

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN ELECTRÓNICA Y REDES DE COMUNICACIÓN



**“IMPLEMENTACIÓN DE UNA INFRAESTRUCTURA COMO SERVICIO PARA
DESARROLLO DE PRÁCTICAS SDN Y NFV.”**

**TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO
EN ELECTRÓNICA Y REDES DE COMUNICACIÓN**

AUTOR: SANTIAGO ISRAEL ESTÉVEZ ALMEIDA

DIRECTOR: Msc. CARLOS ALBERTO VASQUEZ

Ibarra - Ecuador

2022



UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA
UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DEL CONTACTO		
CÉDULA DE IDENTIDAD:	1003111067	
APELLIDOS Y NOMBRES:	Estévez Almeida Santiago Israel	
DIRECCIÓN:	Avenida 17 de Julio	
E-MAIL:	siesteveza@utn.edu.ec	
TELÉFONO FIJO:	062-593-258	TELÉFONO MÓVIL: 0997106944
DATOS DE LA OBRA		
TÍTULO:	IMPLEMENTACIÓN DE UNA INFRAESTRUCTURA COMO SERVICIO PARA DESARROLLO DE PRÁCTICAS SDN Y NFV	
AUTOR (ES):	Estévez Almeida Santiago Israel	
FECHA: DD/MM/AAAA	12/10/2022	
PROGRAMA:	<input checked="" type="checkbox"/> PREGRADO <input type="checkbox"/> POSGRADO	
TÍTULO POR EL QUE OPTA:	Ingeniera en Electrónica y Redes de Comunicación	
ASESOR/DIRECTOR:	Msc. Carlos Alberto Vasquez	

CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrollo, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es la titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros

Ibarra, a los 14 días del mes de octubre del 2022.

EL AUTOR:

A handwritten signature in blue ink, consisting of several loops and flourishes, positioned above a horizontal line.

Santiago Israel Estévez Almeida

CI: 1003111067



UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

3

MAGISTER CARLOS VÁSQUEZ, DIRECTOR DEL PRESENTE TRABAJO DE TITULACIÓN
CERTIFICA:

Que, el presente trabajo de titulación **“IMPLEMENTACIÓN DE UNA
INFRAESTRUCTURA COMO SERVICIO PARA DESARROLLO DE PRÁCTICAS
SDN Y NFV”**. Ha sido desarrollado por el señor Santiago Israel Estévez Almeida bajos mi
supervisión.

Es todo en cuanto puedo certificar en honor a la verdad.

A handwritten signature in blue ink, appearing to read "Carlos Vásquez", is written over a horizontal dotted line.

Msc. Carlos Vásquez

DIRECTOR

AGRADECIMIENTO

En primer lugar, agradecer a mis padres Rocío y Patricio, quienes a lo largo de mi vida han estado presentes con su apoyo, consejo y paciencia en cada una de las metas que me he planteado en mi vida, mismas que han hecho de mí la persona que soy ahora

A mi hermano por su consejo en apoyo en momentos en los que una guía o un enfoque diferente era necesario.

A mis profesores, quienes miraron mi estudiante en quien depositar su conocimiento y plantaron esa semilla de curiosidad e investigación, brindando guía para mi preparación profesional y desarrollo de este proyecto.

A todos mis amigos, por su amistad, apoyo y consideración en todo momento de mi etapa universitaria.

DEDICATORIA

La realización de este trabajo, está dedicado a mis padres, quienes nunca dejaron de creer en mis capacidades, brindando su apoyo, dedicación y paciencia para que cumpla mis metas y sueños, abriéndome puertas mediante mi preparación académica de valores y principios, siendo esta la mejor herencia que un padre puede dar a sus hijos.

Índice de Contenido

1. Capítulo 1. Antecedentes	1
1.1 Tema	1
1.2 Problema	1
1.3 Objetivos	2
1.3.1 Objetivo general.	2
1.3.2 Objetivos específicos.....	2
1.4 Alcance	3
1.5 Justificación.....	5
2. Capítulo 2. Preparación, Marco Teórico	7
2.1 Metodología.....	7
2.2 Redes Tradicionales.....	8
2.3 Jerarquía de red	9
2.4 Ruteo.....	10
2.5 Enrutamiento de Paquetes.	11
2.6 Planos de Control y Datos.	12
2.7 Redes SDN	14
2.7.1 El Cambio de Hardware-Defined Networks (HDN) con Software-Defined Networks (SDN)	15
2.7.2 Funciones de red.....	16
2.7.3 Principios de SDN	18
2.8 Modelos de SDN	19
2.8.1 Modelo Overlay o basado en Host.....	20
2.8.2 Modelo Inundación y No-Inundación	20
2.8.3 Modelo Simétrico y Asimétrico	21
2.9 Componentes SDN.....	21
2.9.1 Tablas de Flujo	21
2.9.2 SDN Switch	22
2.9.3 Controladora SDN	22

2.9.4	Funciones Básicas.	22
2.9.5	Protocolo OpenFlow.....	23
2.9.6	OpenVswitch (OVS)	26
2.9.7	Controladora SDN ONOS.....	27
2.10	NFV (Network Functions Virtualized)	28
	I 32	
2.10.1	implementaciones de NFV	32
2.10.2	NFV Alto Nivel.....	34
2.11	Virtualización	34
2.11.1	Importancia de la Virtualización.....	35
2.11.2	Hypervisor	36
2.11.3	Nomenclaturas de CPU.....	39
2.12	OpenStack.....	39
2.12.1	Nube Pública.....	40
2.12.2	Nube Privada	40
2.12.3	Nube Hibrida.	41
2.12.4	Componentes Openstack.....	41
2.13	Infraestructuras como Servicio.....	43
2.14	Mininet.....	45
2.14.1	Containernet.....	45
3.	CAPITULO 3 Planificación, Diseño y Implementación.....	45
3.1	Planificación y Arquitectura del Proyecto.....	46
3.1.1	Establecimiento de IAAS	47
3.2	Diseño y Dimensionamiento de arquitectura.....	57
3.2.1	Sistema Base SDN.....	57
3.2.2	Establecimiento Software en Nodos	62
3.3	Implementación de la arquitectura.....	63
3.3.1	Instalación SO Base	63
3.3.2	Implementación IAAS.....	65
3.3.3	Generación Imagen de Hosts.	72

3.3.4	Instalación Servicios y Herramientas SDN y NFV.....	75
3.4	Guía de Proceso.....	87
4.	CAPITULO IV Operación y Resultados.....	89
4.1	Funcionamiento IAAS.....	89
4.2	Pruebas de rendimiento.....	97
4.2.1	Utilización de Recursos Físicos.....	98
4.2.2	Pruebas de Red.....	101
4.3	Rendimiento Instancias Simultáneas.....	106
4.4	Rendimiento Nodos Mininet.....	108
4.5	Funciones SASS.....	114
4.5.1	Ejecución Mininet.....	115
4.5.2	Funcionamiento OVS.....	115
4.5.3	Funcionamiento ONOS.....	117
5.	CAPITULO V Guías.....	119
5.1	Guía I – Acceso y Reconocimiento del Sistema.....	119
5.1.1	Reconocimiento Interfaz Web.....	122
5.1.2	Acceso Interfaz CLI.....	126
5.2	Guía 2 Creación Flavor e Instancias.....	131
5.2.1	Creación Flavor.....	134
5.2.2	Creación Instancia.....	136
5.3	Guía 3 Desarrollo SDN/NFV Instancia.....	141
5.3.1	Acceso.....	143
5.4	Guía 4 Práctica NFV - OPNSense.....	148
5.4.1	Generación Topología.....	150
5.4.2	Integración SDN.....	152
5.5	Guía 4 Estadísticas.....	156
5.5.1	Instalación Aplicaciones necesarias.....	158
5.5.2	Revisión de estadísticas.....	159
	Conclusiones.....	162

RECOMENDACIONES162

Anexos163

ÍNDICE DE FIGURAS

Figura 1 Fases Metodología PPDIOO7

Figura 2 Arquitectura Global de redes. Fuente: (Stallings, 2016)9

Figura 3 Costos de Destino en una Arquitectura de Red. Fuente: (Stallings, 2016) ..11

Figura 4 Tablas de Enrutamiento, Fuente: (Stallings, 2016) 11

Figura 5 Planos Datos y Control. Fuente: (Pepelnjak, 2020) 13

Figura 6 Redes SDN Planos Fuente: (Stallings, 2016) 15

Figura 7 Representación Simplificada de HDN Fuente: Adaptado de (Göransson, 2017)

16

Figura 8 Red Simple HDN, Fuente: (Göransson, 2017) 17

Figura 9 Red Simple SDN, Fuente: (Göransson, 2017) 18

Figura 10 Modelo SDN Overlay Fuente: (Göransson, 2017) 20

Figura 11 Interconexión HDN y SDN fuente: (Göransson, 2017) 22

Figura 12 Arquitectura SDN Fuente: (Stallings, 2016) 23

Figura 13 Funciones plano de datos y control Controladora SDN Fuente: (Stallings,

201

6)2

4

Figura 14 Funcionalidad OpenFlow Fuente: (Stallings, 2016) 25

Figura 15 Arquitectura Switch OpenFlow Fuente: (Stallings, 2016) 26

Figura 16 Características OVS Recuperado de: (Openvswitch.org, 2021) 26

Figura 17 Estructura ONOS : Recuperado de: www.opennetworking.org 27

Figura 18 Sub-Sistema ONOS Recuperado de: www.opennetworking.org 28

Figura 19 NFV Fuente: (Stallings, 2016) 29

Figura 20 Arquitectura NFV (Zhang, 2018).....	30
Figura 21 Infraestructura Unificada. Fuente: (Michael S. Bonfim, 2018).....	32
Figura 22 Sistema NFV. Fuente: (Stallings, 2016).....	32
Figura 23 Sistema Tradicional de reenvío de paquetes Fuente: (Stallings, 2016).....	33
Figura 24 Implementación NFV Fuente: (Stallings, 2016)	33
Figura 25 Framework NFV de alto nivel. Fuente: (Stallings, 2016)	34
Figura 26 Monitor de VM Básico (VMM) Fuente: (Portnoy, 2016)	35
Figura 27 Consolidación de Servidores Fuente: (Portnoy, 2016).....	36
Figura 28 Nivel hypervisor, Fuente: (Portnoy, 2016)Hipervisor Tipo	36
Figura 29 Hypervisor Type 1, Fuente: (Portnoy, 2016).....	37
Figura 30 Hypervisor Type 2, Fuente: (Portnoy, 2016).....	37
Figura 31 Asignación de Recursos, Fuente: (Portnoy, 2016)	38
Figura 32 Estructura Openstack. Recuperado de: (Openstack, 2021)	40
Figura 33 Componentes OpenStack Recuperado de: www.openstack.org	41
Figura 34 Categorías IAS, Fuente: Microsoft.com.....	43
Figura 35 Características de IAS. Recuperado de: www.redhat.com	43
Figura 36 Arquitectura del proyecto.	46
Figura 37 Calculadora Virtualización.....	53
Figura 38 Resultados Caso 1.....	53
Figura 39 Resultados caso 2	54
Figura 40 Implementación SDN.....	57
Figura 41 Implementación Infraestructura.....	58
Figura 42 Diseño SDN de la infraestructura.	58
Figura 43 SDN/NFV Instancia virtual	59

Figura 44 Instalador Ubuntu Server.....	64
Figura 45 Instalación KVM.	64
Figura 46 Código Fuente.	66
Figura 47 Selección Branch,.....	66
Figura 48 Fichero local.conf.	67
Figura 49 Directorio Devstack.....	67
Figura 50 Instalación OpenStack,.....	68
Figura 51 Instalación Completa.	69
Figura 52 Pantalla de Login OpenStack.	70
Figura 53 Dashboard OpenStack.	70
Figura 54 Archivo Token.....	71
Figura 55 Documento Descargado.....	71
Figura 56 Ejecución fichero.....	72
Figura 57 Descarga Imagen Base	73
Figura 58 Cambio de derechos	73
Figura 59 KVM Fuente: Autor	74
Figura 60 Actualización del sistema.	75
Figura 61 Obtención Código.....	76
Figura 62 Inicialización código.	77
Figura 63 Configuración del Compilador.	77
Figura 64 Compilación Completa.	78
Figura 65 Instalación de binarios.....	78
Figura 66 Estado OVS.	79
Figura 67 Servicio OVS Activo.....	79

Figura 68 Interfaces Básicas.	79
Figura 69 Interfaces virtuales OVS.....	80
Figura 70 Ficheros ONOS,	81
Figura 71 Arranque Servicio ONOS.....	81
Figura 72 Puertos de escucha del Host.	82
Figura 73 Interfaz Web ONOS.	82
Figura 74 ONOS CLI,.....	83
Figura 75 Activación Aplicaciones ONOS,.....	84
Figura 76 Instalación Requerimientos Containernet. Fuente: Autor	85
Figura 77 Obtención Repositorio Containernet.	86
Figura 78 Compilación Containernet.	86
Figura 79 Compilación Containernet Completa.	86
Figura 80 Prueba Containernet.	87
Figura 81 Diagrama de flujo estructura del proyecto.	88
Figura 82 Monitoreo Básico OpenStack.	89
Figura 83 Infraestructura SDN Interna	90
Figura 84 Gestor Volúmenes OpenStack	92
Figura 85 Información Imagen.	92
Figura 86 Imagen HOST Instalada.	93
Figura 87 Detalles Instancia.	94
Figura 88 Características Físicas Instancia.	95
Figura 89 Conexión a red.	95
Figura 90 Selección Imágen.	96
Figura 91 Instancia Funcionando.	96

Figura 92 Escritorio Virtual.	97
Figura 93 Recursos HOST Principal.	98
Figura 94 Utilización Recursos Host Virtual.	99
Figura 95 Utilización Host físico en carga.	100
Figura 96 Histograma prueba en carga	100
Figura 97 Instancia definida como Servidor.	103
Figura 98 Instancia definida como Cliente.	103
Figura 99 Prueba Rendimiento Red entre Instancias Virtuales.	104
Figura 100 Prueba Latencia	104
Figura 101 Rendimiento Virtual-Físico Red.	105
Figura 102 Latencia Prueba	105
Figura 103 Instancias Simultaneas	107
Figura 104 Prueba I	109
Figura 105 Reporte Prueba 1 ONOS	109
Figura 106 Utilización prueba 1	110
Figura 107 Prueba I Conectividad.	110
Figura 108 Prueba I.....	111
Figura 109 Reporte Prueba 2 ONOS	111
Figura 110 Rendimiento prueba 2.....	112
Figura 111 Prueba I Conectividad.	112
Figura 112 Prueba I.....	113
Figura 113 Reporte Prueba 1 ONOS	113
Figura 114 Rendimiento prueba 3	
Figura 115 Funcionamiento mininet cli	115

Figura 116 Interfaces Virtuales mininet-ovs	116
Figura 117 Mininet GUI.	116
Figura 118 Controladora ONOS.	117
Figura 119 Transferencia Paquetes.	118
Figura 120 Ingreso Gui Web.....	121
Figura 121 Login Web.....	121
Figura 122 Acceso Administración	122
Figura 123 Selección Menú Imágenes.....	123
Figura 124 Menú Imágenes.	123
Figura 125 Acceso Menú Instancias.	124
Figura 126 Redes IAAS.	124
Figura 127 Acceso Proyectos.	125
Figura 128 Menú Flavors	126
Figura 129 Acceso SSH	127
Figura 130 Descarga RC File.....	127
Figura 131 Script RC.	128
Figura 132 imágenes CLI.	128
Figura 133 Opciones imágenes.	128
Figura 134 Lista Instancias.	129
Figura 135 Gestión Instancias.	129
Figura 136 Redes IAAS.	129
Figura 137 Proyectos.	130
Figura 138 Comandos Proyectos.	130
Figura 139 Flavors.....	130

Figura 140 Ingreso IAAS.....	134
Figura 141 Flavors.....	134
Figura 142 Asistente Flavor.....	135
Figura 143 Creación Flavor.	135
Figura 144 Creación Instancia.	136
Figura 145 Selección Flavor	137
Figura 146 Selección de Red	137
Figura 147 Creación Instancia.	138
Figura 148 Opciones Instancia.	138
Figura 149 Instancia Activa.	139
Figura 150 Ingreso Instancia	140
Figura 151 Instancia Estudiante	143
Figura 152 Interfaz Gráfica Instancia.	143
Figura 153 Inicialización OVS	144
Figura 154 Inicialización ONOS.	144
Figura 155 Interfaz ONOS.	145
Figura 156 Resultado Script.	146
Figura 157 Topología.	146
Figura 158 Comunicación Openflow	147
Figura 159 Gestión de paquetes openflow mediante la controladora.....	147
Figura 160 Secuencia Ping	148
Figura 161 Topología de Integración.....	150
Figura 162 Configuración OVS.....	151
Figura 163 Convergencia red SDN y firewall	152

Figura 164 Ping Firewall Host H3.....	153
Figura 165 red WAN del OPNSense	153
Figura 166 Red LAN OPNSense	154
Figura 167 Solicitud internet en la red SDN.....	155
Figura 168 Gestión tráfico WAN en Controladora ONOS	155
Figura 169 Respuesta de controladora.....	156
Figura 171 menú opciones ONOS	158
Figura 172 Activación Aplicación ONOS	159
Figura 173 Características Switch	160
Figura 174 Estadísticas Switch y Flujos.	160
Figura 175 Negociación paquetes hacia destino.	161
Figura 166 Boot Instalador Ubuntu Server.....	163
Figura 167 Selección de Idioma.	164
Figura 168 Selección de Instalador.....	164
Figura 169 Selección Teclado.....	165
Figura 170 Configuración Red.....	166
Figura 171 Configuración Manual Ip.	166
Figura 172 Aplicación Configuración.	167
Figura 173 Configuración Proxi. Fuente: Autor	167
Figura 174 Repositorio Linux.....	168
Figura 175 Configuración Disco.....	168
Figura 176 Particiones Disco.....	169
Figura 177 Confirmación Cambios Disco,	169
Figura 178 Configuración Usuario.	170

Figura 179 Servicio SSH.	170
Figura 180 Características Extra.....	171
Figura 181 Instalación Servidor.....	172
Figura 182 Sistema Instalado.....	173
Figura 183 Boot Terminal.....	173
Figura 184 Interfaz Gráfica.....	174
Figura 185 KVM Fuente: Autor	175
Figura 186 Selección Idioma SO Base	176
Figura 187 Ubicación.....	176
Figura 188 Hostname.....	177
Figura 189 Nombre Cuenta. Fuente: Autor,	177
Figura 190 Nombre usuario.	178
Figura 191 Contraseña usuario.	178
Figura 192 Configuración Disco.....	179
Figura 193 Disco Virtual.	179
Figura 194 Particiones.	180
Figura 195 Actualizaciones.....	180
Figura 196 Características.....	181
Figura 197 Progreso instalación.....	181
Figura 198 Instalación Grub.	182
Figura 199 Instalación Completa.	182
Figura 200 Pantalla Login.....	183
Figura 201 Escritorio Sistema.....	183

Tablas

Tabla 1 Principios de SDN	18
Tabla 2 Componentes Openstack.....	42
Tabla 3 Ventajas y Desventajas	44
Tabla 4 Requerimientos HW/SW	47
Tabla 5 Cálculo almacenamiento Infraestructura.	48
Tabla 6 Requerimientos considerados para la arquitectura	55
Tabla 7 Comparación de infraestructura de HW	56
Tabla 8 Benchmark CPU	56
Tabla 9 Detalle Hipervisor.....	60
Tabla 10 Selección Sistema Operativo	62
Tabla 11 Características imagen de instancia	91
Tabla 12 Utilización Recursos Host Principal.....	98
Tabla 13 Utilización Recursos Host Virtual.	99
Tabla 14 Utilización Host físico en Carga.....	101
Tabla 15 Tabla Latencia.....	101
Tabla 16 Tabla Retardo.....	102
Tabla 17 Valores referenciales para pruebas de red	103
Tabla 18 Pruebas de Red.....	106
Tabla 19 Rendimiento 5 Instancias.....	107
Tabla 20 10 Instancias Simultaneas.....	107
Tabla 21 Utilización recursos host SDN.....	114

RESUMEN

El presente trabajo expone el desarrollo de una infraestructura como servicio, la cual está basada en tecnologías de computación en la nube utilizando OpenStack, la misma tiene como propósito, brindar a los estudiantes de la carrera de Telecomunicaciones de la Universidad Técnica del Norte un sistema de instancias virtuales para el desarrollo de prácticas en tecnologías SDN y NFV.

