" packages that include the core userspace components of the switch. Then starting openvswitch-switch service Using :

```

$ sudo apt-get install openvswitch-switch
$ sudo apt-get install openvswitch-common
$ sudo systemctl start openvswitch-switch.service

root@LTSICOH2008:~# sudo apt-get install openvswitch-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openvswitch-common is already the newest version (2.17.5-0ubuntu0.22.04.2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@LTSICOH2008:~# sudo systemctl status openvswitch-switch.service
● openvswitch-switch.service - Open vSwitch
   Loaded: loaded (/lib/systemd/system/openvswitch-switch.service)
   Active: active (exited) since Tue 2023-05-30 12:14:47 UTC; 1min 10s ago
   Process: 20605 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 20605 (code=exited, status=0/SUCCESS)
     CPU: 4ms

May 30 12:14:47 LTSICOH2008 systemd[1]: Starting Open vSwitch.
May 30 12:14:47 LTSICOH2008 systemd[1]: Finished Open vSwitch.

```

Figure III.11: OVS Service

c) Configuration

The Open vSwitch switch should be created and a physical interface of management should be linked to it using these commands:

```

$ sudo ovs-vsctl add-br ovsbr0
$ sudo ovs-vsctl add-port ovsbr0 ens192

```

```

7: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether fe:03:17:0a:6c:6e brd ff:ff:ff:ff:ff:ff
8: ovsbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether 00:50:56:a4:0a:a5 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::250:56ff:fea4:aa5/64 scope link
        valid lft forever preferred lft forever

```

Figure III.12: Screenshot of "ip a" Command Displaying OVS Switch

```

$ sudo ovs-vsctl show $ sudo ovs-ofctl show ovsbr0

```

```

oneadmin@LTSICOH2008:~$ sudo ovs-vsctl show
860762fb-438c-461a-8646-611c34a546cc
Bridge ovsbr0
  Port ens192
    Interface ens192
  Port ovsbr0
    Interface ovsbr0
    type: internal
  ovs_version: "2.17.5"
oneadmin@LTSICOH2008:~$ sudo ovs-ofctl show ovsbr0
OFPPT_FEATURES_REPLY (xid=0x2): dpid:0000005056a40aa5
  n_tables:254, n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
1(ens192): addr:00:50:56:a4:0a:a5
  config: 0
  state: 0
  current: 10GB-FD COPPER
  advertised: COPPER
  supported: 1GB-FD 10GB-FD COPPER
  speed: 10000 Mbps now, 10000 Mbps max
LOCAL(ovsbr0): addr:00:50:56:a4:0a:a5
  config: PORT_DOWN
  state: LINK_DOWN
  speed: 0 Mbps now, 0 Mbps max
OFPPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0

```

Figure III.13: OVS Configuration

A bridged Open vSwitch virtual network that link the VMs to the Open vSwitch switch should be created from an XML file :

```
$ sudo nano ovs-network.xml
```

```
<network>
  <name>ovs</name>
  <uuid>f58cad29-0455-439a-b533-8362669cec92</uuid>
  <forward mode='bridge' />
  <bridge name='ovsbr0' />
  <virtualport type='openvswitch' />
</network>
```

```
$ sudo virsh net-define ovs-network.xml
```

```
$ sudo virsh net-start ovs
```

```
$ sudo virsh net-autostart ovs
```

```
$ sudo virsh net-list --all
```

```
$ sudo virsh list --all
```

```
root@LTSICOH2004:~# virsh net-list --all
Name      State    Autostart Persistent
-----
default   active  yes         yes
root@LTSICOH2004:~# virsh net-define ovs-network.xml
Network ovs defined from ovs-network.xml
root@LTSICOH2004:~# virsh net-start ovs
Network ovs started
root@LTSICOH2004:~# virsh net-autostart ovs
Network ovs marked as autostarted
root@LTSICOH2004:~# virsh net-list --all
Name      State    Autostart Persistent
-----
default   active  yes         yes
ovs       active  yes         yes
root@LTSICOH2004:~# virsh list --all
Id      Name      State
-----
root@LTSICOH2004:~#
```

Figure III.14: Creating an OVS VNET

In order to add the VM to OVS VNET:

- **For VM already existed and attached to KVM bridge:**

It is necessary to modify the XML file of the VM already created to change the KVM bridge with the OVS switch using:

```
$ sudo virsh edit VM
```

```
<interface type='network'>
  <mac address='52:54:00:84:c7:4c' />
  <source network='ovs' />      <----- modify here
  <model type='virtio' />
  <address type='pci' domain='0x0000' bus='0x01'
    slot='0x00' function='0x0' />
```

```
</interface>
```

```
$ sudo virsh start VM
```

```
$ virsh list
```

- For new VM created:

Precise the ovs as mode of networking on the XML file of the new VM using:

```
$ sudo nano name_of_the_file.xml
```

```
<domain type='kvm'>
<name>name_of_the_VM</name>
<memory unit='KiB'>2097152</memory>
<vcpu placement='static'>2</vcpu>
<os>
  <type arch='x86_64' machine='pc-i440fx-2.1'>hvm</
  type>
  <boot dev='hd' />
  <boot dev='cdrom' />
</os>
<devices>
  <disk type='file' device='cdrom'>
    <driver name='qemu' type='raw' />
    <source file='/home/user/Downloads/ubuntu
      -22.04.2-live-server-amd64.iso' />
    <target dev='hdc' bus='ide' />
  </disk>
  <disk type='file' device='disk'>
    <driver name='qemu' type='qcow2' />
    <source file='/home/user/myvm.qcow2' />
    <target dev='vda' bus='virtio' />
    <address type='pci' domain='0x0000' bus='0x00'
      slot='0x04' function='0x0' />
  </disk>
  <interface type='network'>
    <mac address='52:54:00:01:02:03' />
    <source network='ovs' />
    <model type='virtio' />
    <address type='pci' domain='0x0000' bus='0x00'
      slot='0x03' function='0x0' />
  </interface>
  <serial type='pty'>
    <target port='0' />
  </serial>
```

```
<input type='mouse' bus='ps2' />
<graphics type='vnc' port='-1' autoport='yes' />
</devices>
</domain>
```

```
$ cd /home/user/
$ sudo qemu-img create -f qcow2 myvm.qcow2 20G
$ sudo virsh define name_of_the_file.xml
$ sudo virsh start name_of_the_VM
$ sudo ovs-vsctl show
```

Then we configure manually the IP address of OVS switch using :

```
$ sudo ip addr add 10.1.0.2/24 dev ovsbr0
$ sudo ip link set up dev ovsbr0
```

And the IP address of the VM created using "netplane" commands.

```
6: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc
c noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:d2:cb:99 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global v
irbr0
    valid_lft forever preferred_lft forever
17: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop st
ate DOWN group default qlen 1000
    link/ether fe:03:17:0a:6c:6e brd ff:ff:ff:ff:ff:ff
20: ovsbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UNKNOWN group default qlen 1000
    link/ether 00:50:56:a4:0a:a5 brd ff:ff:ff:ff:ff:ff
    inet 10.1.0.1/24 scope global ovsbr0
    valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fea4:aa5/64 scope link
    valid_lft forever preferred_lft forever
24: vnet3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
fq_codel master ovs-system state UNKNOWN group default qlen
1000
    link/ether fe:54:00:01:02:03 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::fc54:ff:fe01:203/64 scope link
    valid_lft forever preferred_lft forever
root@LTSICOH2008:~#
```

(a) " ip a "

```
root@LTSICOH2008:/home/user#
root@LTSICOH2008:/home/user# sudo ovs-vsctl show
61a082c4-125f-461e-b28a-98bdc28c6b72
    Bridge ovsbr0
        Port ovsbr0
            Interface ovsbr0
                type: internal
        Port vnet3
            Interface vnet3
        Port ens192
            Interface ens192
    ovs_version: "2.17.5"
root@LTSICOH2008:/home/user#
```

(b) " ovs-vsctl show "

Figure III.15: The Virtual Network Interface of the New VM Attached to OVS Virtual Network

III.3.1.3 OVS-ONOS


Here are the necessary commands for attachment of the controller to the switch:

```
$ sudo ovs-vsctl set Bridge ovsbr0 protocols=OpenFlow10
$ sudo ovs-vsctl set-controller ovsbr0 tcp:10.250.11.85:6633
$ sudo ovs-vsctl show
$ sudo ovs-ofctl show ovsbr0
$ sudo ovs-dpctl show
```

```
root@LTSICOH2008:/opt/onos/bin# sudo ovs-vsctl show
61a082c4-125f-461e-b28a-98bdc28c6b72
Bridge ovsbr0
  Controller "tcp:10.250.11.101:6633"
    is_connected: true
  Port ovsbr0
    Interface ovsbr0
      type: internal
  Port vnet3
    Interface vnet3
  Port ens192
    Interface ens192
  ovs version: "2.17.5"
root@LTSICOH2008:/opt/onos/bin#
```