En el análisis situacional actual, mostro la existencia de problemas al momento que los estudiantes realizan prácticas o laboratorios, debido a los altos requerimientos que presentan los softwares a utilizar al simular ambientes de tecnologías SDN y NFV. Por este motivo el presente proyecto presenta una solución donde la carga de procesamiento es relevada hacia un servidor externo, mismo que presta instancias virtuales, con todas las herramientas y servicios ya instalados, de manera que el estudiante pueda realizar sus actividades, de manera remota inmediatamente.

Los resultados obtenidos en el proyecto indican que su aplicación en la carrera de Telecomunicaciones facilitará el aprendizaje y desarrollo de las tecnologías SDN y NFV, debido a su conveniencia de implementación eliminando la necesidad de la instalación de todos los servicios y aplicaciones requeridas en el dispositivo personal, ya que la instancia virtual presenta las mismas herramientas y servicios ya instalados y listos para utilizar bajo demanda del estudiante.

ABSTRACT

This work exposes the development of an infrastructure as a service, which is based on cloud computing technologies using OpenStack, the purpose of which is to provide students of the Telecommunications career of the Universidad Técnica del Norte a system of virtual instances for the development of practices in SDN and NFV technologies.

In the current situational analysis, it showed the existence of problems when students perform practices or laboratories, due to the high requirements of the software to be used when simulating SDN and NFV technology environments. For this reason, this project presents a solution where the processing load is relieved to an external server, which provides virtual instances, with all the tools and services already installed, so that the student can perform their activities remotely immediately.

The results obtained in the project indicate that its application in the Telecommunications career will facilitate the learning and development of SDN and NFV technologies, due to its convenience of implementation, eliminating the need to install all the services and applications required in the personal device, since the virtual instance presents the same tools and services already installed and ready to use on demand by the student.

Capítulo 1. Antecedentes

Este capítulo presenta todos los antecedentes y justificación sobre el desarrollo del proyecto, describiendo la problemática que el proyecto espera poder solucionar.

1.1 Tema

IMPLEMENTACIÓN DE UNA INFRAESTRUCTURA COMO SERVICIO PARA DESARROLLO DE PRÁCTICAS SDN Y NFV.

1.2 Problema

El mundo de las telecomunicaciones siempre ha sido un campo en el que el desarrollo tecnológico es un factor muy importante, permitiendo siempre optimizar los servicios que se están ofreciendo mediante estas redes, además que significa una mayor eficiencia en la transmisión de datos. Es por esto por lo que los ingenieros en redes deben estar capacitados para poder desenvolverse en estas nuevas tecnologías.

Una de las tecnologías que está tomando fuerza en este ámbito de las redes son las Redes definidas por software (SDN) que son cada vez más implementadas en arquitecturas de transmisión de datos y nubes de servicios, debido a sus ventajas en la gestión de redes y datos dentro de las redes virtualizadas, estableciendo protocolos de transmisión y comunicación entre sus nodos (**Paul Göransson,2017**).

De la misma manera otra tecnología es la Virtualización de Funciones de Red (NFV), es una tecnología que está siendo utilizada para la implementación de servicios dentro de redes Tanto tradicionales como SDN, de esta manera optimizando los recursos de un mismo servidor físico, permitiendo una gestión de red dentro del propio servidor, optimizando el manejo y transmisión de datos (Stradling, 2019).

De esta manera, los requerimientos para un perfil profesional están cambiando. Por lo que la Universidad Técnica del Norte, se ha visto en la necesidad de modificar los currículos de las asignaturas actuales y generar nuevas asignaturas, para que los futuros ingenieros en Telecomunicaciones tengan las competencias mínimas en las tecnologías SDN y NFV

Las prácticas sobre estas tecnologías deben desarrollarse en ambientes controlados que permitan un desarrollo e investigación continua sobre las temáticas puntuales indicadas en clase con prácticas enfocadas a reforzar estos conocimientos.

Este proyecto presenta una solución a este problema mediante la implementación de una infraestructura como servicio basado en SDN y NFV en la universidad, Permitiendo a la carrera de telecomunicaciones formar ingenieros preparados y capacitados en las nuevas tecnologías que encontrarán en su campo laboral.

1.3Objetivos

1.3.1 Objetivo general.

Desplegar una infraestructura como servicio SDN utilizando herramientas de virtualización y computación en la nube para la realización de prácticas SDN/NFV en redes virtualizadas para la carrera de Telecomunicaciones.

1.3.2 Objetivos específicos.

- Realizar un estudio del estado del arte en las tecnologías referentes a infraestructuras como servicio, tecnologías de computación en la nube y tecnología SDN y NFV.
- Establecer el diseño de la infraestructura como servicio basado en los requerimientos de hardware y software, permitiendo la generación de nodos de comunicación para la realización de prácticas acorde al contenido de la materia SDR.
- Desplegar una infraestructura como servicio basándose en la metodología PPDIIOO, estableciendo los pasos a seguir en cada fase de su desarrollo, cumpliendo con todos los objetivos planteados del servicio de nodos de redes para la realización de las prácticas de redes.
- Realizar guías que permitan a los estudiantes poder realizar prácticas sobre temáticas indicadas en el sílabo de la materia, permitiendo la comprensión de conceptos y generar un criterio de operatividad sobre las tecnologías SDN y NFV.

1.4 Alcance

El presente proyecto indica el proceso para la realización de una infraestructura como servicio con el objetivo de establecer prácticas, en tecnologías SDN y NFV en la materia de SDR, para desarrollar criterios de aplicación de estas tecnologías dentro de un ambiente virtual para su aprendizaje e investigación.

Para el desarrollo del proyecto se realizará un análisis del estado del arte en infraestructuras como servicio, sus aspectos y criterios de funcionamiento. Considerando tecnologías de computación en la nube que permitan la implementación de nodos de redes, con el objetivo de generar nodos de redes con soporte de tecnologías SDN y NFV.

Paralelamente al desarrollo se utilizará la metodología PPDIOO (Preparación, Planificación, Diseño, Implementación, Operación, Optimización) Desarrollada por la empresa CISCO para implementaciones tecnológicas., donde indica los pasos a realizar para la construcción de la infraestructura como servicio.

Para la etapa de **Preparación** se establecerán las características que la infraestructura como servicio debe tener para proveer los servicios fundamentados en los objetivos de la materia de SDR.

En la etapa de **Planificación** se realiza el análisis de recursos necesarios para el desarrollo de cada una de las etapas de la infraestructura como servicio.

La etapa de **Diseño** estará especificada en la arquitectura del proyecto indicando las capas de administración de bajo nivel, donde se establece la virtualización anidada, siendo como su primer nivel de virtualización un Hypervisor de aplicación para computación en la nube que sea capaz de generar hosts virtuales con capacidades de virtualización, así como también topologías de redes SDN teniendo como sus componentes su controladora SDN.

Para la etapa de **Implementación** una vez establecidos los requerimientos de hardware y software, metodología, y procedimientos a realizar se procede a aplicar cada uno de estos recursos hacia la formación de la infraestructura como servicio. Indicando como primer paso la gestión del hardware mediante el software computación en la nube,

siendo base para el establecimiento de las redes SDN, que permitirán la generación de los nodos con las capacidades de virtualización de redes SDN y NFV.

En el despliegue de la infraestructura como servicio se tendrá acceso a un nodo de red en el cual es posible la configuración de parámetros de red, así como el establecimiento de servicios por medio de un acceso remoto entregado por el Hypervisor de nivel 1 que asignara accesos hacia estos segmentos de red, para el desarrollo de las prácticas.

En estos segmentos de red será posible la generación de topologías de red internas basadas en SDN y NFV en cada uno de los hosts permitiendo la realización de prácticas a nivel individual, o también pueden realizarse en conjunto por medio de la utilización de todo el nodo de red generado por el Hypervisor de nivel 1.

Dentro de esta infraestructura como servicio se establecerán procedimientos y prácticas para que los estudiantes que ingresen a realizar sus prácticas puedan familiarizarse con el entorno de virtualización.

Las prácticas estarán enfocadas al desarrollo de criterios académicos y de desarrollo sobre temas de aplicación revisados en la materia, que pueden ser verificables en el entorno virtual de la infraestructura. permitiendo al estudiante formarse en los aspectos operativos de este tipo de tecnologías.

Dentro de la **O**peratividad de la infraestructura se establecerá políticas de acceso y control sobre los parámetros de operación de las topologías, así como la asignación de recursos virtualizados como redes, Almacenamiento, procesamiento y memoria.

Basado en los contenidos y objetivos de la materia de SDR se generarán guías de prácticas que incluyan. (Familiarización con la plataforma y tecnologías SDN, ambientación y configuraciones básicas de equipos SDN, generación de topologías SDN, integración de Servicios NFV sobre topologías SDN)

Para la fase de **O**ptimización se establecerá métodos que permitan monitorear la operación y gestión de las topologías, su administración, optimización de recursos permitiendo un mejor desempeño.

1.5 Justificación

Dado el crecimiento de aplicaciones y servicios que requieren alta disponibilidad, es necesaria la implementación de tecnologías de gestión de red, estableciendo un mejor control y gestión del tráfico de la red, por esta razón las redes SDN se convierten en una alternativa para aplicaciones de este tipo, por sus ventajas de optimización y compatibilidad con servicios integrados mediante NFV.

Es por esto que los perfiles de los profesionales que deben estar a cargo de la implementación, gestión y mantenimiento de estas redes, debe ser personal capacitado en estas áreas con las aptitudes y criterios referentes a las tecnologías SDN y NFV, Convirtiéndose en un portafolio de habilidades que debe ser parte de los currículos que las universidades ofrecen a los estudiantes.

Debido a los grandes requerimientos de gestión que necesitan las nuevas aplicaciones de alta disponibilidad, las redes tradicionales están llegando a su límite operativo, es por esta razón que las compañías están empezando a virtualizar sus funciones de red, de la misma manera el manejo y gestión de las nuevas redes SDN permite la integración de redes y servicios de maneras más efectivas en versiones empresariales (Stradling, 2019).

La industria de la virtualización está tomando cada vez más dominio sobre la gestión de redes, debido al control que estas pueden ofrecer sobre su tráfico, gestión y seguridad de las redes. donde se han realizado investigaciones donde se verifica la seguridad y los beneficios de SDN en reemplazo de las redes tradicionales con la implementación de protocolos que permitan una mejor gestión y diversificación de los servicios, por lo que las más grandes empresas como son Cisco, DELL y HP están generando sus propias soluciones en tecnologías SDN para poder establecer un portafolio de servicios y productos disponibles en el mercado. (Cabaj,2014)

Las implementaciones más grandes que se están dando a nivel mundial es la implementación de SDN y NFV en servicios de computación en la nube donde es requerida un mayor nivel de control sobre el tráfico que viaja a través de los centros de datos, es por esto que empresas como Amazon, Google, linode, VMware, Microsoft, implementan este tipo de tecnologías. (Narcisi, 2020).

En el país existen estudios en algunas universidades del país como son la Escuela Politécnica Nacional y la Universidad Técnica de Manabí donde ha existido investigaciones sobre las aplicaciones de tecnologías SDN y NFV para reemplazar las redes tradicionales de ciertos departamentos dentro de la universidad. Dando a notar que en el país ya existe una tendencia hacia la investigación y aplicación de estas tecnologías en ambientes de producción y especialmente en ambientes de investigación Académicos. (Mordillo, 2014)(Mendoza. 2016)

Esto establece una necesidad de profesionales capacitados en estas áreas, y por consiguiente una necesidad de que las carreras afines. tomen en consideración agregar estas tecnologías s su currículo para poder satisfacer esta demanda, ya que, al ser carreras tecnológicas, estar a la par con la tecnología ofrece una ventaja a los graduados en estas áreas.

Esta infraestructura ofrecerá servicios para la práctica he investigación de aspectos y competencias necesarias para el desarrollo y gestión de tecnologías SNF y NFV con guías básicas de configuración y desarrollo.

la propuesta de este proyecto es brindar a los estudiantes y docentes de la Universidad Técnica del Norte una infraestructura como servicio donde se pueda formar y capacitar a las nuevas generaciones de ingenieros en telecomunicaciones en áreas de tecnología de punta que estarán siendo implementadas próximamente en el país.

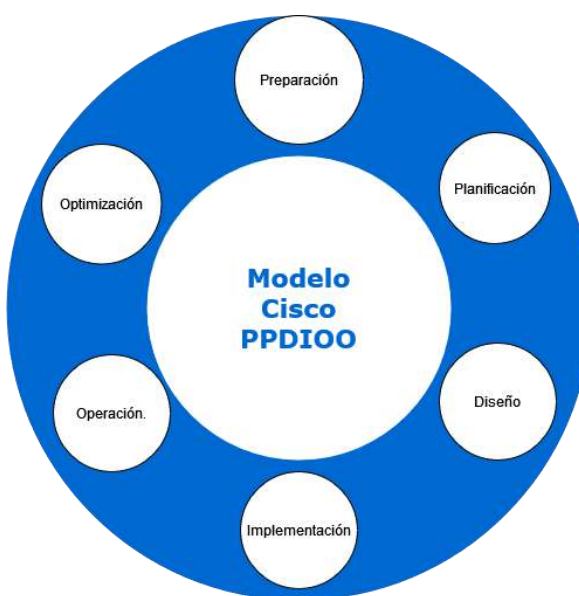
Capítulo 2. Preparación, Marco Teórico

Este proyecto será desarrollado basándose en las etapas o niveles de su metodología por lo que en esta sección de la investigación se desarrolla la fase de **Preparación**, donde se realizará la discusión de los requerimientos y procedimientos a seguir.

1.6 Metodología.

La metodología de Cisco PPDIIO propone un ciclo de vida de proyecto en el cual cada una de sus seis capas realiza una función específica y tiene relación con su capa predecesora y antecesora. Su ciclo de se presenta en la Figura 1.

Figura 1 Fases Metodología PPDIIO



El desarrollo del proyecto se abarca por completo la metodología, por este motivo se describe un concepto breve de cada una de sus fases.

Preparación. Fase la cual se realiza un estudio del estado del arte sobre las tecnologías y un estado de inicial de la problemática a solucionar con el proyecto.

Planificación. Este punto establece las consideraciones y dependencias de diseño que se requiere para la realización de proyecto.

Diseño. Etapa en la cual se desarrolla de manera comprensiva un modelo de ejecución de la idea principal del proyecto.

Implementación. Etapa en la cual se pone en práctica las consideraciones de planificación y se aplica el modelo del punto anterior, con el objetivo de materializar la idea en la solución establecida.

Operación. Se refiere a la gestión y administración después que la solución haya sido implementada.

Optimización. Acciones y medidas que pueden utilizarse en la implementación, a fin de brindar mejoras y optimizaciones en su funcionamiento.

1.7 Redes Tradicionales.

Los sistemas de redes de comunicación están a cargo de proveer servicios a los usuarios, sin importar el dispositivo de visualización o consumo, de esta manera los usuarios se conectan a estos servicios mediante múltiples métodos de conexión los cuales pueden ser canales de suscripción de líneas telefónicas, conexiones inalámbricas o por conexiones mediante fibra óptica. Facilitando de esta manera a los usuarios los accesos hacia sus servicios.

Por esta razón existen diferentes segmentos de red los cuales permiten interconectar secciones de red para obtener un acceso a los servidores de contenido, y proveedores de aplicaciones de acceso multimedia, e-mail, papers y libros. Estos servicios están generalmente localizados en un DataCenter, en el cual existen una gran cantidad de servidores interconectados por redes que permiten realizar la gestión de enrutamiento de los datos hacia los clientes. Según (Stallings, 2016) el 80% de las redes se encuentra en la interconexión de datacenter's y el 20% restante se encuentra en las infraestructuras para conectar a los usuarios finales.

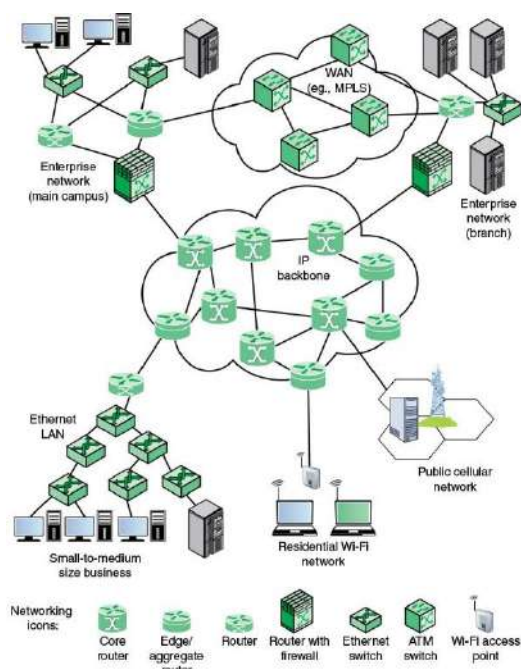
Debido a la diversidad de tipos de tecnologías, servicios, recursos e infraestructuras implementadas, cada una de estas tiene sus propios requerimientos, con

los cuales es necesario utilizar hardware o software específico para realizar la interconexión de los dispositivos.

1.8 Jerarquía de red

Esta sección se refiere a la organización de los diferentes niveles y etapas que conforman un sistema de comunicación mediante redes computacionales. La cual permite una organización de las funciones en cada una de sus capas.

Figura 2 Arquitectura Global de redes. Fuente: (Stallings, 2016)



Basado en la figura 1 se puede indicar que existe una jerarquía en el despliegue de las redes de comunicaciones de acuerdo con la jerarquía en la que se encuentran, esto ayuda a que las redes estén correctamente organizadas, Optimizando el rendimiento en cada segmento de red. Tomando en cuenta esto podemos entender que los sistemas de comunicaciones están designados en los siguientes niveles: Backbone, Distribución, Acceso

El nivel de Backbone representa las conexiones que se establecen entre infraestructuras de redes distantes entre ellas, mediante las cuales interconectan grandes áreas geográficas, permitiendo responder a la demanda de tráfico generado entre ellas.

Contando con enlaces redundantes para asegurar la conectividad entre varias zonas. Dentro de esta misma capa pueden estar alojados los servicios de acceso masivo y de alto rendimiento

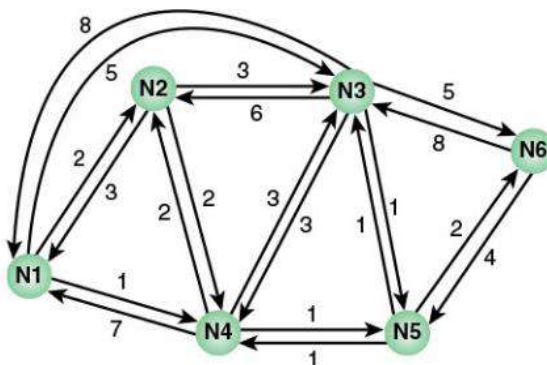
Por otra parte, en el nivel de distribución existen los routers que sirven de puerta de enlace para intercomunicar varias redes y así mismo realizar la conexión hacia el internet y otros segmentos de red, permitiendo acceder a servicios externos no alojados de manera local. Esta división se establece para que la capa de distribución alivie la demanda de recursos dentro de la capa de acceso, cabe recalcar que algunas redes de acceso y de distribución pueden ser internas y no tengan salida al internet mediante un router de borde y sea una red contenida.

De la misma manera la infraestructura de redes más cercana a los usuarios finales es Distribución, comúnmente referidas como redes LAN, estas consisten en switches que permiten la comunicación entre las redes internas y con los niveles superiores mediante Switches de capa 3, estos generalmente pertenecen a una jerarquía de mayor nivel en la organización de la red ya que está establecida únicamente para facilitar a los usuarios, acceso hacia los servicios internos de la organización. (Stallings, 2016)

1.9 Ruteo

Ruteo es la función realizada por los dispositivos routers quienes reenvían paquetes entre las redes a las cuales estos conocen tomando en cuenta una serie de parámetros para obtener el mejor camino por el cual viaja el paquete, esto se basa en un parámetro conocido como COSTO, este costo tiene varias formas de ser calculado por ejemplo dependiendo el camino a llegar a un destino en particular un camino puede ser más costoso que otro aunque sea el destino final el mismo, el trabajo del router es identificar cual es el mejor camino utilizando protocolos de enrutamiento y descubrimiento de vecinos y redes, los valores que se muestran en la figura 2 son valores de costos que se generan las tablas de enrutamiento.

Figura 3 Costos de Destino en una Arquitectura de Red. Fuente: (Stallings, 2016)



1.10 Enrutamiento de Paquetes.

Este enrutamiento de paquetes es lo que permite a los dispositivos obtener un paquete desde una de sus entradas y de acuerdo con su destino poder tomar decisiones de por qué camino de salida enviarlo, utilizando las tablas de enrutamiento de cada uno de los nodos por los cuales este paquete debe pasar, tal como se muestra en la figura 3.

Figura 4 Tablas de Enrutamiento, Fuente: (Stallings, 2016)

CENTRAL FORWARDING TABLE							
		From Node					
		1	2	3	4	5	6
To Node	1	-	1	5	2	4	5
	2	2	-	5	2	4	5
	3	4	3	-	5	3	5
	4	4	4	5	-	4	5
	5	4	4	5	5	-	5
	6	4	4	5	5	6	-

Destination	Next Node
2	2
3	4
4	4
5	4
6	4

Destination	Next Node
1	1
3	3
4	4
5	4
6	4

Destination	Next Node
1	5
2	5
4	5
5	5
6	5

Destination	Next Node
1	2
2	2
3	5
5	5
6	5

Destination	Next Node
1	4
2	4
3	3
4	4
5	6

Destination	Next Node
1	5
2	5
3	5
4	5
5	5

No se requiere que cada uno de los nodos en la arquitectura conozca el camino completo de cómo llegar al destino de un paquete en específico. Es suficiente que cada

nodo conozca las redes y nodos que tiene conexión directa, de esta manera se emplea un algoritmo que permite realizar el ruteo de los paquetes hacia su destino basados en la información de cada nodo.

De la misma manera, al establecer los costos de la comunicación esto puede llevar al punto en que cuando existe un enlace el cual está sobre cargado debido a la congestión generada al ser el enlace más idóneo para muchos paquetes simultáneamente, agregando nodos dentro de esta topología, toda la topología debe volver a realizar la asociación y el cálculo de los costos de la red lo que aumenta el trabajo de los protocolos de comunicación disminuyendo el tráfico eficaz de la red. (Stallings, 2016)

Para las redes tradicionales es común referirse a que su funcionamiento es una combinación entre el plano de control y el plano de datos en el mismo hardware. razón por la cual los dispositivos únicamente pueden trabajar en las limitaciones de sus nodos y no adquirir un control general de la parte de control y tratar los datos independientemente acorde al destino y establecer un set específico de rutas para los paquetes, esto limita la flexibilidad y características de rendimiento que se puede obtener de estos sistemas. (Stallings, 2016)

1.11 Planos de Control y Datos.

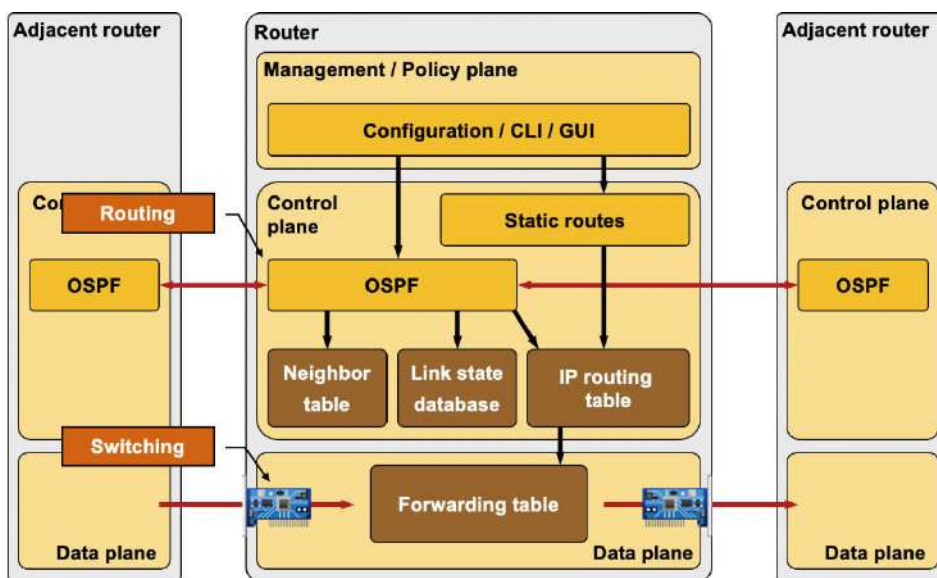
En un dispositivo de red existen 2 planos que trabajan en conjunto con el objetivo de que la red se comporte de manera predecible y cumpliendo las configuraciones deseadas, estos son los planos de datos y de control, mismos que son los responsables de tomar acciones de acuerdo al tipo de tráfico que llegue a sus interfaces.

Plano de Control. En el plano de control de enrutamiento se refiere a todas las funciones y procesos que determinan qué camino utilizar para enviar el paquete o la trama. El plano de control es responsable de poblar la tabla de enrutamiento, dibujar la topología de la red, la tabla de reenvío y por lo tanto permitir las funciones del plano de datos, aquí el router toma su decisión, en una sola línea se puede decir que es responsable de cómo se deben reenviar los paquetes.

Plano de Datos. En el plano de datos de enrutamiento se refiere a todas las funciones y procesos que reenvían paquetes/tramas de una interfaz a otra basándose en la lógica del plano de control. La tabla de enrutamiento, la tabla de reenvío y la lógica de enrutamiento constituyen la función del plano de datos, los paquetes del plano de datos atraviesan el router y la entrada y salida de tramas se realiza basándose en la lógica del plano de control, es decir, en una sola línea se puede decir que es responsable de mover los paquetes desde el origen hasta el destino; también se denomina plano de reenvío.

En una red Tradicional los 2 planos están interconectados y establecen conexiones lógicas con otros dispositivos en su mismo plano, en la figura 4, se aprecia las conexiones lógicas existentes entre ellos y los protocolos utilizados para su funcionamiento, de la misma manera podemos evidenciar que tipo de información y funciones cumple cada plano.

Figura 5 Planos Datos y Control. Fuente: (Pepelnjak, 2020)



Normalmente tenemos en mente los protocolos de enrutamiento cuando hablamos de protocolos del plano de control, pero en realidad los protocolos del plano de control realizan otras numerosas funciones, entre ellas:

- Gestión del estado de la interfaz (PPP, LACP);
- Gestión de la conectividad (BFD, CFM);
- Descubrimiento de dispositivos adyacentes (mecanismos hello presentes en la mayoría de los protocolos de enrutamiento, IS-IS, ARP, IPv6 ND, uPNP SSDP);
- Intercambio de información de topología o alcanzabilidad (protocolos de enrutamiento IP/IPv6, IS-IS en TRILL/SPB, STP);
- Provisión de servicios (RSVP para IntServ o MPLS/TE, llamadas SOAP de uPNP);

El plano de datos debería centrarse en el reenvío de paquetes, pero suele estar sobrecargado por otras actividades como:

- Creación de sesiones NAT y mantenimiento de la tabla NAT;
- Recogida de direcciones de vecinos (ejemplo: aprendizaje dinámico de direcciones MAC en el bridging, IPv6);
- Contabilidad con el protocolo Netflow¹;
- Registro de Listas de Control de Accesos (ACL);

Las funciones en el plano de datos son realizadas mediante hardware, debido a que es la parte encargada de la recepción y envío del tráfico mediante los diferentes medios de transmisión, estas son realizadas en el cpu del dispositivo.

1.12 Redes SDN

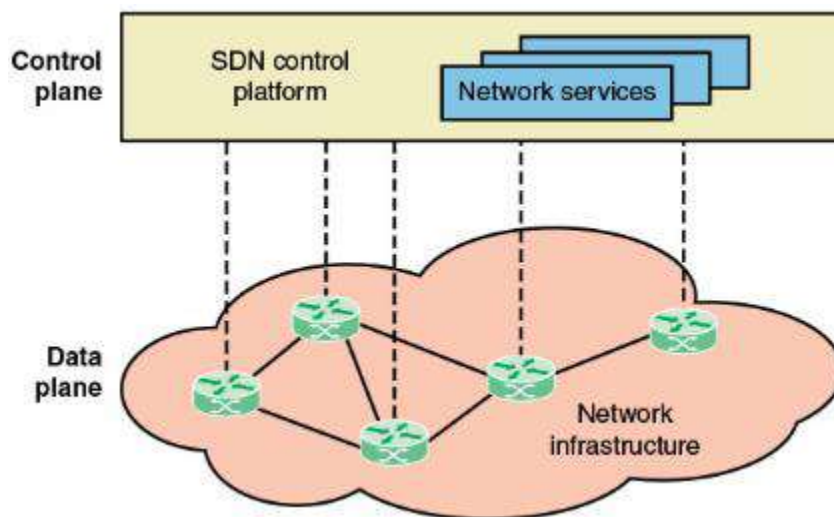
Las redes SDN son la evolución de las telecomunicaciones debido a que simplifican muchos de los problemas de redes tradicionales dividiendo el funcionamiento de la red en los planos de Control y Datos en instancias de trabajo separadas, permitiendo simplificar la carga instrucciones de cpu en los dispositivos, disminuyendo la utilización de recursos que pueden ser empleados para realizar una única tarea de forma más eficiente.

¹ Netflow es un protocolo de red creado por Cisco que recoge el tráfico de red IP activo a medida que entra o sale de una interfaz. (Clavel, 2018)

Una ventaja de separar los planos de datos y control es su administración y configuración centralizada de la red, permitiendo una escalabilidad mayor y su implementación, requiere menor número de recursos en comparación a redes tradicionales, esta separación puede ser apreciada en la figura 5.

El plano de control determina la ruta que el tráfico generado debe tomar para su destino, considerando diversas condiciones de QoS y prioridades asignadas, mientras que el plano de datos realiza el envío de los paquetes, todo esto siguiendo las indicaciones generadas por la controladora SDN, la cual proporcionará las configuraciones y políticas necesarias en cada uno de los dispositivos de red, que están bajo su control. (Stallings, 2016)

Figura 6 Redes SDN Planos Fuente: (Stallings, 2016)

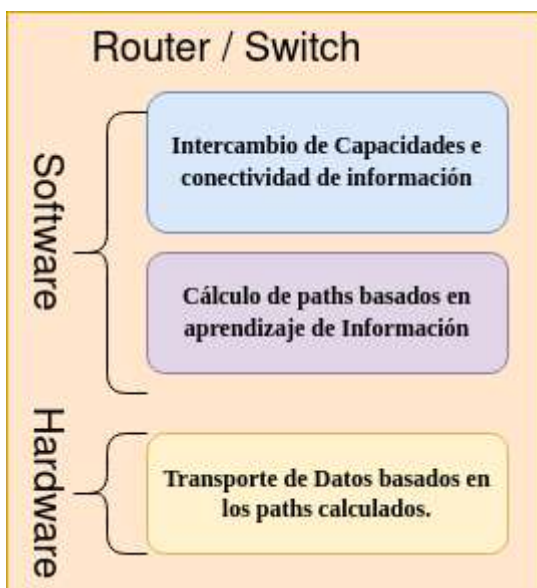


1.12.1 El Cambio de Hardware-Defined Networks (HDN) con Software-Defined Networks (SDN) .

Las redes SDN presentan considerables beneficios comparándolos a las redes tradicionales, especialmente al ser utilizadas en redes de cloud computing, esto se debe a la flexibilidad de configuración y administración que estas presentan. La división de los planos de control y datos en las redes sdn permite que mucho del procesamiento de enrutamiento se convierta responsabilidad de un único componente, en este caso la

controladora SDN, y las funciones de recepción y reenvío de datos se realiza en los dispositivos de red, esta separación de funciones puede ser apreciada en la figura 6. Una ventaja más que existe al utilizar SDN es la reducción de hardware requerido para las funciones de red, debido a la generalización de estas funciones a manera de software, de esta manera (Doherty, 2016)

Ilustración 7 Representación Simplificada de HDN Fuente: Adaptado de (Göransson, 2017)



En la migración hacia redes SDN, se ve como la evolución lógica de las redes debido a la optimización de recursos físicos, la flexibilidad de administración y operación que se puede lograr al utilizar tecnologías centralizadas en las redes SDN, incrementando las capacidades y funcionalidades de la Red.

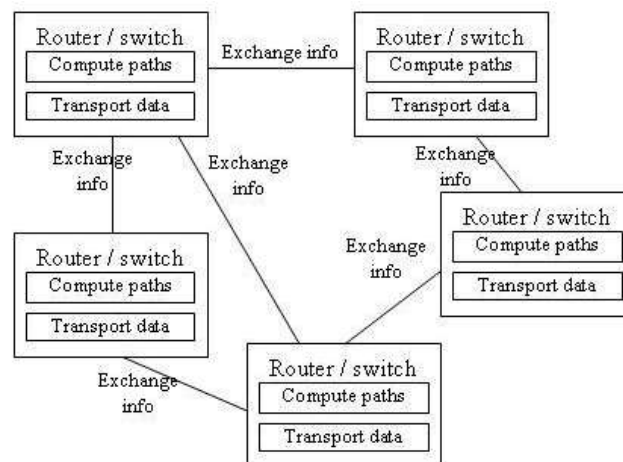
1.12.2 Funciones de red

Las funciones de red son pasos que un dispositivo debe realizar en cada nodo para poder transmitir datos, utilizando estrategias de comunicación que permiten obtener convergencia, estas funcionalidades son detalladas a continuación:

- **Intercambio de Información.** Es una primera fase de comunicación en la cual los dispositivos de red determinan el estado de sus enlaces, incluyendo las redes que son alcanzables por cada una de sus interfaces a través de un protocolo de comunicación establecido en la red.
- **Cómputo de rutas.** Obtenida la información de la red del paso anterior, se realiza un cómputo de rutas, en el cual se considera diversos parámetros para determinar la mejor ruta que tomará el tráfico hacia un destino específico, esto será realizado en cada uno de los puntos de la red generando las tablas de enrutamiento.
- **Transporte de datos.** Con la información del punto anterior los dispositivos pueden tomar decisiones basadas en estas tablas y realizar el reenvío de tráfico por el puerto asignado.

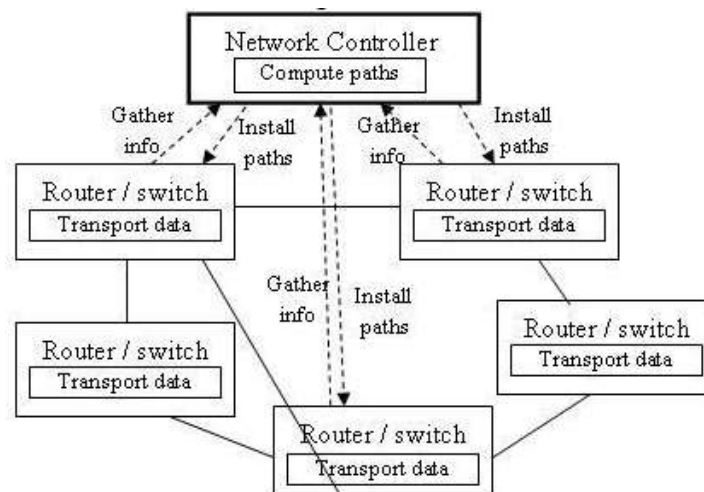
Siendo las funciones anteriormente descritas por (Göransson, 2017) pueden ser representadas tal como se muestra en la figura 7, donde se indica los componentes y el proceso que realiza en cada uno de los nodos y los enlaces correspondientes entre los dispositivos de red.