(a) OVS CLI

```
Welcome to Open Network Operating System (ONOS)!
```



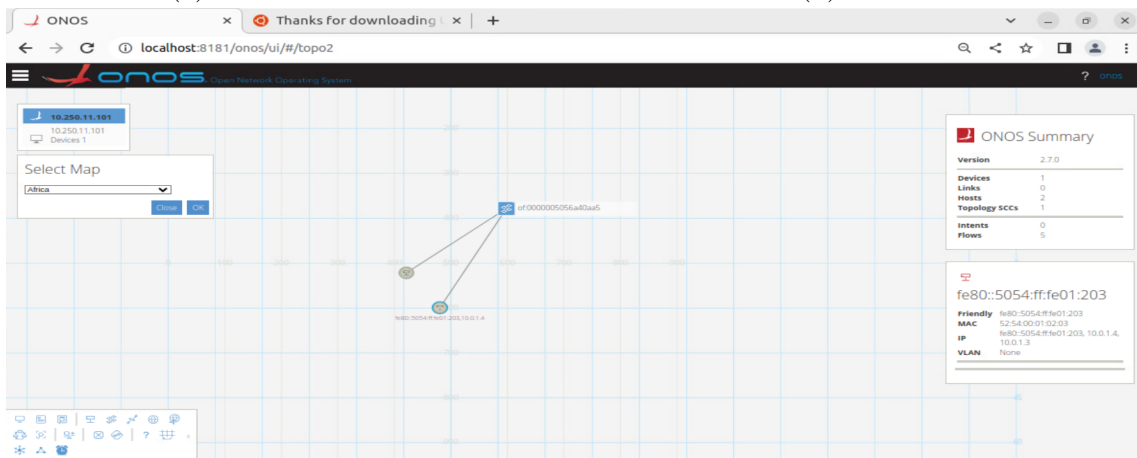
```
Documentation: wiki.onosproject.org
Tutorials: tutorials.onosproject.org
Mailing lists: lists.onosproject.org

Come help out! Find out how at: contribute.onosproject.org

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'logout' to exit ONOS session.

onos@root > summary
node=10.250.11.101, version=2.7.0 clusterId=default
nodes=1, devices=1, links=0, hosts=2, SCC(s)=1, flows=5, intents=0
onos@root > devices
id=of:0000005056a40aa5, available=true, local-status=connected 43m54s ago, role=
MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.17.5, serial=None,
chassis=5056a40aa5, driver=ovs, channelId=10.250.11.101:49658, datapathDescripti
on=None, managementAddress=10.250.11.101, protocol=OF_10
onos@root > hosts
id=00:50:56:A4:05:59/None, mac=00:50:56:A4:05:59, locations=[of:0000005056a40aa5
/2], auxLocations=null, vlan=None, ip(s)=[], innerVlan=None, outerTFID=unknown,
provider=of:org.onosproject.provider.host, configured=false
id=52:54:00:01:02:03/None, mac=52:54:00:01:02:03, locations=[of:0000005056a40aa5
/5], auxLocations=null, vlan=None, ip(s)=[fe80::5054:ff:fe01:203, 10.0.1.4, 10.0
.1.3], innerVlan=None, outerTFID=unknown, provider=of:org.onosproject.provider.h
ost, configured=false
onos@root >
```

(b) ONOS CLI



(c) ONOS GUI

Figure III.16: Attachment the OVS Bridge to the ONOS Controller

So, the SDN-Virtualization part is correctly established manually using line commands. Some tests of features will be done on the next chapter. While in the next section, we will integrate the OpenNebula cloud. So, rather than using commands configurations, the orchestrator will offset this.

III.3.2 SDN-Cloud Part

In this part, the necessary configuration of OpenNebula to could be attached to the

bloc (Open vSwitch switch + ONOS controller) will be done. So we principally interest to this part:

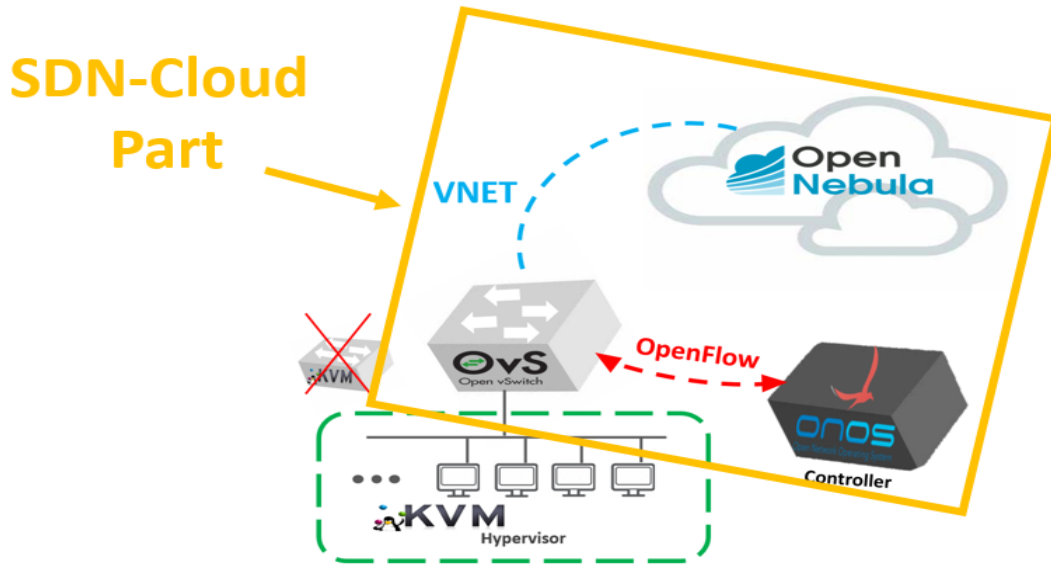


Figure III.17: SDN-Cloud Part

Firstly, The Open vSwitch network drivers must be configured on the OpenNebula Front-end by calculating VLAN-ID (all the VMs attached to the same OVS VNET will get the same VLAN-ID) on `"/etc/one/oned.conf"` , using oneadmin user:

```
$ sudo nano /etc/one/oned.conf
```

```
VLAN_IDS = [
    START      = "2",
    RESERVED   = "0, 1, 4095"
]
```

And by adjusting some parameters in `"/var/lib/one/remotes/etc/vnm/OpenNebulaNetwork.conf"` :

```
$ sudo nano /var/lib/one/remotes/etc/vnm/OpenNebulaNetwork.conf
```

```
:arp_cache_poisoning    true
:keep_empty_bridge      true
:ovs_bridge_conf
:stp_enable              true
```

```
$ su oneadmin
```

```
$ onehost sync -f
```

Secondly, there is no necessary configurations to do on the ONOS controller.

```

/var/lib/one/remotes/etc/vnm/OpenNebulaNetwork.conf *
# Security Group Options
#####
# Maximal number of entries in the IP set
:ipset_maxelem: 65536
#####
# Bridge and Interface Creation Options
#####
# Don't delete bridge with no virtual interfaces left
:keep_empty_bridge: true
# Following options will be added when creating bridge. For ex
#
# ip link add name <bridge name> type bridge stp_state 1
#
# :ip_bridge_conf:
#   :stp_state: on
#
# These options are set on the Ovs bridge. For example,
# this command will be triggered for the following option:
#
# ovs-vsctl set-bridge <bridge name> stp_enable=true
#
# :ovs_bridge_conf:
#   :stp_enable: true

```

(a)

```

root@LTSICOH2004:~# su oneadmin
oneadmin@LTSICOH2004:/root$ cd
oneadmin@LTSICOH2004:~$
onehost sync -f
* Adding 10.250.11.101 to upgrade
[=====] 1/1 10.250.11.101
All hosts updated successfully.
oneadmin@LTSICOH2004:~$

```

(b)

Figure III.18: Opennebula Front-end Configuration with OVS

Thirdly, in order to attach each VM to Open vSwitch, the creation and the network configuration of the VNET should be done on OpenNebula Sunstone. We attach the bridge created on OVS to the VNET, then we create the VM normally :

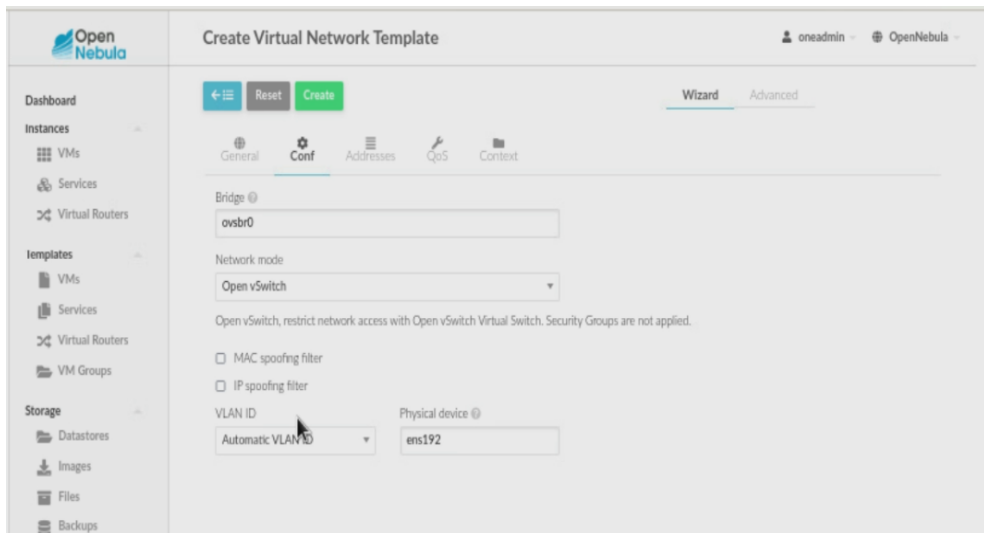
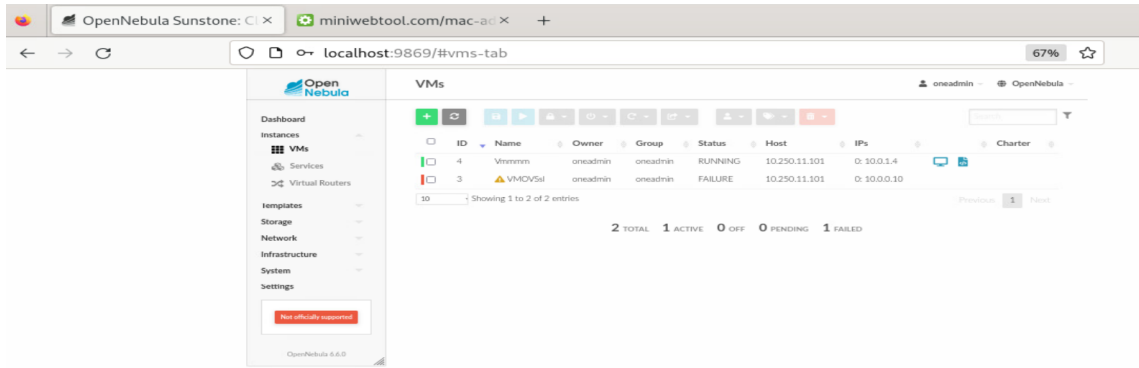
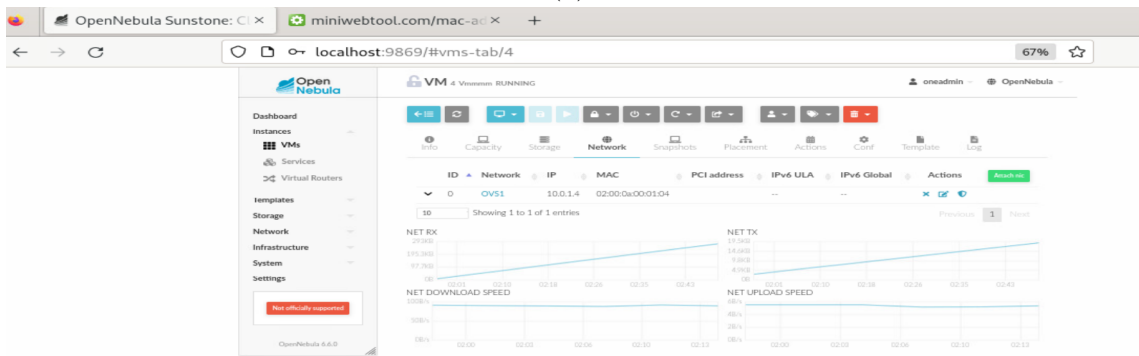


Figure III.19: VNET Open vSwitch

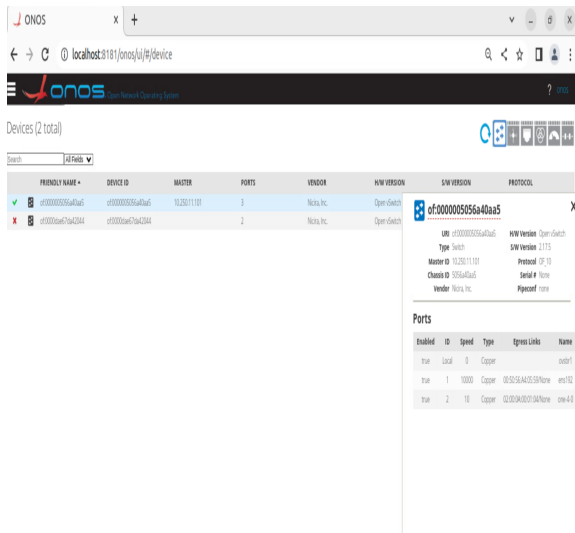
Here is the change that appears after deleting the bridge and the VM creating in the previous section, and creating a virtual machine linked to OVS VNET via OpenNebula sunstone:



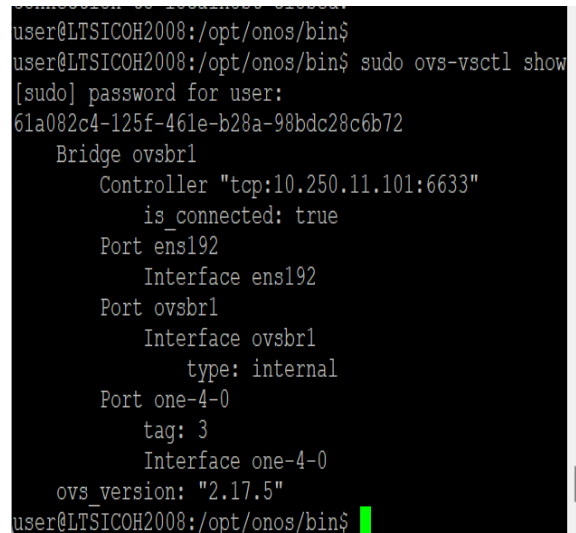
(a)



(b)



(c) ONOS GUI



(d) OVS CLI

Figure III.20: Managing the Controlled Network via OpenNebula Sunstone

III.4 Conclusion

At the end of this chapter, the whole chosen solution is appropriately implemented between the cloud and the hypervisor. Firstly, we decide the suitable architecture. Secondly, both ONOS controller and OVS virtual switch are installed on the host according to the requirements. Thirdly, The configuration is done following two approaches to avoid complication. Finally, via OpenNebula Sunstone we can correctly create and attach the

virtual machines to Open vSwitch VNET and control the virtual network using ONOS controller.

So, the conception and the deployment of the ONOS SDN Controller is well done, we will perform some tests and discuss the issues occurred at the last chapter.

Chapter IV

Results and Discussion

IV.1 Introduction

In the previous chapter, we integrate both Open vSwitch virtual switch and ONOS controller on the prototype used, within the OpenNebula cloud and the KVM virtual machine monitor. While within this chapter, we aim to analyse its capabilities and issues by performing some tests and to propose some solutions.

IV.2 Tests of the Solution

IV.2.1 Virtual Machines Isolation

The isolation between the clients VMs was the crucial need and requirement of the solution. According to the work achieved, the isolation is ensured through the ONOS Reactive Forwarding (FWD) application and Intent Framework.


IV.2.1.1 Reactive Forwarding

Reactive Forwarding on the Open Network Operating System controller refers to a mechanism that allows the controller to dynamically handle and react to network traffic by installing flow rules on network switches based on incoming packets.

A packet-in message is sent to the controller by the switch whenever a packet arrives at a switch and there is no flow rule in place to handle it. The controller then examines the packet to decide what should happen next. Reactive forwarding prevents the requirement to send every packet to the controller for processing by installing a flow rule in the switch to handle future packets with similar characteristics.

This reactive approach helps reduce controller overhead and improves network efficiency. In other words, reactive forwarding is a dynamic forwarding strategy employed by ONOS SDN controllers using OpenFlow protocol.

```
root@LTSICOH2008:/opt/onos/bin# ./onos -l onos
Password authentication
(onos@localhost) Password:
Welcome to Open Network Operating System (ONOS)!



Documentation: wiki.onosproject.org
Tutorials:    tutorials.onosproject.org
Mailing lists: lists.onosproject.org

Come help out! Find out how at: contribute.onosproject.org

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'logout' to exit ONOS session.

onos@root > summary
node=10.250.11.101, version=2.7.0 clusterId=default
nodes=1, devices=1, links=0, hosts=3, SCC(s)=1, flows=5, intents=0
onos@root > app deactivate org.onosproject.fwd
Deactivated org.onosproject.fwd
onos@root > app activate org.onosproject.fwd
Activated org.onosproject.fwd
onos@root > █
```

Figure IV.1: Reactive Forwarding within ONOS

IV.2.1.2 Intent Framework

The Intent Framework is a fundamental component of the ONOS SDN controller. It provides a high-level abstraction for defining the desired behaviour of the network in terms of intents, which represent the connectivity requirements or flow paths between network devices. It allows specifying network behaviour in form of policies, rather than mechanisms. Furthermore, it describes the desired outcome rather than how the outcome should be reached.

ONOS Intent forms the foundation for the scalability of ONOS in terms of network management and control. While creating an intent, it is translated into flow entries and installed in the switch's flow table, allowing the switch to forward packets according to the defined intents without involving the controller for every packet.

```
onos@root > add
add-host-intent (Installs host-to-host connectivity intent)
add-multi-to-single-intent (Installs connectivity intent between multiple ingress devices and a single egress device)
add-optical-intent (Installs optical connectivity intent)
add-point-intent (Installs point-to-point connectivity intent)
add-protected-transport (Adds ProtectedTransportIntent)
add-single-to-multi-intent (Installs connectivity intent between a single ingress device and multiple egress devices)
add-test-flows (Installs a number of test flow rules - for testing only)
add-vnet-intent (Installs virtual network connectivity intent)
```

Figure IV.2: The Types of ONOS Intents

IV.2.1.3 Implementation

In order to isolate all the virtual machines and connect just 2 VMs created on the KVM host, these steps should be followed:

- 2 VMs are created with the same manner as in the previous chapters

```
root@LTSICOH2008:/opt/onos/bin# ovs-vsctl show
61a082c4-125f-461e-b28a-98bdc28c6b72
    Bridge ovsbr0
        Controller "tcp:10.250.11.101:6633"
        is_connected: true
    Port vnet10
        Interface vnet10
    Port ovsbr0
        Interface ovsbr0
        type: internal
    Port vnet3
        Interface vnet3
    Port ens192
        Interface ens192
    ovs_version: "2.17.5"
```

Figure IV.3: Displaying the Creation of the VMs Using the Command "ovs-vsctl show"

- The VMs are automatically controlled by ONOS SDN controller.