Figura 8 Red Simple HDN, Fuente: (Göransson, 2017)



Al convertir las funcionalidades de HDN hacia un modelo SDN, es necesario indicar que las funciones de cómputo de rutas e intercambio de información de transporte de datos esta dado por la controladora SDN, misma que asume las responsabilidades y a su vez envía la información requerida por cada uno de los dispositivos de red para que puedan operar acorde la controladora determine, tal y como se muestra en la figura 8 (Göransson, 2017)

Figura 9 Red Simple SDN, Fuente: (Göransson, 2017)



1.12.3 Principios de SDN

La principal característica de redes SDN es la separación del plano de control y datos, de esta manera es posible establecer una mayor supervisión y organización en la red, centralizándola en un punto que puede tomar decisiones basándose en políticas y comportamientos de la red establecidos por el administrado, de la misma manera permite la aplicación nuevas estrategias de monitoreo y análisis de la red, de acuerdo como se muestra en la tabla 1.

Tabla 1 Principios de SDN

Principios SDN	Red Tradicional	Red SDN
Planos de Control y Datos	Los planos de control y datos están localizados en los elementos de red	Plano de control está separado del de datos y este a su vez está en la controladora SDN
Inteligencia de Control	Está distribuida en cada uno de los elementos de red	Está centralizada en la controladora SDN
Programabilidad de red	La red no puede ser programada mediante aplicaciones. Cada Elemento debe ser configurado por separado.	La red puede ser programada por aplicaciones. La controladora puede exponer interfaces de red para ser manipuladas mediante aplicaciones.

Fuente: (Göransson, 2017)

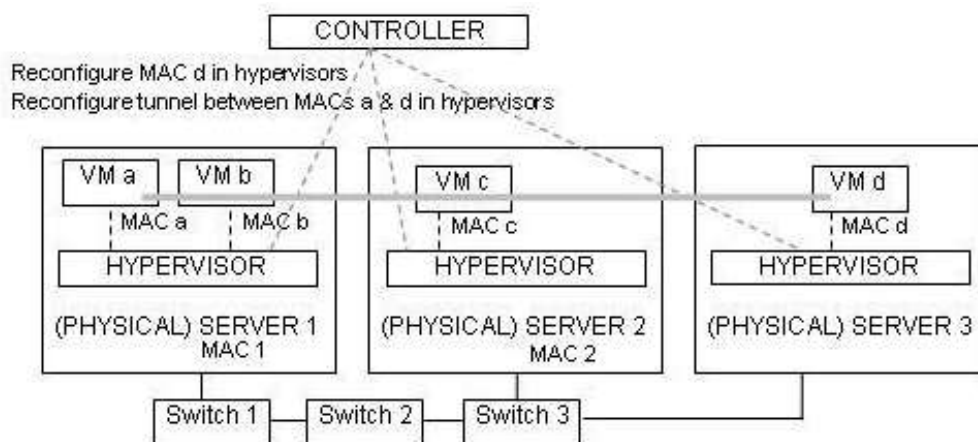
1.13 Modelos de SDN

Las redes SDN pueden ser establecidas en múltiples modelos de funcionamiento, la aplicación de cada uno de ellos está dado de acuerdo con el tipo de aplicación o ambiente donde serán desplegadas y sus requerimientos únicos. Se realiza una breve descripción de cada modelo con el objetivo analizar y entender sus diferencias.

1.13.1 Modelo Overlay o basado en Host.

El funcionamiento del modelo overlay establece identificadores para los hosts de su red, utilizando las direcciones MAC en cada uno de los hipervisor (líneas punteadas), de manera que si un host es desplazado entre ellos, no afecta las configuraciones de red establecida mediante los túneles (línea Gris) que existente entre ellos, permitiendo que la experiencia del usuario sea transparente, este modelo es soportado para máquinas virtuales (VM), host físicos y dispositivos de red como switches. Como se presenta en la figura 9

Figura 10 Modelo SDN Overlay Fuente: (Göransson, 2017)



1.13.2 Modelo Inundación y No-Inundación

En una red SDN cuando se desea agregar un nuevo dispositivo con una dirección IP y MAC, y desea comunicarse mediante la red este tiene 2 opciones, puede elegir entre enviar un paquete mediante Broadcast (Modo Inundación) donde los dispositivos cercanos lo agregaran a sus tablas de enrutamiento, la segunda manera es de Unicast (Modo no-inundación) donde el nuevo dispositivo envía un único paquete hacia la controladora de red, la cual agrega ese host a su lista y envía las configuraciones de enrutamiento a cada uno de los dispositivos de la red, de esta manera conservando ancho de banda, siendo el único costo el tiempo de cómputo de los flujos de datos que realiza la controladora.

1.13.3 Modelo Simétrico y Asimétrico

El caso Simétrico se refiere cuando la autenticación de los hosts conectados en los switches se realiza mediante la controladora de red, manteniendo una configuración centralizada donde todos los dispositivos de red realizan las funciones simples de seguridad y enrutamiento de la red.

Para el caso Asimétrico, la autenticación y las funciones simples son realizadas en los switches finales, reduciendo de esta manera la carga en la controladora permitiendo que la red pueda ser escalable para infraestructuras de mayor tamaño, incluso si superan la capacidad de la controladora, reduciendo la complejidad de la red.

1.14 Componentes SDN

En este apartado se realiza una descripción de componentes y elementos comunes y aplicables en redes SDN, indicando su relación con las redes tradicionales y su aplicación en el presente proyecto.

1.14.1 Tablas de Flujo

Las Tablas de flujo representan las decisiones que un dispositivo SDN toma a la llegada de un paquete por cualquiera de sus interfaces, utilizándolas para determinar la acción que debe tomar basándose en la información de destino propia de cada paquete, estas tablas son generadas por la controladora SDN e instaladas en cada uno de los switches o routers de la infraestructura SDN.

Para la ejecución de estas tablas se establece campos de comparación para ser referenciados al momento de recibir un paquete por alguna de sus interfaces, estas están asociadas a decisiones que permiten al dispositivo tomar una acciones al obtener un acierto en su comparación, su funcionamiento se asemeja a una implementación listas de acceso(ACL), permitiendo establecer reglas que permitan optimizar el flujo de datos. (Göransson, 2017)

1.14.2 SDN Switch

Estos dispositivos funcionan como instancias virtuales, que permiten la integración con sistemas de redes SDN en el mismo hypervisor, o de redes interconectadas, siendo su control y administración bajo la controladora SDN.

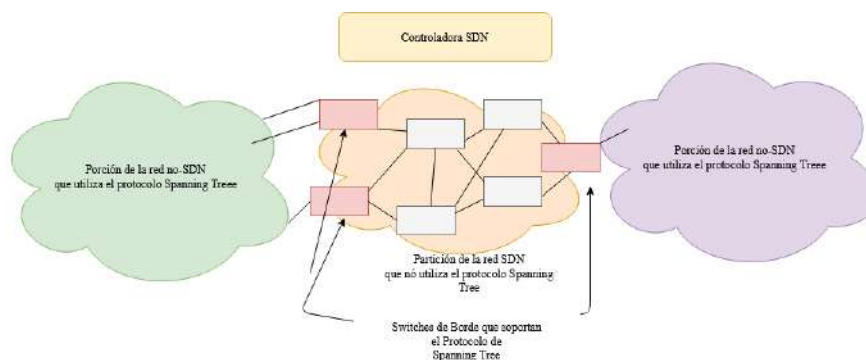
1.14.3 Controladora SDN

Su trabajo principal es el de mantener una vista y control completo sobre la red, implementar políticas de decisión, reenvío, balanceo de carga, entre otros. Controla todos los dispositivos SDN que conforman la infraestructura SDN. Permitiendo además establecer un acceso de API para aplicaciones que requieran utilizar la controladora, ya sean las aplicaciones por defecto para sus funciones básicas o implementaciones específicas de terceros. (Göransson, 2017)

1.14.4 Funciones Básicas.

Al trabajar en redes SDN los protocolos tradicionales que se utilizan para la comunicación entre distintos nodos como son IS-IS, OSPF y RIP no son requeridos ya que SDN utiliza su propio protocolo, generando un problema al conectarse a redes de salida tradicionales, para esto se requiere routers de borde que permitan la traducción con estas tecnologías, este escenario se presenta en la figura 10, donde se permite la integración de múltiples protocolos en los bordes de la red SDN.

Figura 11 Interconexión HDN y SDN fuente: (Göransson, 2017)



mediante este ejemplo indicamos los diferentes protocolos que se pueden utilizar dentro de una misma topología donde se mezclan redes tradicionales con SDN, por lo que

se debe considerar las funciones que se deben cumplir los nodos de borde para establecer una comunicación hacia redes tradicionales, a continuación, realizamos un resumen de estas.

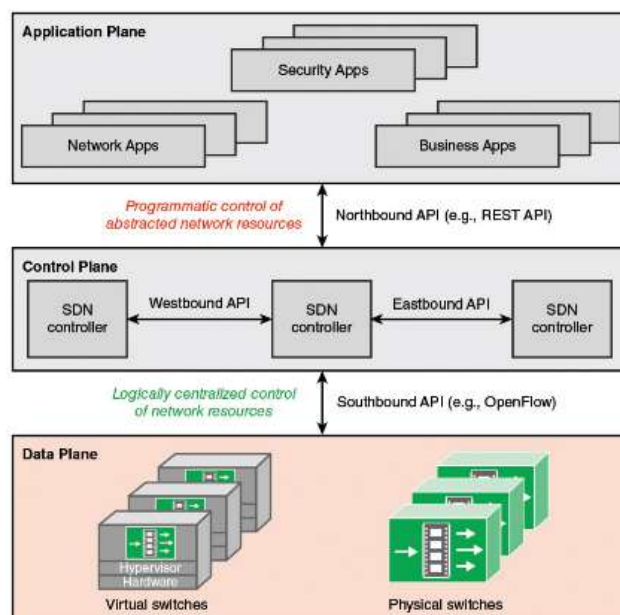
- **Discovery.** Permite a cada router encontrar y comunicarse con los vecinos y obtener todas sus capacidades y rutas
- **Keep Alive.** Es un método el cual monitorea el estado del enlace con cada uno de los routers vecinos confirmando su disponibilidad.
- **Advertisement.** Es la propagación del anuncio de las redes conocidas por ese router con sus routers vecinos.
- **Router Computation.** Se refiere a la generación y cálculo de las rutas para los diferentes flujos de paquetes entre routers para llegar a un destino final.

En este sentido, al ser SDN un sistema centralizado donde la controladora conoce cada uno de los aspectos de la red todas estas funcionalidades pasan a ser de su control, asignando las rutas de los flujos de paquetes y reglas de enrutamiento en los switches. (Göransson, 2017)

1.14.5 Protocolo OpenFLow

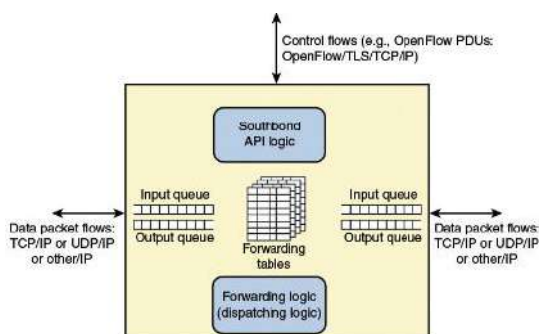
Es un protocolo del plano de datos enfocado a las operaciones lógicas que existen entre los dispositivos de red y la controladora SDN mostrados en la figura 11; donde se muestra a manera de ejemplo distintos flujos de comunicación en los cuales este protocolo es utilizado en la comunicación entre las controladoras SDN y los dispositivos de red que envían los paquetes; interconectando los planos de datos y control dentro de la infraestructura de red mediante la Interfaz de programación de aplicaciones (por sus siglas en inglés API) Southbound, permitiendo la comunicación entre múltiples instancias de cada uno de los planos centralizando todo desde la controladora SDN.

Figura 12 Arquitectura SDN Fuente: (Stallings, 2016)



De esta manera la controladora puede encargarse de la generación de las tablas de enrutamiento así como la asignación de flujos para el transporte de datos entre nodos como se muestra en la figura 13, donde puede obtener datos e información de diversos protocolos mediante sus interfaces.

Figura 13 Funciones plano de datos y control Controladora SDN Fuente:
(Stallings, 2016)



- **Soporte de funciones de control**

Es la parte encargada de la interactividad entre dispositivos del plano de control de SDN, permitiendo programabilidad de los recursos, siendo estos gestionados mediante la controladora.

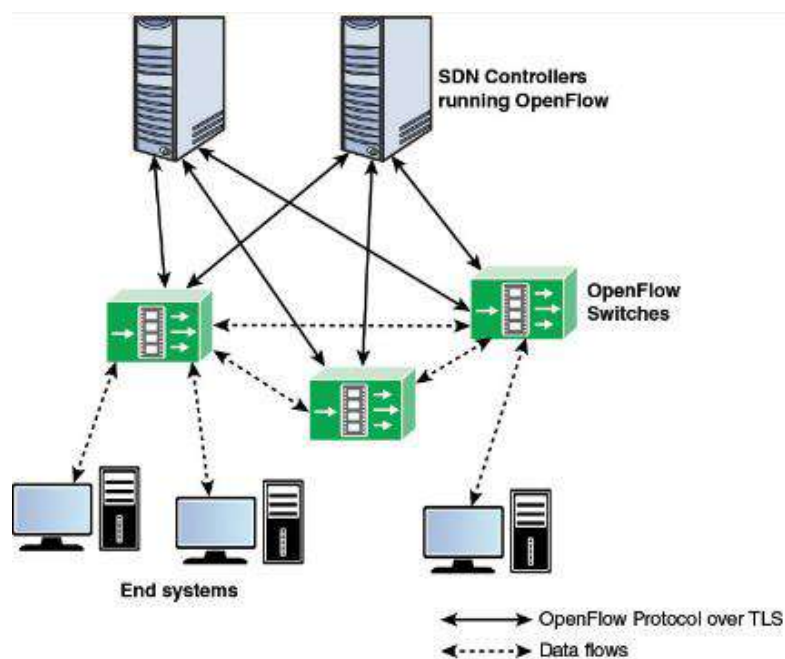
- **Función de reenvío de datos**

Esencialmente realiza funciones de un Switch al tomar información de una entrada y la reenvía por el puerto necesario cumpliendo las políticas establecidas por la controladora SDN, de la misma manera puede tomar diferentes decisiones basados en una prioridad o eliminar el paquete si no está permitido su paso. Las implementaciones de un dispositivo SDN se debe cumplir con ciertos requerimientos, permitiendo de esta manera su interoperabilidad y cumplimiento de los requisitos mínimos de los protocolos. Siendo los requerimientos los siguientes:

- Debe existir una lógica común entre todos los switches y routers satisfaciendo la conectividad con la controladora SDN
- Debe utilizar un protocolo seguro entre el dispositivo y la controladora SDN.

Todos estos requisitos están implícitos en el protocolo OpenFlow ya que es un protocolo de comunicación entre dispositivos de red y la controladora SDN, para comprender de mejor manera la implementación del protocolo se muestra en la figura 14, las conexiones de los diferentes dispositivos de red con la controladora.

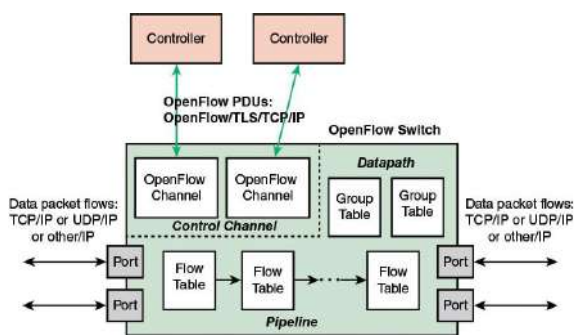
Figura 14 Funcionalidad OpenFlow Fuente: (Stallings, 2016)



Como se observa en la figura 15 las comunicaciones entre los switches OpenFlow y la controladora SDN establece un flujo de comunicación separado de los datos

transmitidos en la red. Esto es necesario para poder tener un control centralizado de la red y es un componente de los switches compatibles con el protocolo OpenFlow como se muestra en la figura 14 (Stallings, 2016), donde se aprecia las funciones básicas de un switch normal con su canal de control estableciendo comunicación con la controladora, pero sus puertos pueden transmitir todo tipo de datos.

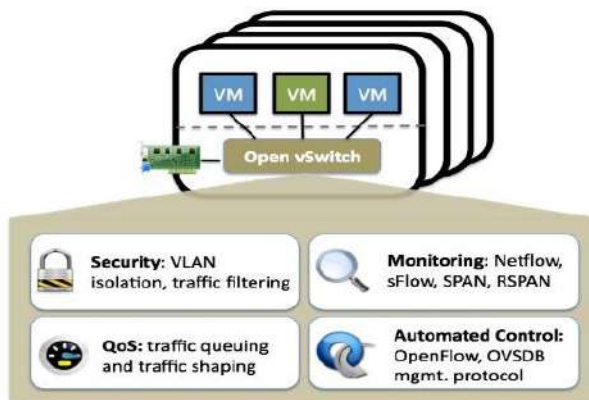
Figura 15 Arquitectura Switch OpenFlow Fuente: (Stallings, 2016)



1.14.6 OpenVswitch (OVS)

Es un Switch virtual multicapa de código abierto, que es capaz de ser desplegado en infraestructuras de producción, con soporte para protocolos NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag, entre otros, permitiendo integración con redes tradicionales y redes SDN, así como ofreciendo toda la flexibilidad de un software SDN para NFV. Ofrece las siguientes capacidades mostradas en la figura 16 (Openvswitch.org, 2021) de monitoreo, seguridad y aplicaciones de QoS.