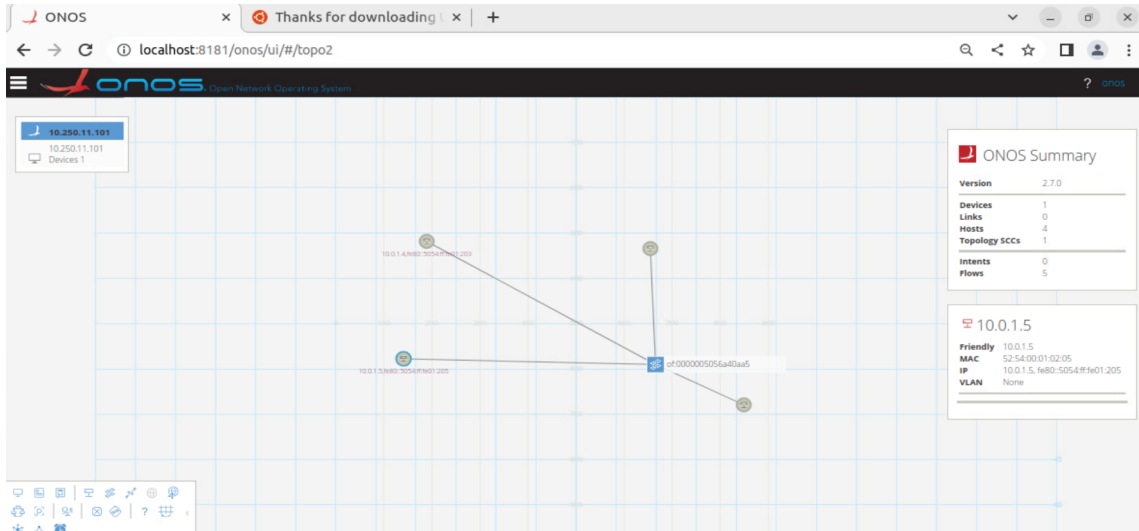


Figure IV.4: Displaying the Two VMs on ONOS GUI

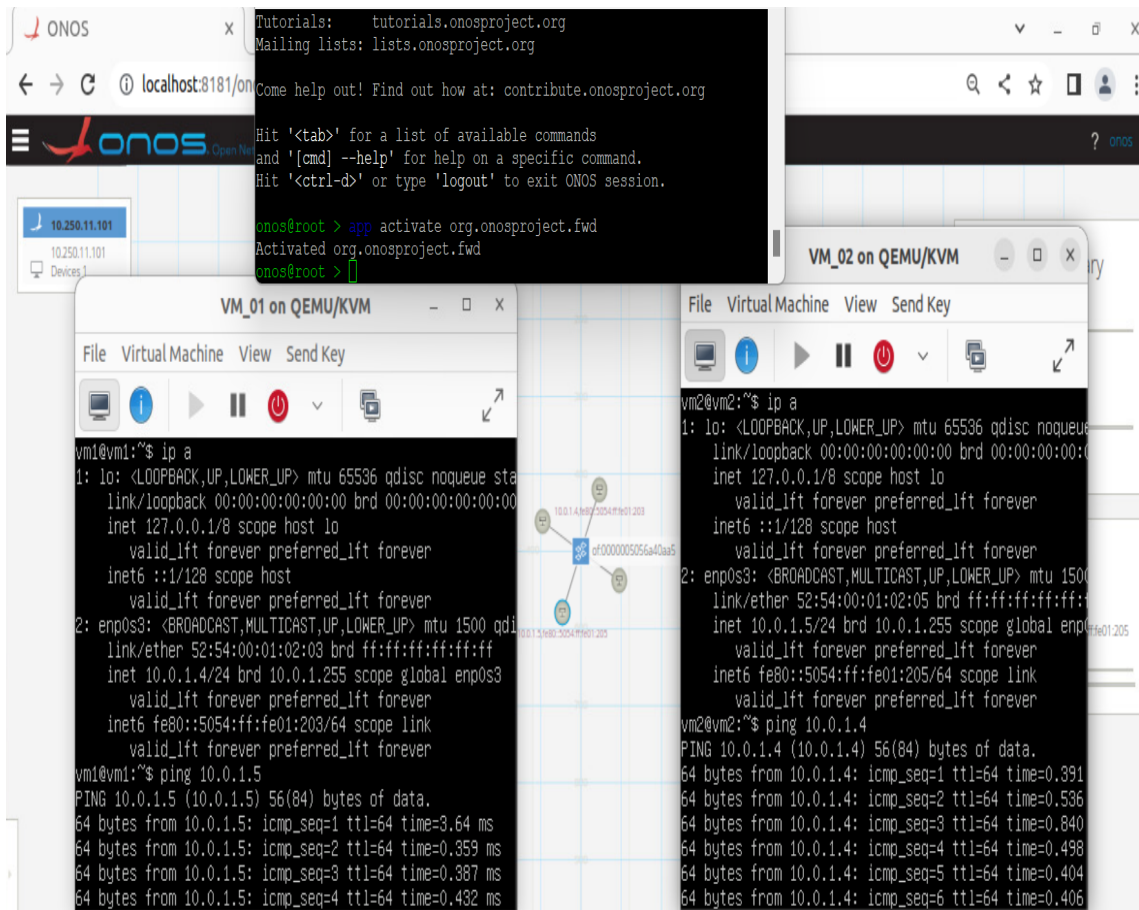


Figure IV.5: Connection Establishment between the 2 VMs by activating the FWD

- The connection between the 2VMs is established by activating the Fwd. (FigIV.5)
- The 2 VMs are disconnected because reactive forwarding is deactivating using.

```
$ onos@root > app deactivate org.onosproject.fwd
```

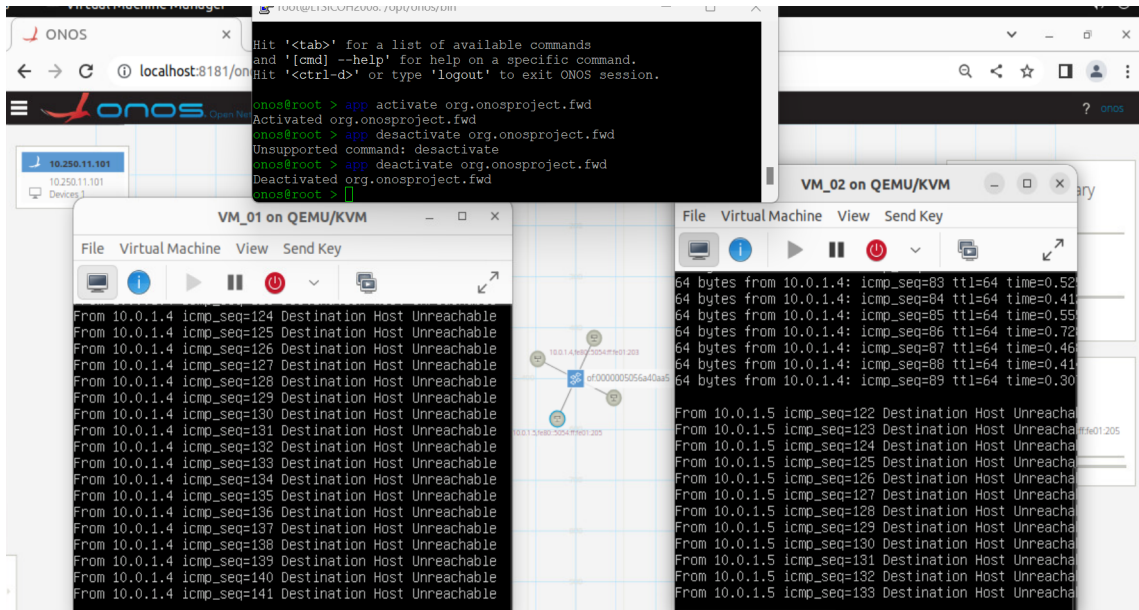


Figure IV.6: Disconnection between the VMs because of the FWD Deactivation

- An intent is created between the two hosts using :

```
onos@root> add-host-intent 52:54:00:01:02:03/None
52:54:00:01:02:05/None
```

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	31,780	5	0	ETH_TYPE:arp	imm[OUTPUT.CONTROLLER], cleared:true	*core
Added	0	31,780	40000	0	ETH_TYPE:bdp	imm[OUTPUT.CONTROLLER], cleared:true	*core
Added	25	102	100	0	IN_PORT:12, ETH_DST:52:54:00:01:02:03, ETH_SRC:52:54:00:01:02:05	imm[OUTPUT:5], cleared:false	*net.intent
Added	38	102	100	0	IN_PORT:5, ETH_DST:52:54:00:01:02:05, ETH_SRC:52:54:00:01:02:03	imm[OUTPUT:12], cleared:false	*net.intent
Added	66	120	5	0	ETH_TYPE:ipv4	imm[OUTPUT.CONTROLLER], cleared:true	*core
Added	1,064	31,780	40000	0	ETH_TYPE:ldp	imm[OUTPUT.CONTROLLER], cleared:true	*core
Added	5,065	31,780	40000	0	ETH_TYPE:arp	imm[OUTPUT.CONTROLLER], cleared:true	*core

(a)

```
onos@root > intents
Id: 0x0
State: INSTALLED
Key: 0x0
Intent type: HostToHostIntent
Application Id: org.onosproject.cli
Leader Id: 10.250.11.101
Resources: [52:54:00:01:02:03/None, 52:54:00:01:02:05/None]
Treatment: [NOACTION]
Constraints: [LinkTypeConstraint{inclusive=false, types=[OPTICAL]}]
Source host: 52:54:00:01:02:03/None
Destination host: 52:54:00:01:02:05/None
```

(b)

Figure IV.7: Intent's Flow

- The connection is again established through the intent:

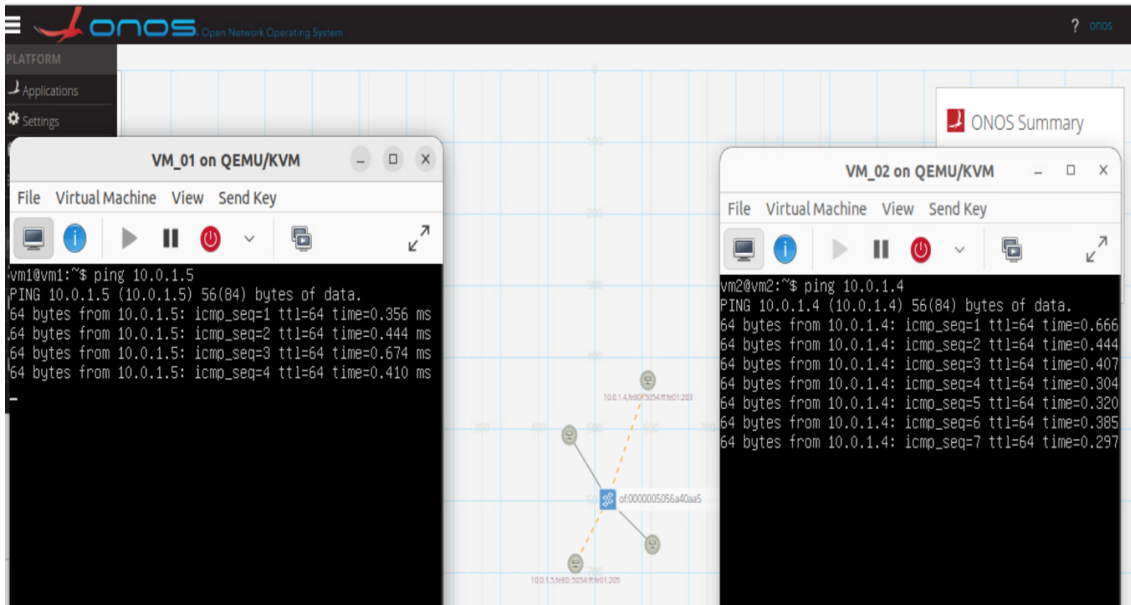


Figure IV.8: VMs Connected through Intent

In this way, all the virtual machines are isolated, even if they are in the same subnet, and only the VMs wanting to connect via intent ping each other.

IV.2.2 Necessary Commands

The CLI and GUI are the most important and useful interfaces for managing the virtual network. The GUI is a web application that provides a visual interface for the ONOS controller. On the other hand, the CLI is the main interface for configuring and managing the various aspects of running ONOS instances via numbers of commands which used to:

- Provide a detailed overview about the controlled network:

```
$ onos@root > summary
$ onos@root > nodes
$ onos@root > devices
$ onos@root > hosts
$ onos@root > flows
```

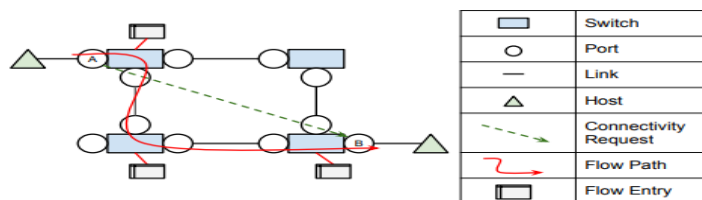


Figure IV.9: Network View: Connectivity Requests Cause Flow [19]

```

onos@root > summary
node=10.250.11.101, version=2.7.0 clusterId=default
nodes=1, devices=1, links=0, hosts=4, SCC(s)=1, flows=5, intents=0
onos@root > nodes
id=10.250.11.101, address=10.250.11.101:9876, state=ACTIVE, version=2.7.0, updated=1d4h ago *
onos@root > hosts
id=00:50:56:A4:05:59/None, mac=00:50:56:A4:05:59, locations=[of:0000005056a40aa5/2], auxLocations=null
, vlan=None, ip(s)=[], innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, c
onfigured=false
id=00:50:56:A4:5D:E4/None, mac=00:50:56:A4:5D:E4, locations=[of:0000005056a40aa5/2], auxLocations=null
, vlan=None, ip(s)=[], innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider.host, c
onfigured=false
id=52:54:00:01:02:03/None, mac=52:54:00:01:02:03, locations=[of:0000005056a40aa5/5], auxLocations=null
, vlan=None, ip(s)=[10.0.1.4], innerVlan=None, outerTPID=unknown, provider=of:org.onosproject.provider
.host, configured=false
id=52:54:00:01:02:05/None, mac=52:54:00:01:02:05, locations=[of:0000005056a40aa5/12], auxLocations=nul
l, vlan=None, ip(s)=[10.0.1.5, fe80::5054:ff:fe01:205], innerVlan=None, outerTPID=unknown, provider=of
:org.onosproject.provider.host, configured=false
onos@root > devices
id=of:0000005056a40aa5, available=true, local-status=connected 8h10m ago, role=MASTER, type=SWITCH, mf
r=Nicira, Inc., hw=Open vSwitch, sw=2.17.5, serial=None, chassis=5056a40aa5, driver=ovs, channelId=10.
250.11.101:49658, datapathDescription=None, managementAddress=10.250.11.101, protocol=OF_10
onos@root > flows
deviceId=of:0000005056a40aa5, flowRuleCount=5
  id=10000487112c4, state=ADDED, bytes=0, packets=0, duration=29425, liveType=UNKNOWN, priority=4000
0, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:bddp], treatment=DefaultTrafficTreatment{
immediate=[OUTPUT:CONTROLLER], deferred=[], transition=None, meter=[], cleared=true, StatTrigger=null,
metadata=null}
  id=100006eeebd82, state=ADDED, bytes=389075, packets=985, duration=29425, liveType=UNKNOWN, priori
ty=40000, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:lldp], treatment=DefaultTrafficTre
atment{immediate=[OUTPUT:CONTROLLER], deferred=[], transition=None, meter=[], cleared=true, StatTrigge
r=null, metadata=null}
  id=10000ebb2cb83, state=ADDED, bytes=95760, packets=2280, duration=29425, liveType=UNKNOWN, priori
ty=40000, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:arp], treatment=DefaultTrafficTrea
tment{immediate=[OUTPUT:CONTROLLER], deferred=[], transition=None, meter=[], cleared=true, StatTrigger
=null, metadata=null}
  id=10000485e46f4, state=ADDED, bytes=339934, packets=4111, duration=5374, liveType=UNKNOWN, priori
ty=5, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:ipv4], treatment=DefaultTrafficTreatme
nt{immediate=[OUTPUT:CONTROLLER], deferred=[], transition=None, meter=[], cleared=true, StatTrigger=nu
ll, metadata=null}
  id=10000f789c730, state=ADDED, bytes=0, packets=0, duration=29425, liveType=UNKNOWN, priority=5, t
ableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:arp], treatment=DefaultTrafficTreatment{immed
iate=[OUTPUT:CONTROLLER], deferred=[], transition=None, meter=[], cleared=true, StatTrigger=null, meta
data=null}
onos@root >

```

Figure IV.10: GUI ONOS Network Overview

```
$ onos@root > masters
```

```

onos@root > masters
10.250.11.101: 1 devices
  of:0000005056a40aa5
onos@root > devices
id=of:0000005056a40aa5, available=true, local-status=connected 8h12m ago, role=MASTER, type=SWITCH, mf
r=Nicira, Inc., hw=Open vSwitch, sw=2.17.5, serial=None, chassis=5056a40aa5, driver=ovs, channelId=10.
250.11.101:49658, datapathDescription=None, managementAddress=10.250.11.101, protocol=OF_10
onos@root >

```

Figure IV.11: Cluster's Informations

- Get informations and configure network routing and security, using:

```
onos@root > bgp-routes
  Network          Next Hop          Origin LocalPref      MED BGP-ID
Total BGP IPv4 routes = 0

  Network          Next Hop          Origin LocalPref      MED BGP-ID
Total BGP IPv6 routes = 0
onos@root > bgp-
  bgp-neighbors          (Lists the BGP neighbors)
  bgp-peer-add (Adds an external BGP router as peer to an existing BGP ...)
  bgp-peer-remove          (Removes a BGP peer)
  bgp-routes          (Lists all BGP best routes)
  bgp-speaker-add          (Adds an internal BGP speaker)
  bgp-speaker-remove          (Removes an internal BGP speaker)
  bgp-speakers          (Lists all BGP speakers)
```

(a) BGP Routing

```
onos@root > proxy-list
URL | ProxyTo | Balancing Policy
onos@root > proxy-
  proxy-add          (Add a new HTTP proxy)
  proxy-balancing-list (List the available balancing policies)
  proxy-list          (List the HTTP proxies)
  proxy-remove          (Remove an existing HTTP proxy)
```

(b) Virtual Proxy

Figure IV.12: Routing and Security Configurations

- Delete intents, using:

```
$ onos@root > remove-intent -p org.onosproject.cli intent_ID
```

```
onos@root > intents
Id: 0x19
State: INSTALLED
Key: 0x19
Intent type: HostToHostIntent
Application Id: org.onosproject.cli
Leader Id: 10.250.11.101
Resources: [52:54:00:01:02:03/None, 52:54:00:01:02:05/None]
Treatment: [NOACTION]
Constraints: [LinkTypeConstraint{inclusive=false, types=[OPTICAL]}]
Source host: 52:54:00:01:02:03/None
Destination host: 52:54:00:01:02:05/None

onos@root > remove-intent org.onosproject.cli 0x19
onos@root > intents
Id: 0x19
State: WITHDRAWN
Key: 0x19
Intent type: HostToHostIntent
Application Id: org.onosproject.cli
Leader Id: 10.250.11.101
Resources: [52:54:00:01:02:03/None, 52:54:00:01:02:05/None]
Treatment: [NOACTION]
Constraints: [LinkTypeConstraint{inclusive=false, types=[OPTICAL]}]
Source host: 52:54:00:01:02:03/None
Destination host: 52:54:00:01:02:05/None

onos@root > remove-intent -p org.onosproject.cli 0x19
Using "sync" to remove/purge intents - this may take a while...
Check "summary" to see remove/purge progress.
onos@root > remove-intent -p org.onosproject.cli 0x19
Using "sync" to remove/purge intents - this may take a while...
Check "summary" to see remove/purge progress.
onos@root > intents
onos@root >
```

Figure IV.13: Managing Intents

IV.2.3 ONOS REST API

To facilitate the management and configuration of the network, ONOS provides various access tools, including the Representational State Transfer Application Programming Interface (REST API). This API allows for the addition and removal of network rules and can be accessed at <http://localhost:8181/onos/v1/>. By examining the source code of this application, we can extract the following API endpoints:

- GET: Get all rules
- POST: Add a new rule
- DELETE: Remove a rule

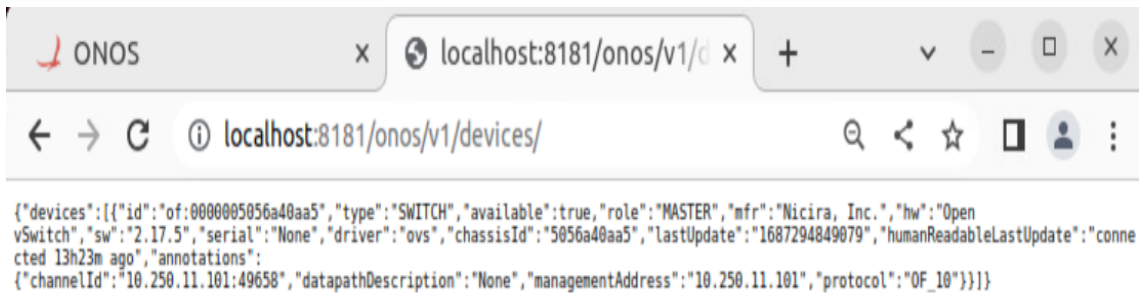


Figure IV.14: ONOS API

Rather than using ONOS CLI for configuration, we can simply use "curl" command or Postman to send rules to the Rest API, under the form of JavaScript Object Notation (JSON) or Python file. This method is designed more for programmers to customise the solution.

Here is an example of how configuring NAT using curl and json file:

```
curl -X POST --user onos:rocks --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{
  "priority": 40000,
  "isPermanent": true,
  "selector": {
    "ipv4Src": "10.0.0.10/24",
    "ethType": "0x0800"
  },
  "treatment": {
    "setField": {
      "ipv4Dst": "203.0.113.10"
    }
  }
}
```

```

"setField": {
  "ethDst": "12:34:56:78:9a:bc"
},
"nat": {
  "natAction": "dnat",
  "natAddresses": [
    {
      "networkAddress": "203.0.113.10",
      "prefixLength": 24
    }
  ]
} } } }' 'http://localhost:8181/onos/v1/flows/of:0000005056
a4bb51?appId=org.onosproject.openflow'

```

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	4.110	5	0	ETH_TYPE:ip4	imm[OUTPUT.CONTROLLER], cleared:true	*core
Added	0	4.110	5	0	ETH_TYPE:arp	imm[OUTPUT.CONTROLLER], cleared:true	*core
Added	0	4.110	40000	0	ETH_TYPE:bddp	imm[OUTPUT.CONTROLLER], cleared:true	*core
Added	0	4.110	40000	0	ETH_TYPE:lldp	imm[OUTPUT.CONTROLLER], cleared:true	*core
Added	387	4.109	40000	0	(No traffic selector criteria for this flow)	imm[NOACTION], cleared:true	*openflow

Figure IV.15: ONOS Flows

IV.3 Problems Experienced and Solutions

IV.3.1 Problems

Several challenges were encountered throughout this project, including:

- The challenge of unknowing the appropriate and compatible versions between ONOS, OVS and OpenNebula and the appropriate version of openFlow (10, 13, 15) that provides the necessary recommendations, because of the lack of the information.