Figura 16 Características OVS Recuperado de: (Openvswitch.org, 2021)

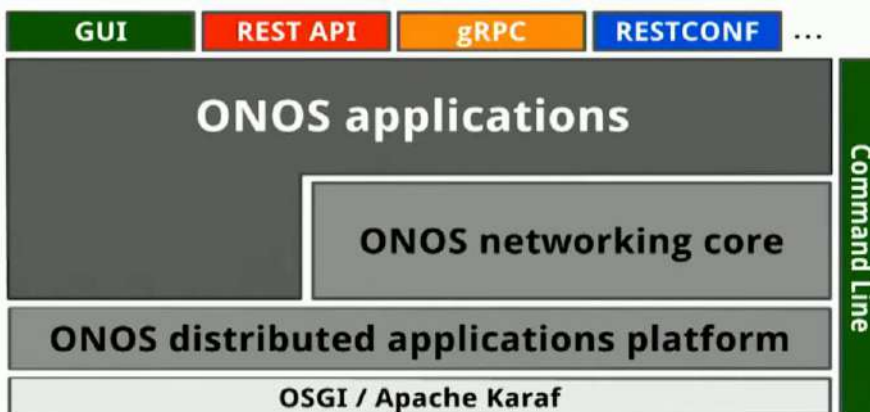


1.14.7 Controladora SDN ONOS

En el presente proyecto se considera ONOS como controladora específica para SDN, sus siglas significan **Open Network Operating System**, este software provee el plano de control para las redes SDN siendo complementada con OVS (Open Virtual Switch) para el desarrollo de redes SDN, estableciendo las comunicaciones entre servicios y host en las redes que este tiene a su cargo, así como redes vecinas, beneficiándose de la usabilidad y flexibilidad que este sistema presenta gracias a su compatibilidad de protocolos como son OpenFlow para SDN y protocolos estándar de redes tradicionales.

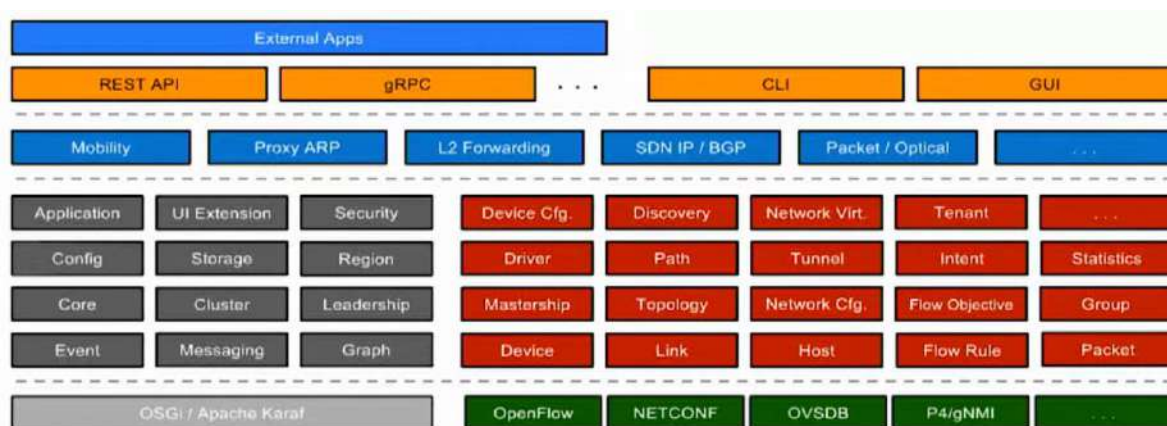
La estructura de la controladora ONOS mostrado en la figura 16 esta basada en un sistema modular java de OSGI(Open Service Gateway Initiative) sobre el cual se establece el sistema distribuido de decisiones el cual representa el core funcional de la controladora, en la siguiente capa esta la funcionalidad de la red, donde se tiene la lógica de funcionamiento de todos los protocolos soportados, conjuntamente se tiene las aplicaciones SDN las cuales pueden ser accedidas mediante la Interfaz Gráfica del usuario (GUI), así como su interfaz de programación de aplicaciones mediante un API(Application Programming Interface), el acceso a las aplicaciones de la controladora puede ser de forma remota mediante gRPC así como su configuración mediante su componente de RESTCONF, obteniendo una controladora robusta, configurable , extensible y de alta compatibilidad para aplicaciones de infraestructura como se indica en la figura 17.

Figura 17 Estructura ONOS : Recuperado de: www.opennetworking.org



Al ser la controladora ONOS de código abierto permite la integración de aplicaciones desde su repositorio o la creación e implementación de aplicaciones personales entendiendo su funcionamiento en sus subsistemas mostrados en la figura 18 separadas en capas y colores, donde la capa inferior presenta los accesos al sistema básico de la controladora, en su segunda capa se indica en color gris y rojo las funciones de administración de la controladora y de red respectivamente, y sobre esta se encuentran una capa acceso y forwarding, en su capa superior se tiene el acceso de las aplicaciones de la controladora con las entradas de API e interfaces de usuario gráficas y mediante terminal.

Figura 18 Sub-Sistema ONOS Recuperado de: www.opennetworking.org



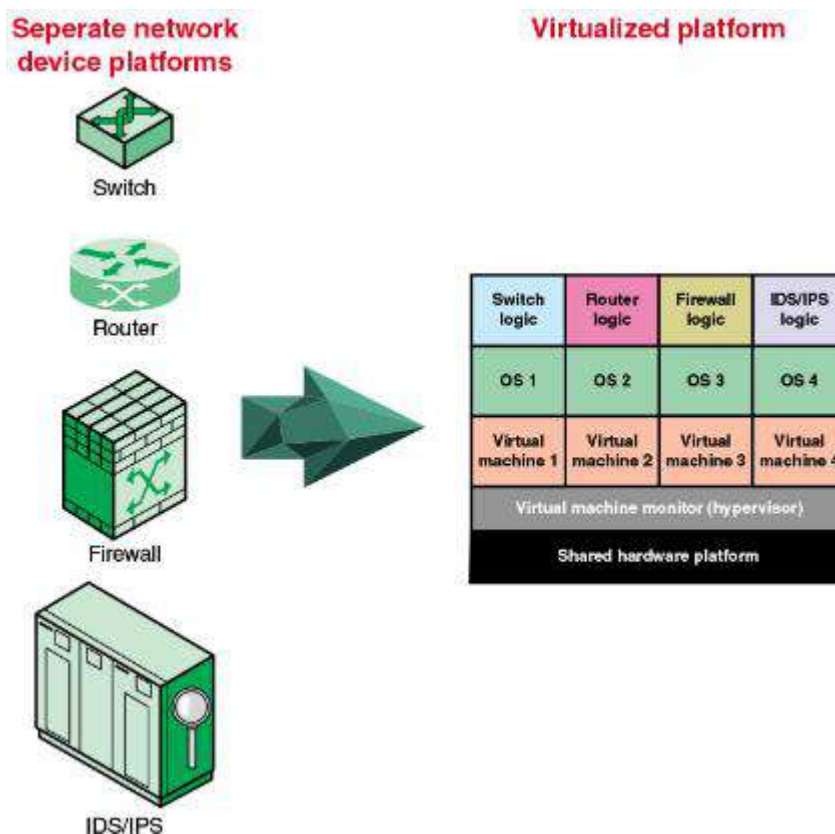
La gran cantidad de módulos de la controladora y su jerarquía permite la una gran adaptabilidad e integración en múltiples ambientes y escenarios. (opennetworking.org, 2021)

1.15 NFV (Network Functions Virtualized)

Uno de los avances en telecomunicaciones es la capacidad de obtener funcionalidades de dispositivos de red de manera virtualizada, permitiendo establecer múltiples servicios como instancias en un mismo hipervisor como se muestra en la figura 19, reduciendo de esta manera la necesidad de adquirir dispositivos físicos. Las implementaciones NFV principalmente son utilizadas en ambientes de virtualización donde se requiere mayor control de acceso hacia los hosts alojados en el hipervisor.

NFV puede ser utilizado a la par con redes SDN, esto permite que la gestión de red resultante en un hipervisor hacia cada dispositivo de red y host virtualizado sea controlado y administrado de manera centralizada, así se obtiene los beneficios de las 2 tecnologías, un ejemplo de esto son las implementaciones de computación en la nube, donde todo el control de tráfico y acceso a host se realiza mediante redes SDN que controla dispositivos NFV. (Stallings, 2016)

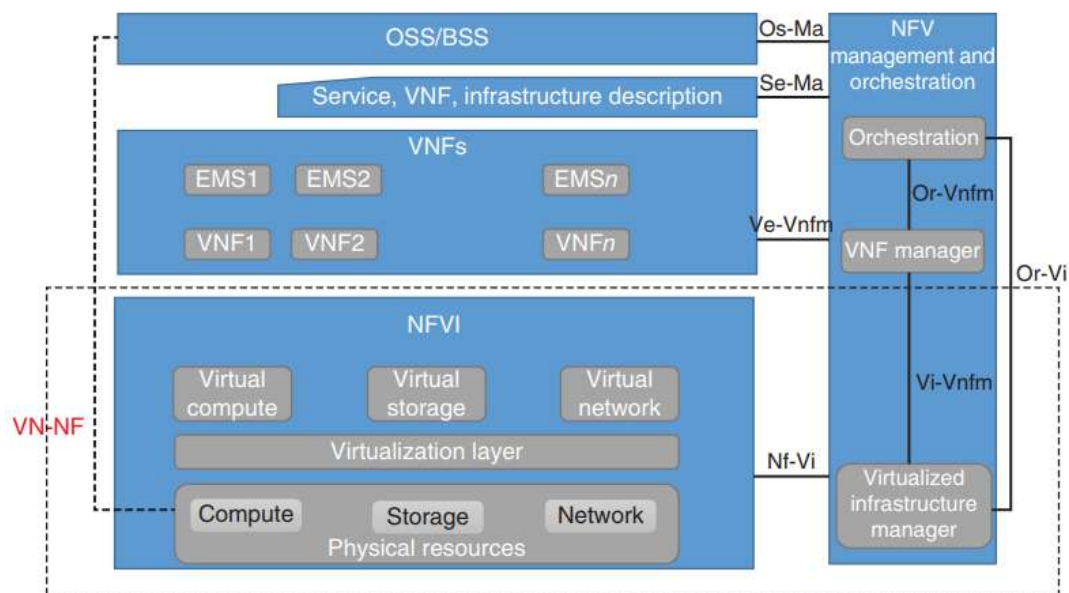
Figura 19 NFV Fuente: (Stallings, 2016)



NFV está relacionada de manera muy cercana con tecnologías como SDN y cloud computing, esta relación a permitido que implementaciones de NFV sean cada vez más comunes en el mundo de las telecomunicaciones, gracias a su flexibilidad en acceso a la red desde un punto central conocido como un punto de presencia(PoP), esto permite desplegar instancias de funciones VNF basado en demanda de la red y en locaciones requeridas, presentando características de elasticidad y de optimización de recursos.

Las funciones y despliegues de instancias VNF son realizadas mediante un sistema de orquestación, encargado de la asignación de recursos y de su funcionamiento, es posible analizarlo como un hipervisor para instancias VNF, por su modelo de capas mostrado en la figura 20 es apreciable los recursos físicos y virtualizados (a la izquierda) que serán utilizados basándose en instrucciones dadas por el orquestador de recursos (a su derecha) funcionando de manera virtual en el mismo sistema. (Zhang, 2018)

Figura 20 Arquitectura NFV (Zhang, 2018)



Con la arquitectura mostrada en la figura 19 es posible analizar sus puntos más importantes que determinan su funcionamiento, administración y características como se presenta a continuación.

NFVI (Network Functions Virtualization Infrastructure). Funciona a manera de un hipervisor el cual realiza la asignación de recursos a las instancias VNF indicados por el sistema de orquestación. Este bloque además comprende el dominio de trabajo, el cual es el responsable por proveer y extender la visibilidad, configuración, contabilidad, rendimiento y seguridad de las instancias.

VNF. Es la instancia virtual generada por NFVI, acompañado de un EMS (Sistema de control de Elementos), la cual asigna las funciones de red solicitadas por el sistema de orquestación VNF

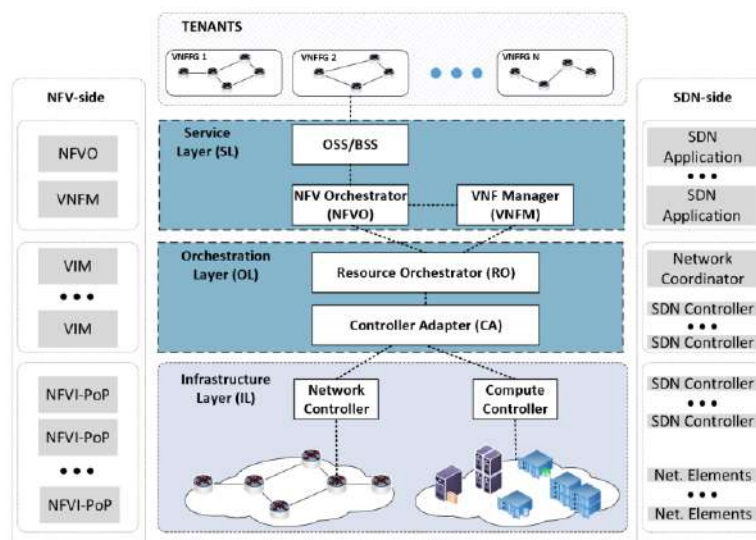
VNF M&O (Sistema de Orquestación y Control-MANO).

Se refiere a la orquestación y gestión del ciclo de vida de los recursos físicos y/o de software soportados por la virtualización de la infraestructura, y la gestión del ciclo de vida de las VNF. La gestión y orquestación de la NFV se centra en las tareas de gestión específicas de la virtualización necesarias en el marco de la NFV. La gestión y orquestación de la NFV también interactúa con el entorno OSS/BSS (externo a la NFV), lo que permite que la NFV se integre en un entorno de gestión ya existente en toda la red.

Todo el sistema NFV se rige por un conjunto de metadatos que describen los requisitos de servicio, VNF y requisitos de infraestructura, de modo que los sistemas de gestión y orquestación de la NFV puedan actuar adecuadamente. Estas descripciones, junto con los Servicios, VNFs e Infraestructura, pueden ser proporcionadas por diferentes actores de la industria. (Etsi, 2018).

La arquitectura unificada NFV/SDN tiene como objetivo crear y gestionar los servicios de red dinámicos e integrales desde las redes domésticas y empresariales hasta el centro de datos del operador, proporcionando un marco MANO que integra los dominios de la nube y la WAN e incluye tres capas: La capa de servicios (SL), la capa de orquestación (OL) y la capa de infraestructura (IL). La figura 21 muestra la visión simplificada de la arquitectura UNIFY, destacando los principales componentes funcionales de los modelos de referencia NFV del ETSI (a la izquierda) y SDN del ONF (a la derecha).

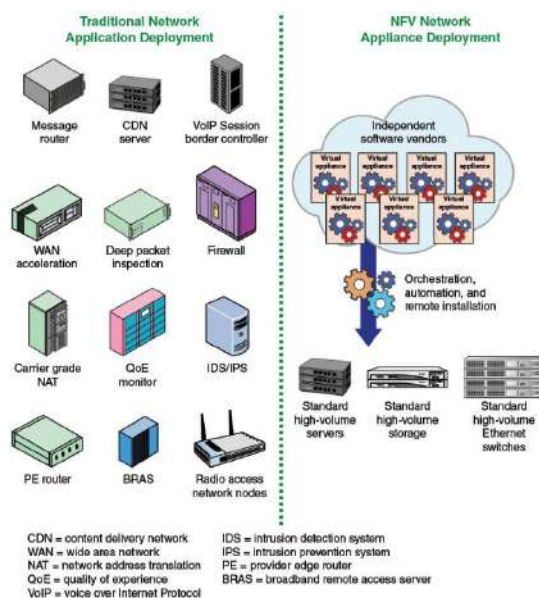
Figura 21 Infraestructura Unificada. Fuente: (Michael S. Bonfim, 2018)



1.15.1 Implementaciones de NFV

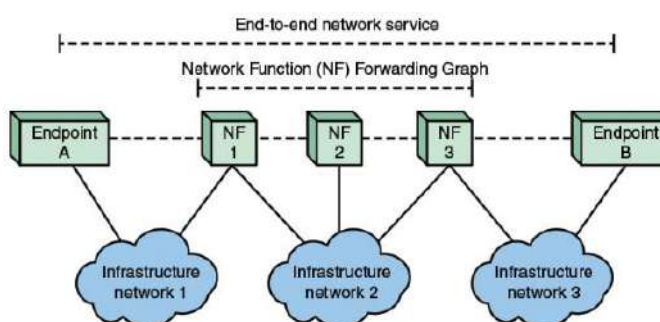
La virtualización facilita la gestión, configuración y monitoreo de las aplicaciones de red tradicionales, tal como se muestra en la Figura 21, debido a la flexibilidad que brinda esta tecnología y su escalabilidad para infraestructuras de gran tamaño, permite integrar múltiples funciones de red en un mismo hardware ejecutándose como instancias independientes.

Figura 22 Sistema NFV. Fuente: (Stallings, 2016)



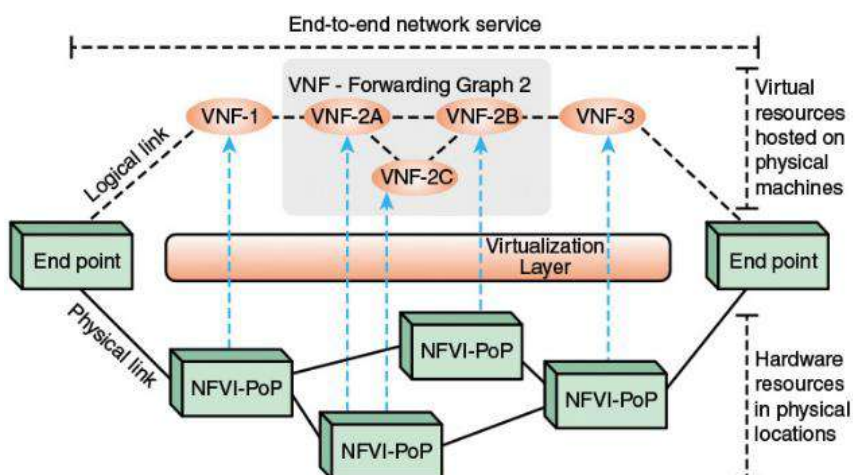
Mediante la figura 21 es evidente la reducción de recursos en aplicaciones y servicios mediante NFV, a comparación a las redes tradicionales basadas en Hardware. Como parte de la arquitectura para NFV se observa en la figura 23 y 24 la diferencia entre las implementaciones tradicional y NFV respectivamente, en la primera es apreciable los dispositivos que se utilizan entre cada uno de los nodos entre los puntos de destino y origen (en verde) así como las interconexiones existentes para cada infraestructura y red (en azul).

Figura 23 Sistema Tradicional de reenvío de paquetes Fuente: (Stallings, 2016)



En la figura 25 se muestra las etapas de transmisión que un flujo de datos debe pasar con el objetivo de llegar a su destino final, en este punto la red está en una capa de virtualización separando los recursos y enlaces físicos de las instancias VNF (Virtualized Network Function) alojadas en el hipervisor.