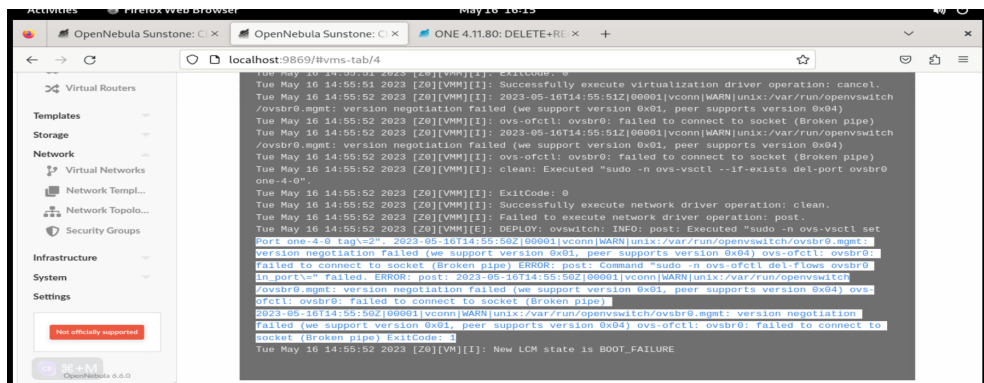
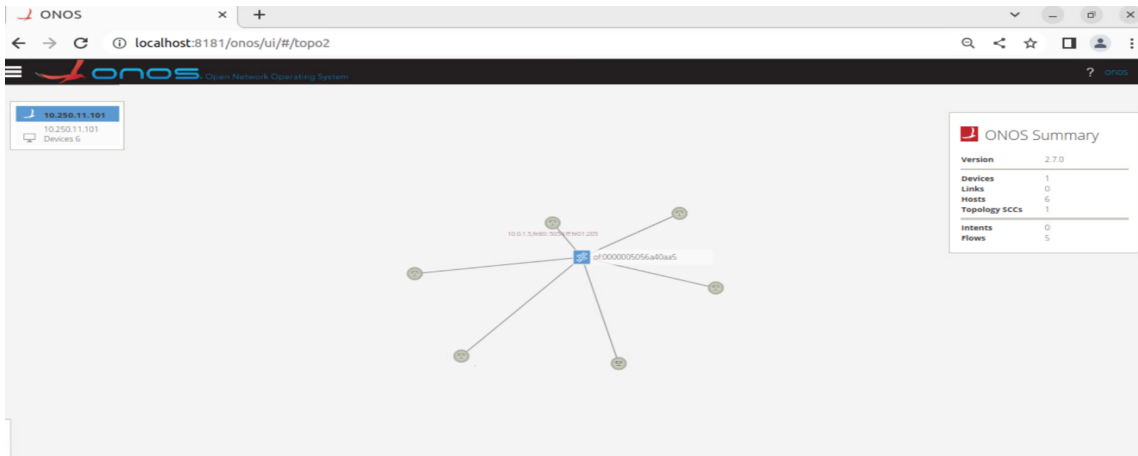


Figure IV.16: Issue of Incompatibility between the Versions

- Icosnet's security requirement that obliges closing some ports and disabling some protocols fearing of cyberattacks on the two VMs, which disable, for example, OpenNebula Fire-edge public endpoint access.
- Nested Virtualization of the two VMs used as a prototype, disable a lot of features and engender compilation problems. For example, while opening the VM's ports for remote access, the ONOS controller is able to control the devices on the first layer of abstraction, which endanger security problems.



(a) ONOS GUI

```
root@LTSICOH2005:~# sudo ovs-vsctl show
3293fec8-22d4-45f1-8d78-13cb973db9da
Bridge ovsbr0
  Controller "tcp:10.0.0.2:6653"
    is connected: true
  Port vnet17
    Interface vnet17
      error: "could not open network device vnet17 (No such device)"
  Port ens192
    Interface ens192
  Port vnet20
    Interface vnet20
      error: "could not open network device vnet20 (No such device)"
  Port vnet7
    Interface vnet7
      error: "could not open network device vnet7 (No such device)"
  Port vnet1
    Interface vnet1
  Port ovsbr0
    Interface ovsbr0
      type: internal
  Port vnet2
    Interface vnet2
      error: "could not open network device vnet2 (No such device)"
  Port vnet3
    Interface vnet3
      error: "could not open network device vnet3 (No such device)"
  Port vnet9
    Interface vnet9
      error: "could not open network device vnet9 (No such device)"
  Port vnet8
    Interface vnet8
      error: "could not open network device vnet8 (No such device)"
  Port vnet0
    Interface vnet0
  ovs version: "2.17.5"
root@LTSICOH2005:~#
```

(b) OVS CLI

Figure IV.17: Effect of the Nested Virtualization

IV.3.2 Solutions

In real implementations, the controller will be used directly on the server hardware,

which eliminates the main obstacle of nested virtualisation, and console access reduces security risks. We were compelled to test many product versions to create a matrix of compatibility.

IV.4 Propositions for a Real Implementation

During our project, we integrate the solution only on one KVM host machine, i.e. the controller on the KVM server. While on the production environment, Icosnet hosts its machines within a KVM cluster. In order to ensure a high availability and load balancing of controllers, it is recommended to realize an SDN cluster too.

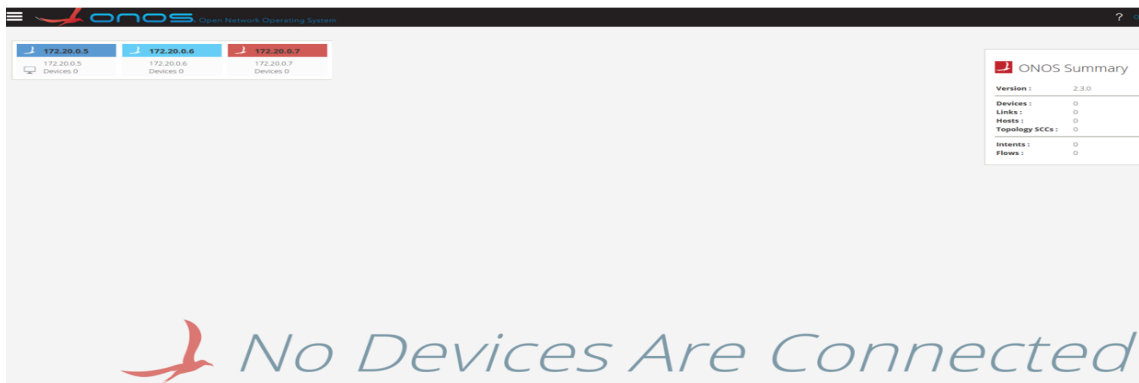


Figure IV.18: ONOS Cluster[18]

Our approach using the virtual switch and the controller proved to be effective at the switching level, i.e., at the second layer of the OSI model of the computer network, while Icosnet still use a physical router and firewall. In order to centralize all the network and improve the performances, we propose eliminating the hardware and replacing them with an OpenNebula virtual router and an ONOS proxy.

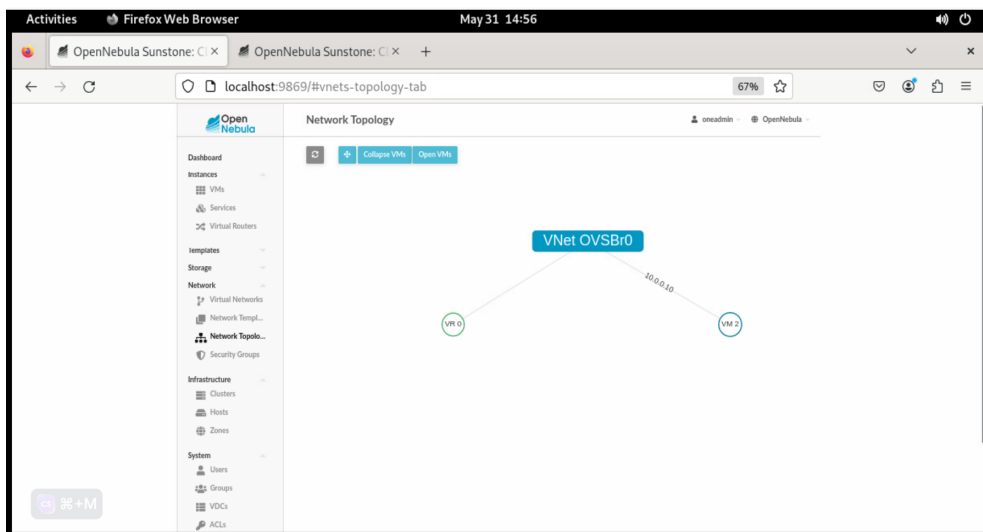


Figure IV.19: Implementation of an OpenNebula Virtual Router

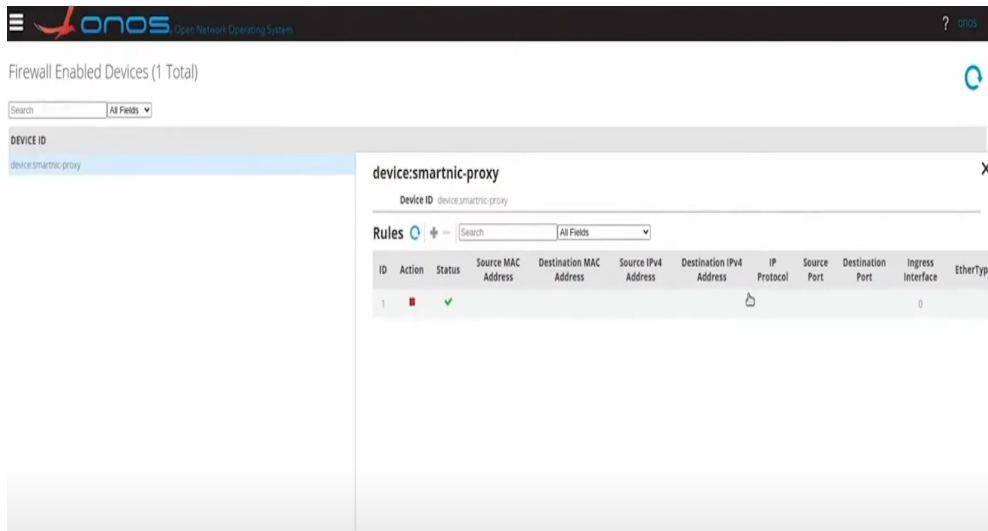


Figure IV.20: Implementation of the Proxy[20]

IV.5 Conclusion

Through this chapter, we evaluate the performance of the solution already integrated by performing the tests of isolation between the virtual machines, and by manipulating some commands via both ONOS GUI and REST API. Then, we highlight the problems faced during this project, and its solutions. Finally, some requirements are proposed for well executing the solution on the production environment.

General Conclusion

In this end-of-study project, the engineering methodologies, and theoretical aspects that we acquired throughout our Networks and Telecommunications studies were applied to solve a real-world technical problem in Icosnet's datacenter, the Algerian cloud computing service provider.

Cloud computing and virtualization are two cutting-edge technologies that continue to develop in profound and unprecedented ways due to advances in technology and hardware costs. As part of its strategy to remain at the forefront of current developments, Icosnet has decided to develop its cloud platform by integrating a software-defined networking solution. Our project aims to respond to this need.

In the dissertation, we took a closer look at the different types of network virtualization, its fundamental concepts, and tools. Next, we explained in more detail how to manage and orchestrate a virtual network using cloud tools. Next, we analyzed Icosnet's cloud in-depth, with a focus on the kernel-based virtual machine monitor (KVM) and the OpenNebula public cloud, both used in its data center. A prototype test was then created to evaluate the specific area of the landscape under study.

The concept and technologies of software defined networking using the OpenFlow protocol to centralize and control virtual network flows are then discussed, in order of preparing a needs study and the choice of a coherent and suitable SDN solution. This study led to the integration of the entire Open vSwitch virtual switch block and Open Network Operating System SDN controller into the prototype.

Finally, in addition to centralizing, controlling, and simplifying network administration using the ONOS API, the specifications required by the company have been tested, in particular the isolation between virtual machines. In this way, we were able to implement control of the second layer of the computer network OSI model.

Several challenges were encountered throughout the project. In particular, we encountered problems with the nested virtual environment used for testing, which resulted in certain functionalities being disabled. In addition, we encountered difficulties due to the lack of documentation and similar projects, as well as security issues when connecting a controller to another machine within Icosnet's local network.

This project represents exclusive use of the OpenNebula cloud and KVM hypervisor for

the first time, adding significant value for the company given the high cost of deploying such solutions.

For future work, we plan to control the third layer of the OSI model of the computer network to manage routing protocols, and network address translation, and eliminate the need for physical hardware by integrating a virtual router on OpenNebula. Furthermore, security can be controlled by creating a virtual proxy and controlled using ONOS.

Appendices

Appendix A

Complementary Informations

A.1 Snapshot

A virtual machine snapshot is a copy of a virtual machine's state and data at a certain moment. It comprises the VM's network interface cards, discs, RAM, and power state. A snapshot can be used to duplicate the same virtual machine or restore it to a previous state. Snapshots are helpful for testing, VM migration, and backup and restore procedures.

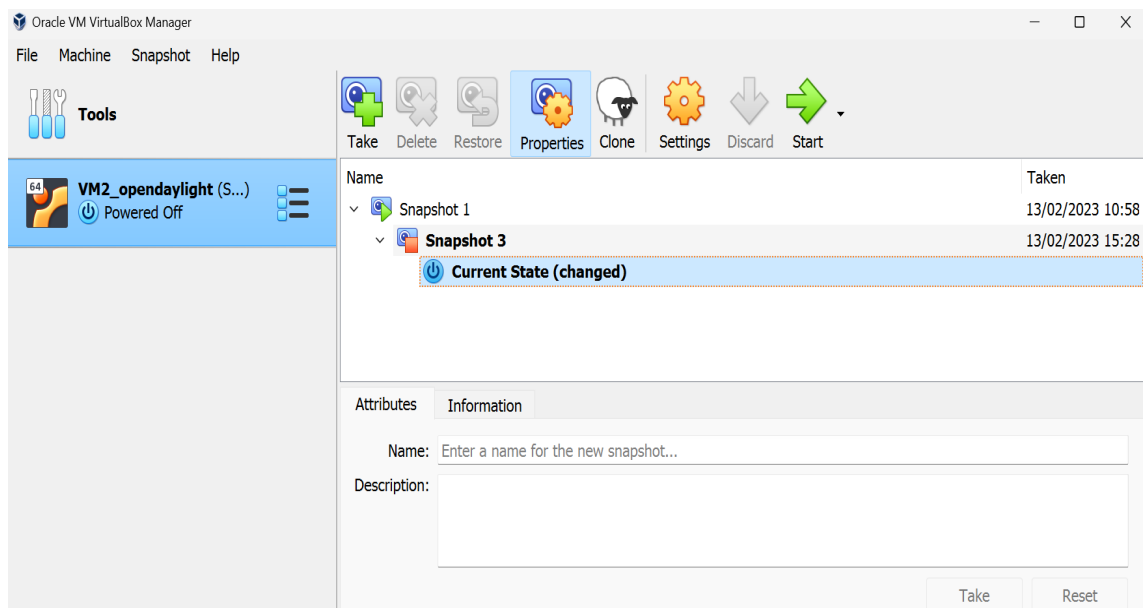


Figure A.1: VirtualBox's Snapshot

A.2 Creating VM Using Virsh Commands or virt-manager:

To create On KVM, it is possible to use both virt-manager or libvirt API via virsh commands:

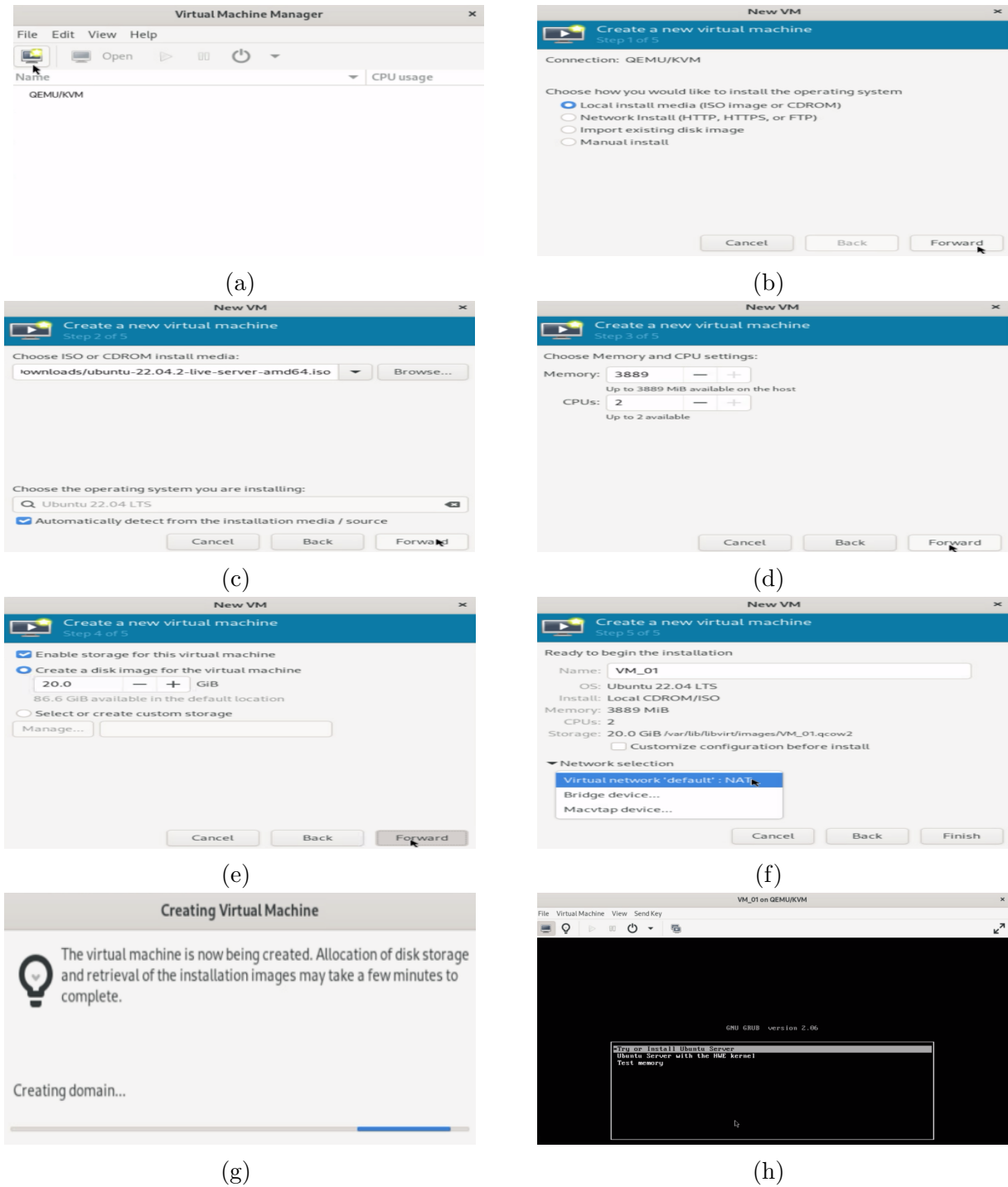


Figure A.2: Creation of a Virtual machine Using virt-manager

- **Using virt-manager:** We follow these steps

Then, we complete the rest of the installation of the operating system.

- **Using libvirt API:** We use theses commands to create an Ubuntu 22.04 LTS machine:

```
$ sudo qemu-img create -f qcow2 VM_KVM 20G
$ sudo apt install virtinst
$ sudo virt-install --name VM_KVM
```

```
--os-type=Linux
--os-variant=ubuntu22.04
--vcpu=4
--ram=4096
--disk path=/home/user/VM_KVM.img,size=20
--graphics spice
--cdrom=/home/user/Downloads/ubuntu-22.04.1-live-server-amd64.iso
--network bridge:virbr0
```

Then, we complete the rest of the installation of the operating system via Virtual Network Computing (VNC) access.