Figura 24 Implementación NFV Fuente: (Stallings, 2016)



1.15.2 NFV Alto Nivel

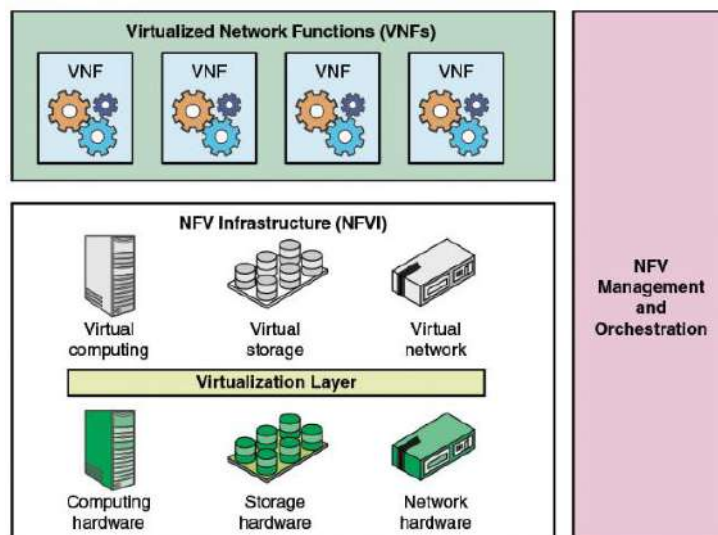


Figura 25 Framework NFV de alto nivel. Fuente: (Stallings, 2016)

De la misma manera en que un host físico puede ser utilizado para correr máquinas virtuales mediante la utilización de un hipervisor, NFV de alto nivel utiliza el mismo concepto para la implementación de instancias de VNF al asignar recursos como se muestra en la figura 24, esto recursos pueden ser computacionales, almacenamiento o red y orquestados mediante un módulo de administración central.

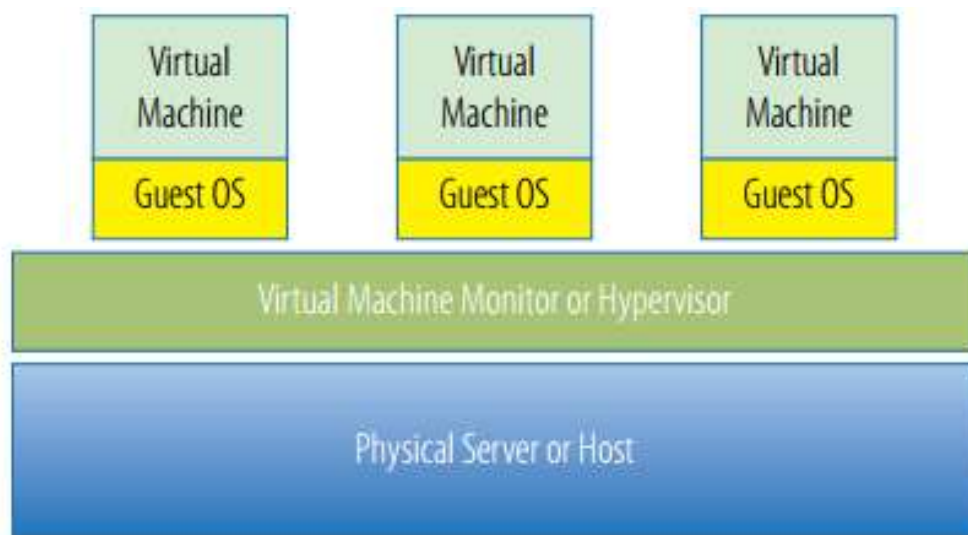
1.16 Virtualización

La virtualización es una tecnología que permite utilizar de mejor manera los recursos de hardware de un sistema, esto es posible mediante un hipervisor mostrado en la figura 24 que se establece entre el hardware los sistemas operativos que se desean utilizar. La asignación de recursos para cada una de las instancias virtualizadas se realiza mediante la abstracción a manera de software de componentes físicos como son, dispositivos de almacenamiento, de red entre otros, esto pretende simular un ambiente transparente donde el sistema operativo funcione interpretando estos dispositivos virtuales como físicos sin perder sus funcionalidades y características. Este proceso permite la utilización y aprovechamiento de todo el potencial de los sistemas de

procesamiento disponibles, reduciendo costos de adquisición de nuevos equipos para un fin específico.

La virtualización permite mayor flexibilidad y mejor disponibilidad de los recursos físicos. Para esto es necesario la utilización de un hypervisor que realice el proceso de abstracción de estos recursos, este se coloca generalmente entre los recursos físicos del computador y los hosts virtualizados como se muestra en la Figura 26

Figura 26 Monitor de VM Básico (VMM) Fuente: (Portnoy, 2016)

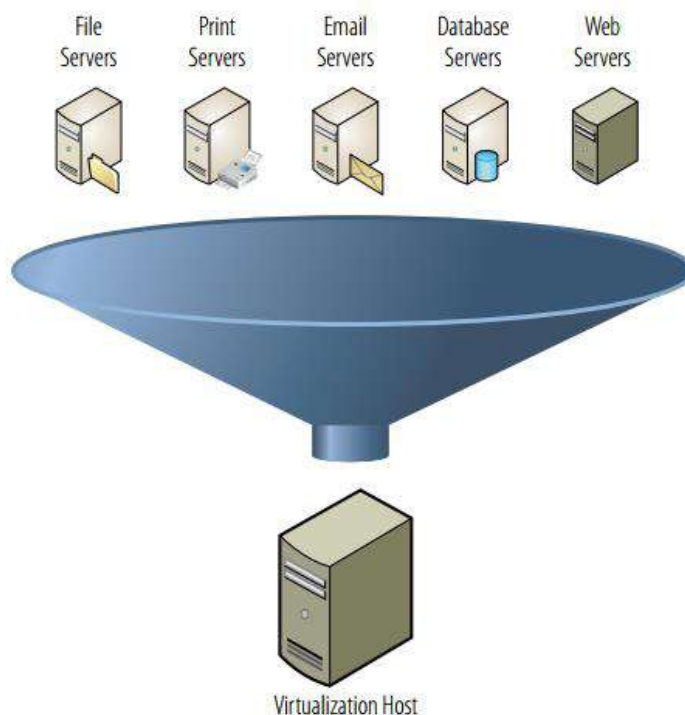


Sabiendo que los sistemas de virtualización emulan características de un dispositivo físico, es necesario tomar en cuenta que se pueden aplicar las mismas reglas de seguridad y de gestión, permitiéndolos trabajar independientemente sin compromiso de rendimiento o funcionalidad.

1.16.1 Importancia de la Virtualización

La virtualización permite desplegar servicios y aplicaciones dentro de una misma infraestructura, utilizando los recursos ya existentes en el Mainframe y así no adquirir nuevos dispositivos, obteniendo un beneficio mayor del mismo hardware, por lo que el resultado del costo-beneficio es favorable, integrando múltiples servicios en los data centers permitiendo resolver problemas críticos de disponibilidad de recursos como se muestra en la figura 27.

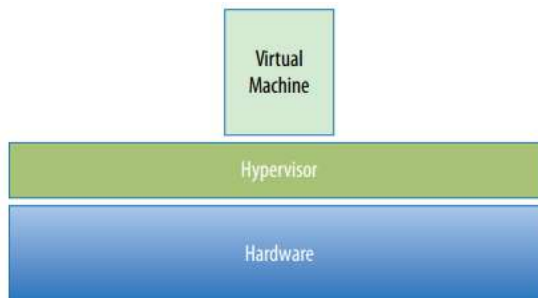
Figura 27 Consolidación de Servidores Fuente: (Portnoy, 2016)



1.16.2 Hypervisor

Es el software intermediario con el hardware que será utilizados por las instancias virtualizadas, este genera ambientes de virtualización basado en los requerimientos solicitados y virtualiza la sección de red para ser compartida y gestionada por cada una de ellas; dependiendo de su forma de interacción con el hardware se clasifican en dos tipos de hipervisores en el modelo de capas su ubicación se muestra en figura 28.

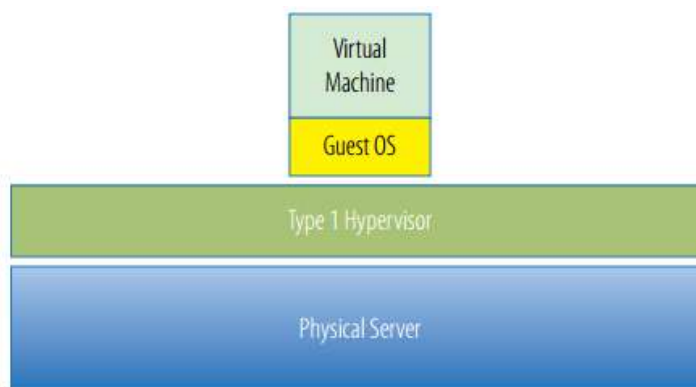
Figura 28 Nivel hypervisor, Fuente: (Portnoy, 2016) Hipervisor Tipo



- **Hipervisor Tipo 1**

Estos son los que se encuentran directamente establecidos sobre la capa de hardware, ofreciendo mayores prestaciones en rendimiento y una mejor administración de los equipos debido a su directa interactividad con los recursos físicos, tal como se muestra en la figura 29. Como ejemplo de hipervisor tipo 1 se puede considerar aplicaciones como Proxmox, Vmware ESXI entre los más comunes en el mercado.

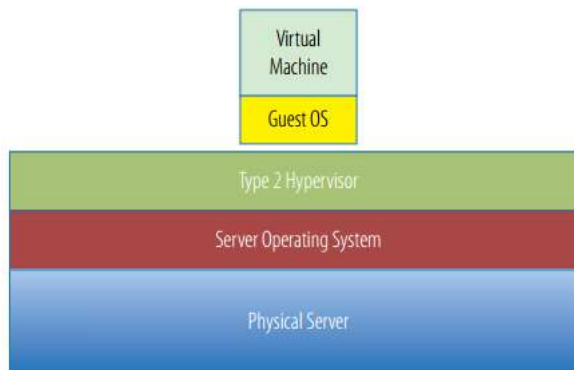
Figura 29 Hypervisor Type 1, Fuente: (Portnoy, 2016)



- **Hipervisor Tipo 2**

Este es un software que se ejecuta sobre un sistema operativo ya establecido en el hardware, es la implementación más común gracias a que la mayor parte de la integración y compatibilidad con dispositivos son heredados del propio sistema como se muestra en la figura 30, facilitando su gestión de recursos y administración. Como ejemplo de este tipo se puede mencionar a VMware Workstation/Player, VirtualBox y KVM como los más conocidos y utilizados.

Figura 30 Hypervisor Type 2, Fuente: (Portnoy, 2016)

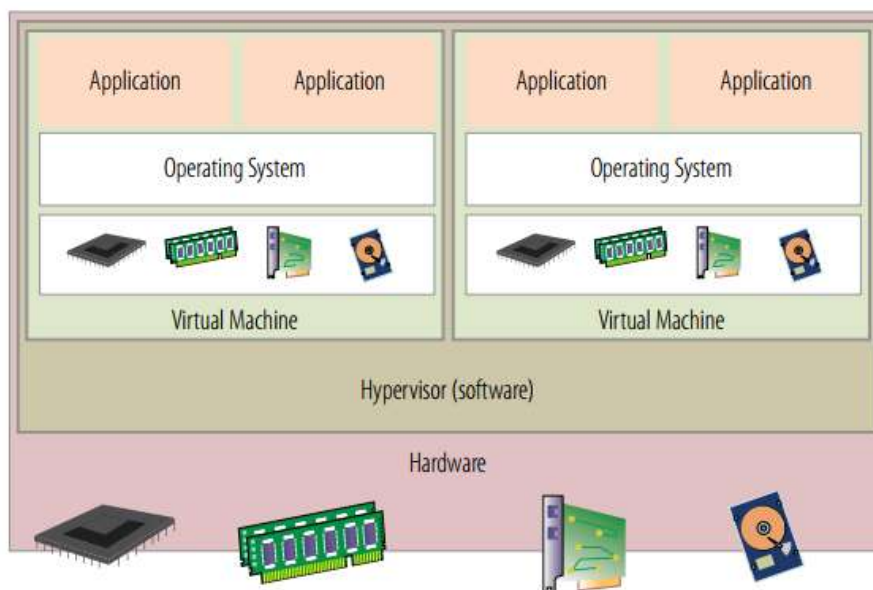


Algo notable sobre los hipervisores tipo 2, es que, al correr sobre un sistema operativo previo al hardware, este permitirá mayor nivel de integración y compatibilidad con dispositivos que se requiera virtualizar y a diferencia del tipo 1, las llamadas de sistema e instrucciones son ejecutadas basados basándose en la prioridad asignada del sistema operativo, esto lo convierte dependientes a la estabilidad del sistema sobre el cual están instalados. (Portnoy, 2016)

Funcionamiento.

Su funcionamiento es relativamente simple, una capa de software la cual se establece entre el hardware y las instancias virtuales que se desea correr. Realiza 3 tareas específicas: Proveer un ambiente virtualizado idéntico al físico, proveer un ambiente virtualizado con el mínimo costo de rendimiento y retener un completo control de los recursos del sistema, una representación de la asignación abstracta de recursos se presenta en la figura 31.

Figura 31 Asignación de Recursos, Fuente: (Portnoy, 2016)



bajo este esquema de compartición de recursos, se aprecia que a cada una de las instancias tiene asignada componentes virtualizados los cuales tienen independencia de control. (Portnoy, 2016)

1.16.3 Nomenclaturas de CPU

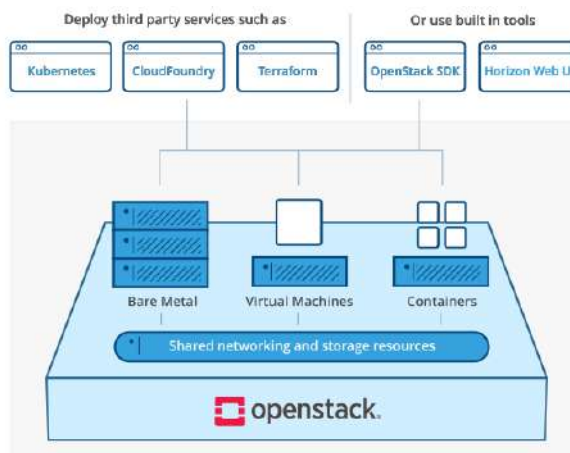
En virtualización existe designaciones de nomenclaturas para el cálculo y descripción de requerimientos de CPU, entre las cuales se tiene vCPU (Virtual CPU) el cual es utilizado para indicar cpu que son asignados a las instancias virtuales. pCPU (CPU físico) son notaciones que se da a los CPU físicos existentes, las notaciones de pCPU y vCPU se consideran a cada uno de los core, threads o hilo de procesamiento presentes en cada uno de los CPU

1.17 OpenStack

Es un software OpenSource de computación en la nube, el cual provee de las herramientas para establecer múltiples topologías de red basadas en SDN y NFV, gracias a su amplio stack de protocolos y módulos para la ejecución de sistemas virtualizados. Su estructura se asemeja a la de un hipervisor como se aprecia en la figura 31, con sus implementaciones de máquinas vituales, contenedores (Docker) y Bare-metal (instalación sobre hardware físico), además, permitiendo su administración a través de

múltiples herramientas de automatización externas y propietarias mostradas en la figura 32.

Figura 32 Estructura Openstack. Recuperado de: (Openstack, 2021)



1.17.1 Nube Pública.

Según NIST. Una nube pública es aquella en la que la infraestructura está abierta al público en general para su consumo, las nubes públicas de OpenStack suelen estar a cargo de un proveedor de servicios y pueden ser consumidas por individuos, corporaciones o cualquier cliente que pague. Un proveedor de nube pública puede exponer un conjunto completo de características, como redes definidas por software o almacenamiento en bloque, además de múltiples tipos de instancias. (NIST, 2021), estas pueden ser de servicios web, correo, o genéricas donde se provee un sistema Unix o Windows para que el cliente pueda establecer su funcionalidad, se debe notar que este tipo de aplicaciones e implementaciones pueden ser aplicadas en todos los tipos de nubes.

1.17.2 Nube Privada

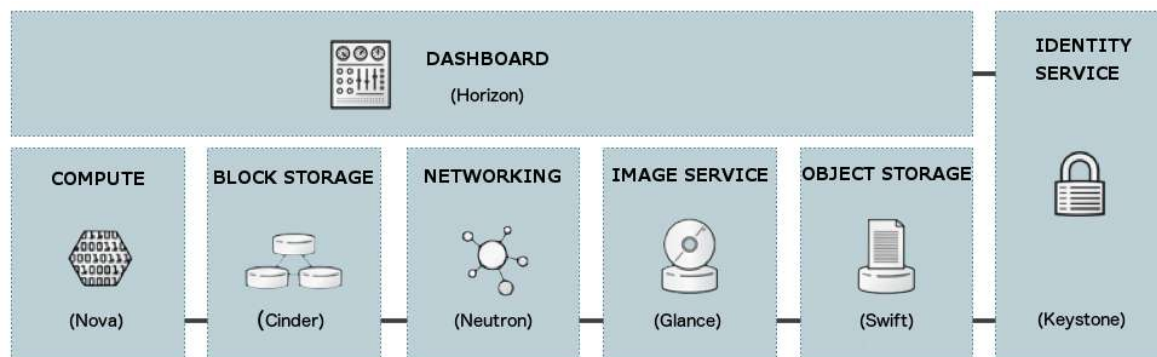
Según NIST. Está provista para uso exclusivo por una sola organización que comprende múltiples consumidores, como unidades de negocios. la nube puede ser propiedad administrada y operada por la organización, un tercero o alguna combinación de ellos, y puede existir dentro o fuera de las instalaciones, los casos de uso de la nube privada son diversos y, como tales, sus preocupaciones de seguridad individuales varían. (NIST, 2021)

1.17.3 Nube Híbrida.

Es una composición de dos o más infraestructuras de nubes distintas, como privadas, comunitarias o públicas, que siguen siendo entidades únicas, pero están unidas por tecnología estandarizada o patentada que permite la portabilidad de datos y aplicaciones, como su exposición para el equilibrio de carga entre nubes. por ejemplo, un minorista en línea puede presentar su publicidad y catálogo en una nube pública que permite un aprovisionamiento elástico, esto les permitiría manejar cargas estacionales de una manera flexible y rentable, una vez que un cliente comienza a procesar su pedido, se lo transfiere a una nube privada más segura cumpliendo el standard PCI(Consejo de Normas de Seguridad) (Openstack, 2021)

1.17.4 Componentes Openstack

Figura 33 Componentes OpenStack Recuperado de: www.openstack.org



Como se aprecia en la figura 33, existen múltiples módulos que conforman el sistema openstack, los mismos permiten un funcionamiento independiente de sus componentes, a su vez estos presentan una sinergia de integración y compatibilidad que los permite funcionar en conjunto de manera sistemática y eficiente, como ejemplo de esta integración presentamos una descripción breve de dos de sus componentes.

Compute. Es el componente encargado de gestionar las solicitudes de cómputo de CPU para toda la nube y las instancias que estén corriendo dentro de ella, estas pueden ser de tipo mono hilo y multi hilo.

Object Storage. Este componente provee el almacenamiento de objetos o bloques de información que son utilizados por las instancias internas de la nube o de la propia nube permitiendo un manejo de muy grandes cantidades de datos en los petabytes.

Para una mayor comprensión se muestra los componentes y sus funciones indicadas en la tabla 2 (Openstack, 2021)

Tabla 2 Componentes Openstack.

Nómbre	Nombre-Clave	Localización
Dashboard	horizon	Panel de control basado en navegador web que utiliza para administrar los servicios de OpenStack.
Identity	keystone	Servicio centralizado para la autenticación y autorización de servicios OpenStack y para la gestión de usuarios, proyectos y roles.
OpenStack Networking	neutron	Proporciona conectividad entre las interfaces de los servicios OpenStack.
Block Storage	cinder	Administra volúmenes de almacenamiento en bloque persistentes para máquinas virtuales.
Compute	nova	Administra y aprovisiona máquinas virtuales que se ejecutan en nodos de hipervisor.
Image	glance	Servicio de registro que usa para almacenar recursos como imágenes de máquinas virtuales e instantáneas de volumen.
Object Storage	swift	Permite a los usuarios almacenar y recuperar archivos y datos arbitrarios.
Telemetry	ceilometer	Proporciona mediciones de los recursos de la nube.

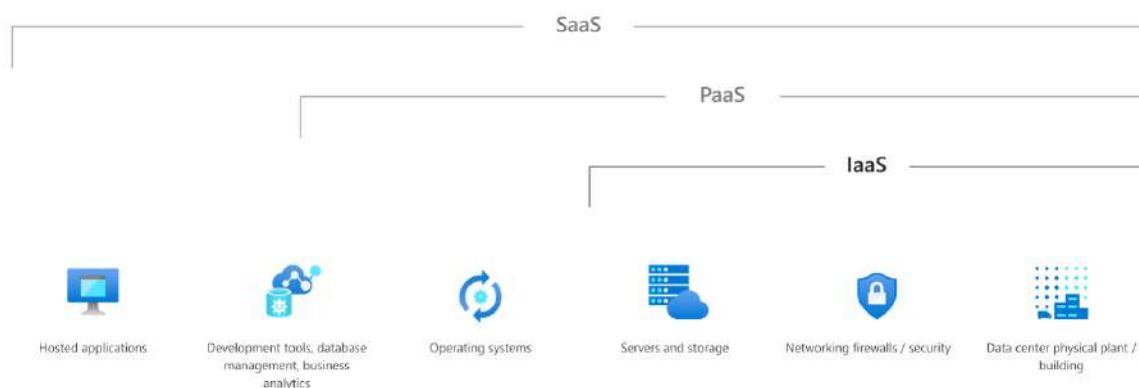
Orchestration	heat	Motor de orquestación basado en plantillas que admite la creación automática de pilas de recursos.
----------------------	------	--

Recuperado de: (Openstack, 2021)

1.18 Infraestructuras como Servicio.

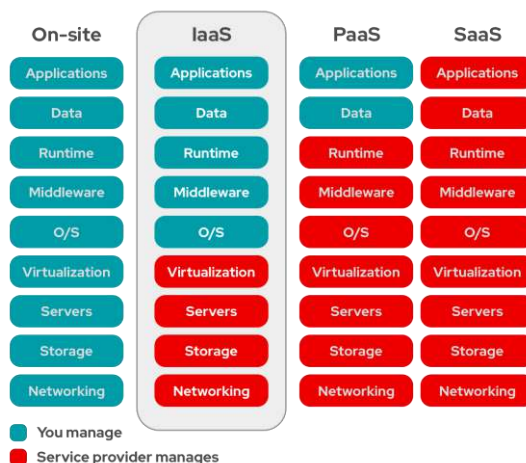
Según Microsoft, la infraestructura como servicio (IaaS) es un tipo de servicio de informática en la nube que ofrece recursos esenciales de proceso, almacenamiento y redes. IaaS es uno de los cuatro tipos de servicios en la nube, junto con el software como servicio (SaaS), la plataforma como servicio (PaaS) y la tecnología sin servidor, como se muestra en la figura 34

Figura 34 Categorías IAS, Fuente: Microsoft.com



Una infraestructura como servicio asume los costos de implementación de hardware para el despliegue de aplicaciones y servicios dentro de la misma, los usuarios externos deberán pagar una suscripción únicamente por los recursos que necesiten permitiendo que éstos puedan ser implementados con mayor facilidad y con la flexibilidad de poder actualizar su suscripción acorde a sus requerimientos.

Figura 35 Características de IAS. Recuperado de: www.redhat.com



Como se aprecia en la figura 35 al establecer una infraestructura existen niveles que pueden ser gestionados por el cliente (en azul) y niveles que son gestionados por el proveedor del servicio (en rojo), esto trae ventajas y desventajas comparándola a implementaciones Self-Hosting (implementación local propia), como se muestra en la tabla 3

Tabla 3 Ventajas y Desventajas

Ventajas de IaaS	Desventajas de IaaS
<p>✓ Desaparecen los costes por hardware, pueden regularse los gastos corrientes.</p>	<p>✗ Dependencia del proveedor, responsable absoluto de la disponibilidad y la seguridad del servicio.</p>
<p>✓ La implementación de proyectos nuevos es rápida.</p>	<p>✗ El acceso online es fundamental: cualquier problema de conexión repercute en el entorno virtualizado.</p>
<p>✓ Es muy flexible porque los recursos pueden escalarse fácilmente.</p>	<p>✗ Cambiar de proveedor es difícil.</p>
<p>✓ Desaparece el gasto por instalación, mantenimiento y modernización del hardware.</p>	<p>✗ La localización de los servidores del proveedor puede ser origen de conflictos con las normativas de protección de datos.</p>
<p>✓ Las diversas sedes de la empresa</p>	

IAS Recuperado de: (IONOS, 2019)

1.19 Mininet.

Mininet es una solución de software capaz de crear redes virtuales, las cuales se ejecutan sobre un kernel de Linux, corriendo código de aplicaciones de funciones de red dentro de una máquina virtual o bare-metal, su interfaz es mostrada en la figura 35 donde es apreciable una topología básica y los dispositivos de red que pueden ser utilizados en la misma.

Mininet provee un ambiente de pruebas y prototipado de redes SDN y aplicaciones Openflow permitiendo controlar y probar todos los aspectos de los dispositivos perteneciente a cada tipo de topologías, ya sean estas simples, complejas, paramétricas o arbitrarias. Su programación y configuración es sencilla gracias a su implementación en código Python que permite integraciones mediante su API. (García, 2016)

1.19.1 Containernet.

Este software es una variación de mininet el cual permite la integración de contenedores Docker, permitiendo agregar servicios y aplicaciones sobre el sistema base de host, integrándolos a las topologías de la red y así es posible realizar prácticas sobre escenarios reales. (Containernet, 2021), a su vez permite implementar toda la funcionalidad que tiene mininet, para el cumplimiento de ofrecer una plataforma de desarrollo de topologías y prácticas de redes, en los temas de SDN y NFV.

CAPITULO 3 Planificación, Diseño y Implementación

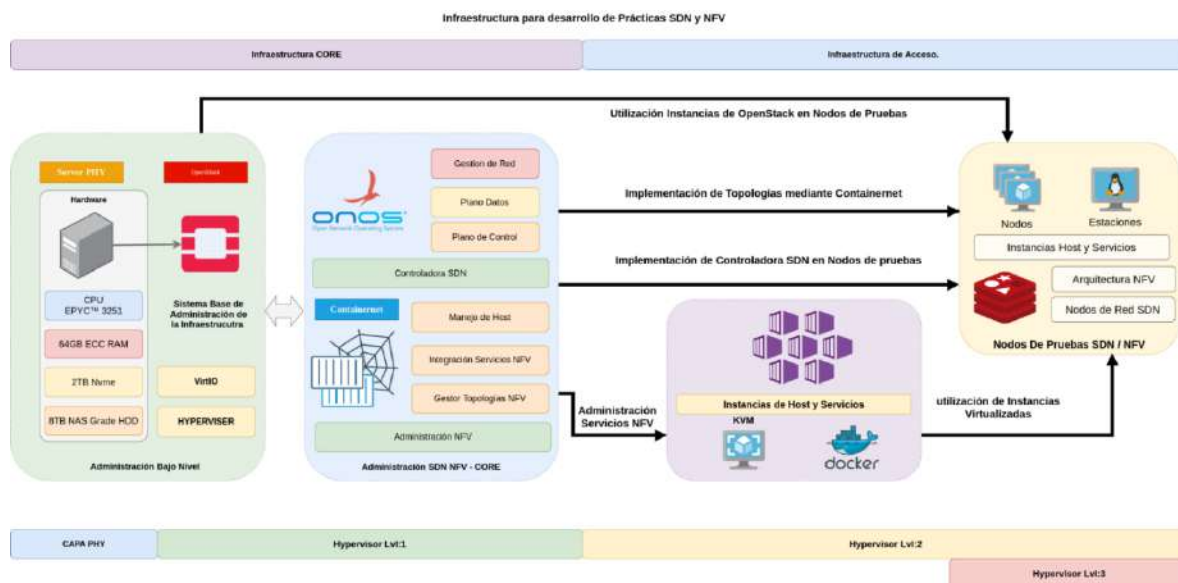
El presente capítulo se enfoca en la planificación, diseño e implementación del IAAS, tomando en consideración sus requerimientos de software y hardware.

1.20 Planificación y Arquitectura del Proyecto.

Para el desarrollo se considera la arquitectura mostrada en la ilustración 36 donde se presenta el funcionamiento y las relaciones que existen entre los diferentes componentes del proyecto, se destaca su organización en capas, la interconectividad desde la capa física hasta instancias y servicios alojados sobre un hipervisor de nivel 3. La capa física comprende el hardware del proyecto, sobre el cual se establecerá un hipervisor alojado mediante un sistema de Cloud Computing.

En un primer nivel de virtualización se tendrá las instancias generadas que permiten la administración SDN y NFV privadas. El segundo nivel presenta las instancias de nivel uno realizando virtualización anidada y uso de contenedores para alojamiento de servicios y gestión de red, puede existir un tercer nivel dependiendo de los requerimientos. Esta sección representa la Planificación indicada en su metodología.

Figura 36 Arquitectura del proyecto.



El sistema está establecido de acuerdo a los niveles de servicio cumpliendo funciones específicas de cada capa, permitiendo que sea modular y configurable, de esta manera se procede a realizar la etapa de **Diseño** considerando la metodología establecida para el proyecto.

1.20.1 Establecimiento de IAAS

Basándose en la arquitectura mostrada en la figura 36, es necesario considerar que cada capa de la implementación de la IAAS tiene su respectiva funcionalidad, la cual se debe cumplir para que los niveles superiores funcionen correctamente, siendo necesario realizar un análisis de requerimientos mínimos y recomendados para la implementación en cada una de ellas.

Una de las principales capas de la arquitectura IAAS es el hardware el cual soporta la carga de los sistemas e instancias de Cloud Computing, misma que se inicia con la instalación del sistema operativo base, en el cual se administrará todas las funciones básicas de acceso y gestión de recursos, tanto para los servicios iniciales del proyecto como también para la implementación de las instancias.

Requerimientos Hardware

Como punto inicial se establece los requerimientos fundamentales para la ejecución del proyecto, considerando las funciones de cada una de las capas establecidas, estos son:

- Sistema Operativo(OS) Base.
- Hypervisor Tipo 2.
- OS de nodo
- Software y Servicios en Nodo

Cada uno de los puntos indicados anteriormente establece requerimientos de hardware que debe cumplir para poder funcionar adecuadamente, estos se establecen en base a lo establecido por los diferentes fabricantes que se consideran en el presente proyecto, la tabla 4 muestra los valores sugeridos por los mismos.

Tabla 4 Requerimientos HW/SW

Software	CPU[Cores]	RAM[GB]	Almacenamiento [GB]
-----------------	-------------------	----------------	--------------------------------

OS BASE	1	1	2.5
Hypervisor N1T2	4	8	25
OS Nodo	1	1	2.5
ONOS	2	2	10

Adaptado de : (IONOS, 2019) (Openvswitch.org, 2021) (Ubuntu.com, 2021) (Portnoy, 2016).

Una vez que los requerimientos han sido establecidos en la tabla 4, es necesario conocer el número de instancias virtuales que serán implementadas, de esta manera se realiza un dimensionamiento general de la infraestructura en aspectos de hardware.

Se establece la implementación de 10 instancias virtuales considerando el análisis de requerimientos realizado en el **anexo 3**, conociendo este valor, a continuación, se realiza los cálculos necesarios para obtener los requerimientos de hardware de la infraestructura los cuales incluye Almacenamiento, memoria RAM y CPU.

Requerimientos de Almacenamiento.

En este apartado se realiza el cálculo del almacenamiento requerido para la infraestructura considerando las demandas indicadas por cada uno de los fabricantes de software. A estos valores se les agregará un 20% por motivos de crecimiento y expansión del sistema, los requerimientos son mostrados y tabulados en la tabla 5.

Tabla 5 Cálculo almacenamiento Infraestructura.

Software	Requerimiento (GB)	Cantidad	Total (GB)
OS BASE	3	1	3
Hypervisor N1T2	30	1	30

Imagen Nodo	20	10	200
ONOS	12	1	12
		TOTAL	245 GB

Con la tabla 5 se establece que existe un requerimiento de almacenamiento mínimo de 245GB, considerando que comercialmente la capacidad de disco más cercano es de 256GB, pero, sin embargo, debido al espacio que ocupa el sistema de archivos queda una capacidad utilizable de 240 GB, por lo cual esta opción se descarta. La capacidad siguiente en discos es de 512GB con 480GB utilizables, lo que lo convierte en el disco a elegir para el despliegue de la infraestructura en el presente trabajo, adicionalmente, considerando un crecimiento de un 50% en el número de instancias virtuales se podrá observar que el almacenamiento llegaría a una utilización del 76.5%, por esta razón como proyección de crecimiento se establece como disco recomendado uno de 1TB de capacidad.

Requerimiento Memoria RAM

En este apartado se dimensiona la cantidad de memoria RAM a ser utilizada en la infraestructura en el presente trabajo, para ello se considera lo que utiliza cada uno del software a implementar, así como los requerimientos generales de las instancias virtuales. Se debe señalar que para este análisis se considera la norma IEEE-830, (IEEE, 1998), la cual especifica que los requerimientos indicados de un software para ser instalado sobre un sistema operativo incluyen los requerimientos de este, permitiendo restar los requerimientos del sistema a los requerimientos del software a implementar.

Requerimientos memoria RAM instancia virtual,

Considerando los requerimientos indicados en la tabla 4, considerando el estándar IEEE-830 antes mencionado se establece los siguientes requerimientos para la instancia virtual, donde la cantidad de memoria RAM se obtiene mediante la Ec.1, donde, se toma en consideración la memoria del SO Nodo y la controladora ONOS.

$$RAM_{Instancia_{total}} = (RAM_{Sistema_{nodo}} + RAM_{ONOS}) \text{ Ec. 1}$$

Donde $RAM_{Sistema_{nodo}}$ es equivalente a 1GB y RAM_{ONOS} es equivalente a 1GB, estos valores son reemplazados en la Ec.1, por tanto, el resultado de $RAM_{Instancia_{total}}$ es equivalente a: 2GB como requerimiento de memoria RAM para las instancias virtuales.

Requerimientos memoria RAM de la infraestructura

Este requerimiento se obtiene considerando los requerimientos de memoria RAM del software a utilizar en la implementación sobre el hardware de la infraestructura indicados en la tabla 4, donde la cantidad de RAM se obtiene mediante la Ec. 2, como son SO BASE y Hypervisor, considerando la recomendación de la IEEE-830, realizando un análisis similar al de requerimientos de instancia virtual.

$$RAM_{Infraestructura} = (RAM_{Sistema_{base}} + RAM_{Hypervisor}) \text{ Ec. 2}$$

Donde $RAM_{Sistema_{base}}$ es equivalente a 1GB y $RAM_{Hypervisor}$ es equivalente a 7GB, por tanto, el resultado de $RAM_{Infraestructura}$ es equivalente a 8GB como requerimiento de memoria RAM para la infraestructura.

Requerimiento memoria RAM general de la infraestructura.

En este punto se realiza el cálculo del requerimiento de memoria total de la infraestructura mediante la Ec. 3 la cual incluye los valores de instancia virtual obtenidos en la Ec.1, así como los valores de infraestructura obtenidos en la Ec. 2 y el número de instancias virtuales a considerar.

$$RAM_{total} = (RAM_{Instancia_{total}} \times No. instancia) + RAM_{Infraestructura} \text{ Ec. 3}$$

Donde $RAM_{Instancia_{total}}$ es equivalente a 2GB, $RAM_{Infraestructura}$ es equivalente a 8GB, $No. instancias$ es equivalente a 10, por tanto, el resultado de RAM_{total} es equivalente a 28GB

Mediante el cálculo anterior se obtiene un valor de memoria requerida de 28GB de RAM, pero considerando el apartado de (Microsoft, Virtual vs Physical Memory?, 2017) especifica que no se requiere cumplir con toda la memoria de manera física, ya que

el sistema puede establecer 1.5 veces esta memoria utilizando memoria virtual, por lo que la memoria física requerida es calculada mediante la Ec. 4, la cual considera la constante indicada por Microsoft y la RAM total calculada mediante la Ec. 3

$$Memoria_{Total\ Física} = \frac{RAM_{total}}{k} \quad Ec. 4$$

Donde $Memoria_{Total}$ es equivalente a 28GB, la constante k dada por Microsoft equivale a 1.5, obteniendo como resultado 18.6GB

Mediante este cálculo se obtiene la memoria total requerida por el sistema, la cual será de 18.6GB de RAM, valor que no puede ser obtenido comercialmente, sin embargo el valor más próximo equivale a 24GB, a su vez, es equivalente al 85% de la memoria total requerida. Si se considera un crecimiento del 50% en sus instancias virtuales, esta cantidad es insuficiente, por tal motivo, para permitir este crecimiento a futuro se recomienda la implementación de 32 GB de RAM

Requerimientos de CPU

El dimensionamiento de CPU de la infraestructura, se basa en lo señalado por (VMware, Performance Best Practices for VMware vSphere, 2022) en donde establece una relación entre vCPU (CPU virtuales) y pCPU (CPU Físico), de 3:1 es decir 3 vCPU por cada pCPU, además se considera la norma IEEE-830 donde indica que es posible restar los requerimientos del sistema operativo a los requerimientos del software a ser instalado.