A.3 Switch OS server - desktop

Using these commands:

```
$ sudo apt install tasksel
```

```
$ sudo tasksel
```

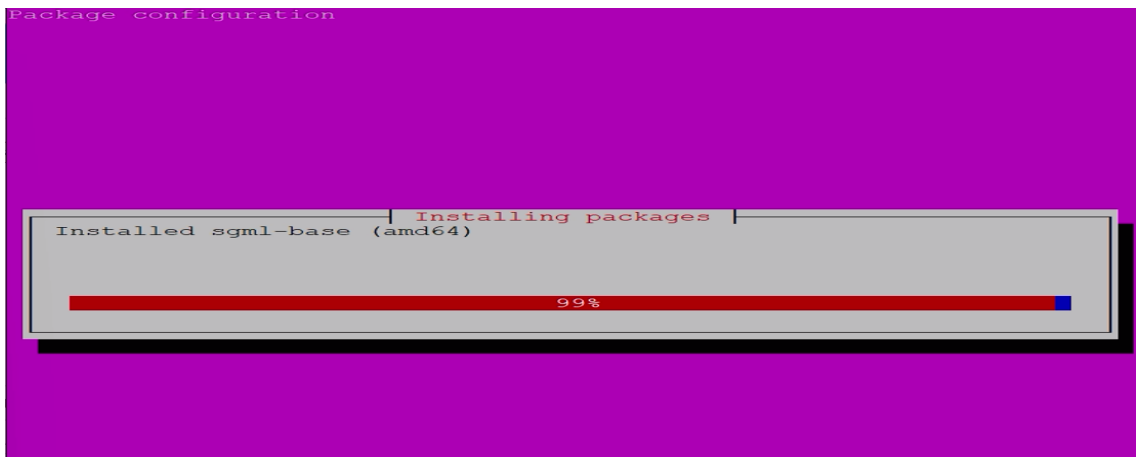


Figure A.3: Switching OS to Desktop

```
$ adduser name_of_user
```



Figure A.4: OS Desktop

A.4 OpenNebula installation

Firstly, we set up MySQL/MariaDB Back-end database, using these commands:

```
$ sudo apt update
$ sudo apt -y install mariadb-server
$ sudo mysql_secure_installation
$ sudo mysql -u root -p
$ MariaDB [(none)]> CREATE USER 'oneadmin' IDENTIFIED BY 'oneadmin';
$ MariaDB [(none)]> GRANT ALL PRIVILEGES ON opennebula.* TO 'oneadmin';
$ MariaDB [(none)]> SET GLOBAL TRANSACTION ISOLATION LEVEL READ COMMITTED

root@LTSICOH2004:~# sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 32
Server version: 10.6.12-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> CREATE USER 'oneadmin' IDENTIFIED BY 'oneadmin';
Query OK, 0 rows affected (0.003 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON opennebula.* TO 'oneadmin';
Query OK, 0 rows affected (0.002 sec)
MariaDB [(none)]> SET GLOBAL TRANSACTION ISOLATION LEVEL READ COMMITTED;
Query OK, 0 rows affected (0.000 sec)
MariaDB [(none)]> exit
Bye
```

Figure A.5: MariaDB Back-end

Then, we configure the packaging tools on the Front-end host to include OpenNebula repositories, using these commands:

```
$ sudo apt-get update
$ sudo apt-get -y install gnupg wget apt-transport-https
$ wget -q -O- https://downloads.opennebula.io/repo/repo2.key gpg --dearmor |
  > /etc/apt/trusted.gpg.d/opennebula.gpg
$ echo "deb https://downloads.opennebula.io/repo/6.6/Ubuntu/22.04 stable
  opennebula" > /etc/apt/sources.list.d/opennebula.list
$ sudo apt-get update
$ apt-get -y install opennebula opennebula-sunstone opennebula-fireedge
  opennebula-gate opennebula-flow opennebula-provision
$ sudo nano /etc/one/oned.conf

DB = [ BACKEND = "mysql",
        SERVER  = "localhost",
        PORT    = 0,
        USER    = "oneadmin",
        PASSWD  = "oneadmin",
```

```
DB_NAME = "opennebula",  
CONNECTIONS = 25,  
COMPARE_BINARY = "no" ]
```

After that, we install a complete OpenNebula Front-end from binary packages available in the software repositories already configured:

```
$ sudo -u oneadmin /bin/sh  
$ echo 'oneadmin:oneadmin' > /var/lib/one/.one/one_auth  
$ exit  
$ sudo nano /etc/one/sunstone-server.conf
```

```
:public_fireedge_endpoint: http:// @_IP_Public
```

```
$ systemctl start opennebula opennebula-sunstone opennebula-fireedge  
opennebula-gate opennebula-flow  
$ systemctl enable opennebula opennebula-sunstone opennebula-fireedge  
opennebula-gate opennebula-flow  
$ oneuser show
```

We log in through the Sunstone GUI via http://<frontend_address>:9869



Figure A.6: OpenNebula Sunstone

```
After that, we configure OpenNebula KVM Node from the binary packages, using  
these commands: $ apt-get -y install gnupg wget apt-transport-https  
$ wget -q -O- https://downloads.opennebula.io/repo/repo2.key gpg --dearmor  
> /etc/apt/trusted.gpg.d/opennebula.gpg  
$ echo "deb https://downloads.opennebula.io/repo/6.6/Ubuntu/22.04 stable  
opennebula" > /etc/apt/sources.list.d/opennebula.list  
$ apt-get update  
$ apt-get -y install opennebula-node-kvm  
$ systemctl restart libvirtd
```

```
$ hostname
```

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

```
@_IP_OpenNebula_Front -end
```

```
@hostname_OpenNebula_Front -end
```

Finally, The OpenNebula Front-end connects to the hypervisor Nodes using SSH once we configure passwordless SSH:

- **Front-end:**

```
$ hostname
```

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

```
@_IP_OpenNebula_KVM-Node
```

```
@hostname_OpenNebula_KVM-Node
```

```
$ su - oneadmin
```

```
$ ssh-keyscan Hostname_OpenNebula_Front-end Hostname_OpenNebula_KVM-Node
```

```
>> /var/lib/one/.ssh/known_hosts
```

```
$ ssh-copy-id -i /var/lib/one/.ssh/id_rsa.pub Hostname_OpenNebula_KVM-Node
```

- **KVM Node:**

```
$ su
```

```
$ sudo passwd oneadmin
```

```
$$$_PASSWORD_$$$
```

- **front end:**

```
$ su - oneadmin
```

```
$ scp -p /var/lib/one/.ssh/known_hosts @hostname_OpenNebula_KVM-Node:/var/lib/one/.ssh/
```

```
$ scp -p /var/lib/one/.ssh/id_rsa @hostname_OpenNebula_KVM-Node:/var/lib/one/.ssh/
```

Finally, we can properly create a remote KVM host and deploy Virtual machines.

Bibliographies

- [1] Thomas Olzak et al. “Microsoft virtualization: master Microsoft server, desktop, application, and presentation virtualization”. In: *Syngress Publishing* (2010).
- [2] David Rule and Rogier Dittner. “The Best Damn Server Virtualization Book Period: Including Vmware; Xen; and Microsoft Virtual Server”. In: *Syngress Publishing* (2007).
- [14] Omayma Belkadi et al. “An Integration of OpenDaylight and OpenNebula for Cloud Management Improvement using SDN”. In: *IEEE* (2019).
- [15] Sreenivas Subramanian Sriram Voruganti. “Software-Defined Networking (SDN) with OpenStack”. In: *Packt Publishing* (2016).
- [19] Pankaj Berde et al. “ONOS: towards an open, distributed SDN OS”. In: *HotSDN 2014 - Proceedings of the ACM SIGCOMM 2014 Workshop on Hot Topics in Software Defined Networking* (2014).

Webographies

- [3] *The Company - Singapore Managed Cloud Hosting*. URL: <https://www.secureax.com/about-us/>. (visited on 2023).
- [4] Amazon Web Services (AWS). *What Is Virtualization? - Cloud Computing Virtualization Explained - AWS*. URL: <https://aws.amazon.com/what-is/virtualization/>. (visited on 2023).
- [5] Bimosaurus. *5 Mesin Virtualisasi Paling Populer untuk Kampus - Blog eCampuz*. Oct. 24, 2019. URL: <https://blog.ecampuz.com/5-mesin-virtualisasi-terpopuler-untuk-kampus/>. (visited on 2023).
- [6] RisingStack Engineering. *Operating System Containers vs. Application Containers*. URL: <https://blog.risingstack.com/operating-system-containers-vs-application-containers>. (visited on 2023).
- [7] VMware Business Infrastructure Virtualization: Beyond Virtual Machines Servers | VirtualizationWorks.com. URL: <https://www.virtualizationworks.com>. (visited on 2023).
- [8] Vrapolinario. *Containers vs. virtual machines*. URL: <https://learn.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm>. (visited on 2023).
- [9] *2.3.Clusters Red Hat Virtualization 4.4 | Red Hat Customer Portal*. URL: https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.4/html/administration_guide/chap-clusters. (visited on 2023).
- [10] NAKIVO. *What Is Hyper-V Virtual Machine Load Balancing?* Dec. 20, 2022. URL: <https://www.nakivo.com/blog/hyper-v-virtual-machine-load-balancing/>. (visited on 2023).
- [11] *What Is Cloud Computing? | Microsoft Azure*. URL: <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing/>. (visited on 2023).
- [12] *What is KVM?* URL: <https://www.redhat.com/en/topics/virtualization/what-is-kvm>. (visited on 2023).
- [13] *OpenNebula 6.6 Documentation — OpenNebula 6.6.2 documentation*. URL: <https://docs.opennebula.io/6.6/>. (visited on 2023).
- [16] *Home - OpenDaylight*. URL: <https://www.opendaylight.org/>. (visited on 2023).
- [17] *Open vSwitch*. URL: <http://www.openvswitch.org/>. (visited on 2023).
- [18] *ONOS - ONOS - Wiki*. URL: <https://wiki.onosproject.org/>. (visited on 2023).

- [20] *Low-level Software Development Services Company - CodiLime*. URL: <https://codilime.com>. (visited on 2023).

الهدف من مشروع نهاية الدراسة هذا هو مركزية ومراقبة إدارة الشبكة الافتراضية لمزود الخدمات السحابية Icosnet داخل مركز بياناته. من خلال دمج حل الشبكة المحدد بالبرمجيات ONOS مع المبدل الافتراضي OVS ، بين السحابة العامة OpenNebula مراقب الأجهزة الافتراضية KVM .

الكلمات المفتاحية : السحابة، الافتراضية، مراقب الأجهزة الافتراضية، الشبكات المعرفة بالبرمجيات.

Abstract

The objective of this end-of-study project is to centralize and administer the virtual network of Icosnet cloud service provider within its data center. This will be achieved by integrating the ONOS software-defined network solution with the OVS virtual switch, connecting the OpenNebula public cloud and the KVM hypervisor.

Keywords : Cloud, Virtualization, Hypervisor, Software Defined Networking.

Résumé

L'objectif de ce projet de fin d'étude est de centraliser et de contrôler l'administration du réseau virtuel du fournisseur de services cloud Icosnet au sein de son centre de données. Cela sera réalisé en intégrant la solution de réseau défini par logiciel ONOS avec le commutateur virtuel OVS, reliant le cloud public OpenNebula et l'hyperviseur KVM.

Mots clés : Nuage, Virtualisation, Hyperviseur, Réseau Défini par Logiciel.