Requerimiento vCPU instancia virtual.

En este punto se realiza el cálculo de los vCPU requeridos por la instancia virtual considerando los valores indicados en la tabla 4, este valor se obtiene mediante la Ec. 5 la cual considera SO Nodo y ONOS.

$$vCPU_{Instancia} = (vCPU_{SO\ Nodo} + vCPU_{ONOS}) \quad Ec. 5$$

Donde $vCPU_{SO\ Nodo}$ es equivalente a 1 vCPU y $vCPU_{ONOS}$ es equivalente a 1 vCPU, por tanto, el resultado de $vCPU_{Instancia}$ es equivalente a 2 vCPU obteniendo como resultado 2 vCPU

Dimensionamiento pCPU para infraestructura base.

Se considera los elementos a utilizar directamente en el host físico y antes de las instancias virtuales por lo que el cálculo se realiza en pCPU, utilizando los requerimientos nombrados en la tabla 4, este valor se obtiene mediante la ecuación 6, la cual considera SO Base y Hypervisor.

$$pCPU_{base} = (pCPU_{So\ Base} + pCPU_{Hypervisor}) \quad \text{Ec. 6}$$

Donde $pCPU_{So\ Base}$ es equivalente a 1 pCPU y $pCPU_{Hypervisor}$ es equivalente a 3 pCPU por tanto, el resultado de $vCPU_{base}$ es equivalente a 4 pCPU

De esta manera se establece que la infraestructura base requiere 4pCPU mismos que se utilizarán para el cálculo general de requerimientos el cual se muestra a continuación donde se considera la relación de 3:1 entre vCPU:pCPU.

Con los valores obtenidos anteriormente se procede al cálculo de pCPU totales, este valor se obtiene mediante la Ec. 7.

$$pCPU_{Total} = (pCPU_{base} + \frac{vCPU_{Instancia} * No.Instancias}{3}) \quad \text{Ec. 7}$$

Donde $pCPU_{base}$, es equivalente a 4 pCPU, $pCPU_{Instancia}$, es equivalente a 2 pCPU No. Instancias se establece en 10 instancias, por tanto, el resultado de $pCPU_{Total}$ es equivalente a 10.3 pCPU

Mediante el cálculo indicado en la Ec. 7 se establece como requerimiento de la infraestructura un requerimiento de 10.3 pCPU. En el mercado no es posible obtener un procesador que cuente con ese número de pCPU por tal motivo se recomienda utilizar un procesador con el número de pCPU más aproximado de 12 pCPU, mismo que se establece como requerimientos mínimos para la infraestructura y de la misma manera con el objetivo de expansión y crecimiento en un 50% de la misma se recomienda un procesador de 16pCPU.

Los valores obtenidos mediante los cálculos anteriores serán verificados utilizando la herramienta alojada en <https://wintelguy.com/vmcalc.pl>, la cual permite obtener información sobre utilización de recursos considerando la relación de vCPU y pCPU así como la utilización de memoria RAM. Para ello se consideran 2 casos que se describen a continuación.

Caso 1: Se establece un CPU de 12 Cores y 24GB de RAM, en la figura 37 se muestra la asignación de los recursos virtuales indicados en la sección 1, así como los del host físico de la sección 2, valores con los que se realiza el análisis y se obtiene los resultados presentados en la figura 38.

Figura 37 Calculadora Virtualización.

This calculator evaluates the number of hosts in a Hyper-V cluster for the given legacy server workload and host specification.

Physical servers to be consolidated (VM candidates):

# of CPU	Cores per CPU	CPU Speed (GHz)	Aver. CPU util. (%)	RAM (GB)	Aver. RAM util. (%)	## of servers
1) 1	1) 2	1) 4.1	1) 50	1) 2	1) 50	1) 20
2)	2)	2)	2)	2)	2)	2)
3)	3)	3)	3)	3)	3)	3)
4)	4)	4)	4)	4)	4)	4)

Host specification:

# of CPUs	Cores per CPU	CPU speed (GHz)	RAM (GB)
1	12	4.1	24

Figura 38 Resultados Caso 1

Number of hosts:	2
Total number of physical CPUs:	2
Total number of physical CPU cores:	24
Total Virtual Processors:	40
Virtual Processors per physical CPU core ratio:	A 1.67 : 1
Average number of Virtual Processors per host:	20
Total cluster CPU capacity (GHz):	98.4
Total hypervisor CPU overhead (GHz):	5.4
Cluster CPU capacity available for VMs (GHz):	93
CPU capacity required for VM workload (GHz):	90.2
Average host CPU utilization (%):	B 96.99
Total cluster memory capacity (GB):	48
Total hypervisor memory overhead (GB):	4
Cluster memory capacity available for VMs (GB):	44
Memory capacity required for VM workload (GB):	24
Average host memory utilization (%):	C 54.55
Average number of VMs per host:	10 D

Como resultados del caso 1 se tiene que la relación de utilización de vCPU:pCPU es de 1.67:1(ver figura 38 ítem a) siendo este valor menor a la relación 3:1 establecida en la Ec.7, esto significa que la división de pCPU a vCPU es menor lo que indica un mejor rendimiento. La utilización de CPU(ver figura 38 ítem b) presenta un valor de 96.99% es decir que el procesador estará trabajando al límite de su capacidad y no ofrece espacio para expansión o crecimiento. Por otra parte en cuanto a la memoria RAM(ver figura 38 ítem c) la herramienta arroja una utilización de 54.55%, indicando que esta será suficiente para la aplicación, tal y como se esperaba en la Ec. 4 donde se indicaba una utilización del 85%. Al verificar el número de instancias virtuales que el sistema puede soportar se obtiene un valor de 10 (ver figura 38 ítem d), el cual corresponde al número de instancias requerido por el proyecto.

Caso 2: Se considera un sistema con un CPU de 16 Cores y con 32 de Ram, datos que son ingresados en la calculadora obteniendo los resultados presentados en la figura 39.

Figura 39 Resultados caso 2

Number of hosts:	2
Total number of physical CPUs:	2
Total number of physical CPU cores:	32
Total Virtual Processors:	40
Virtual Processors per physical CPU core ratio:	A 1.25 : 1
Average number of Virtual Processors per host:	20
Total cluster CPU capacity (GHz):	131.2
Total hypervisor CPU overhead (GHz):	5.4
Cluster CPU capacity available for VMs (GHz):	125.8
CPU capacity required for VM workload (GHz):	90.2
Average host CPU utilization (%):	B 71.7
Total cluster memory capacity (GB):	64
Total hypervisor memory overhead (GB):	4
Cluster memory capacity available for VMs (GB):	60
Memory capacity required for VM workload (GB):	24
Average host memory utilization (%)	C 40
Average number of VMs per host:	10 D

Como resultados del caso 2 se tiene que la relación de utilización de vCPU:pCPU es de 1.25:1(ver figura 39 ítem a) siendo este valor menor a la relación 3:1 establecida en la Ec.7, esto significa que la división de pCPU a vCPU es menor lo que representa un mejor rendimiento. La utilización de CPU(ver figura 39 ítem b) presenta un valor de 71.7% lo que indica que el procesador no se encuentra a su límite de su capacidad y es posible realizar una expansión o crecimiento del proyecto. Por otra parte en cuanto a la memoria RAM(ver figura 39 ítem c) la herramienta arroja una utilización de 40%, indicando que la memoria asignada será suficiente, tal y como se esperaba en la Ec. 4 donde se indicaba una utilización del 85%. Al verificar el número de instancias virtuales que el sistema puede soportar se obtiene un valor de 10 (ver figura 39 ítem d), el cual corresponde al número de instancias requerido por el proyecto.

Una vez realizado el análisis y comprobación de los requerimientos obtenidos, se establecen estos como los valores a utilizar para la implementación del proyecto, utilizando los requerimientos mínimos indicados en la tabla 9

Como resultado del análisis de requerimientos realizado anteriormente, se realiza la tabla 7 donde se indican los 2 casos y sus valores recomendados, para ser utilizados en la implementación de la infraestructura.

Tabla 6 Requerimientos considerados para la arquitectura

Requerimiento	CPU/Cores	RAM/GB	Almacenamiento /GB
Mínimo	6	24	512 SSD
Recomendado	12	32	1024 SSD

Fuente: Autor

Selección Infraestructura de HW

Con los requerimientos de infraestructura de HW obtenidos de la tabla 7, a continuación se procede con la selección del hardware a utilizarse, para ello, tal como se presenta en la tabla 8, se compara los parámetros de CPU, RAM y almacenamiento, características importantes para el rendimiento de la infraestructura, considerando los fabricantes que se tiene disponibles para el presente proyecto.

Tabla 7 Comparación de infraestructura de HW

Fabricante	CPU-Model/cores	RAM/GB	Almacenamiento
HP Pavilion	i3-2310M/4	8	1TB HDD
Asus Strix Scar	I7-8750h/12	32	1TB NVME
MSI GP72VR	I7-9750h/12	32	512GB NVME

La comparación de rendimiento se basa en la metodología de Benchmarking, para ello se utiliza los puntajes disponibles en (cpubenchmark, 2021) de cada uno de los CPU que se dispone para el desarrollo del presente trabajo (Ver tabla 8). En este sentido los valores de rendimiento/puntaje se interpretan en el sentido de que mayor es mejor. De manera que, se evidencia que el sistema **MSI GP72V** con el procesador i7-9750h es el sistema óptimo y por ende el que se utilizará en el desarrollo de la Infraestructura.

Tabla 8 Benchmark CPU

Sistema	CPU-Model/cores	Rendimiento/Puntaje
HP Pavilion	i3-2310M/4	1,166
Asus Strix Scar	I7-8750h/12	10,171
MSI GP72VR	I7-9750h/12	11,318

Obtenido de: (cpubenchmark, 2021)

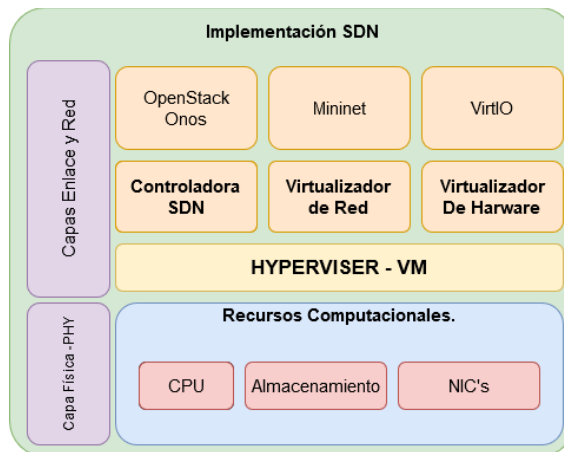
1.21 Diseño y Dimensionamiento de arquitectura.

Basado en los resultados obtenidos en el paso anterior de la metodología, se realiza el proceso de **Diseño** de la IAAS, delimitando el hardware y software a utilizar, donde se considerará los requerimientos de cada capa de la arquitectura del proyecto, con el fin de obtener el mejor desempeño posible del hardware seleccionado.

1.21.1 Sistema Base SDN

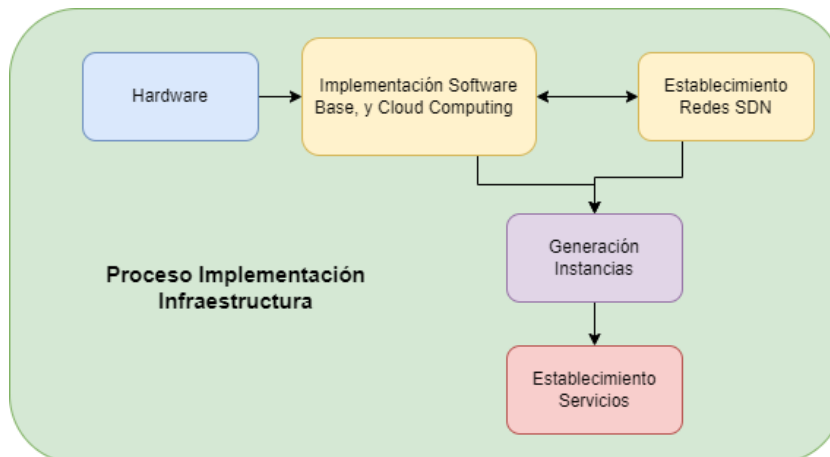
El sistema base para el proyecto según esta establecido en la arquitectura en la figura 40 presenta una configuración en capas donde se define las funciones y software necesario de cada una de ellas indicadas en la parte superior como son Hypervisor, funciones de red y alojamiento de instancias virtuales, mientras que en la parte inferior se indican los recursos físicos de la infraestructura.

Figura 40 Implementación SDN



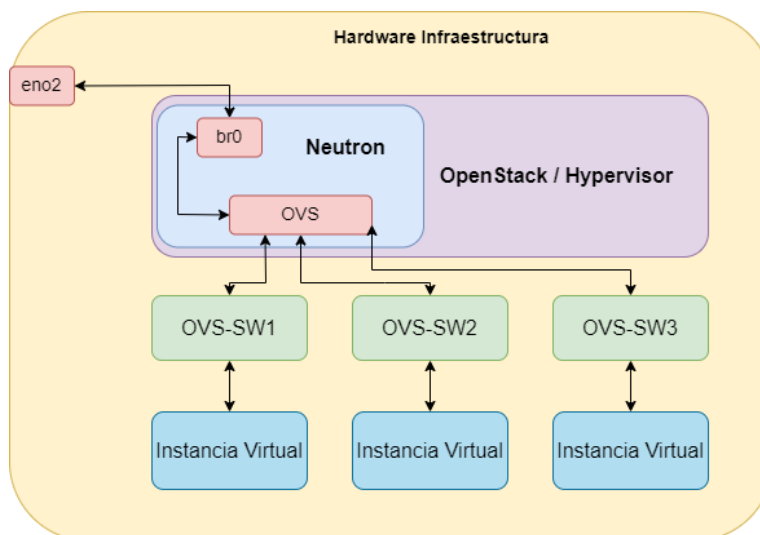
Al considerar las capas y funciones necesarias para la IAAS, se requiere un proceso de implementación el cual sigue un orden específico mostrado en la figura 41, empezando desde el hardware determinado en el paso anterior y el software requerido por cada capa de la IAAS indicada en la topología, hasta llegar a la implementación de nodos.

Figura 41 Implementación Infraestructura.



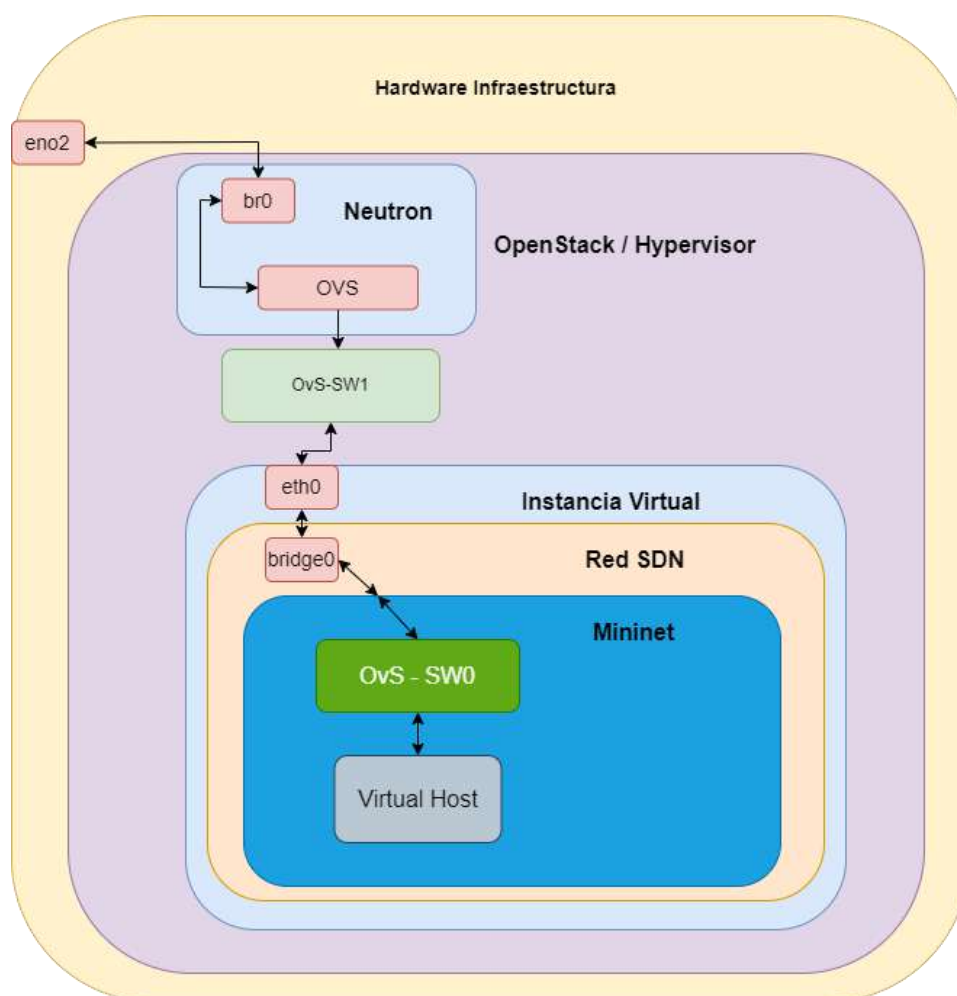
Para la implementación de la red SDN en la infraestructura, se establece el diagrama presentado en la figura 41, la cual presenta la arquitectura, así también los niveles de administración de redes SDN, este indica que existe una conectividad al exterior mediante una interfaz física eno2, esta interfaz estará conectada mediante un puente hacia la interfaz virtual br0, misma que está gestionada mediante el software de cloud computing OpenStack, esta permite establecer redes SDN mediante su controladora interna Neutron, estas pueden ser asignadas a las instancias mediante interfaces virtuales utilizando OpenvSwitch, estableciendo la funcionalidad de redes SDN a nivel del Hypervisor.

Figura 42 Diseño SDN de la infraestructura.



En la figura 42 se muestra la interconectividad que existe entre las instancias virtuales, mediante las interfaces virtuales OVS asignadas por el hypervisor, mientras que en la figura 43 se presenta las conexiones existentes con las redes SDN y NFV que pueden ser generadas en la instancia mediante su interfaz eth0 la cual se interconecta con la interfaz virtual bridge0, misma que esta administrada por mininet y la controladora SDN para la gestión de las redes SDN y administración NFV.

Figura 43 SDN/NFV Instancia virtual



Selección Hypervisor

Para la selección de un Hypervisor es necesario tomar en cuenta los siguientes requerimientos:

- Contenido a virtualizar
- Aprovisionamiento de recursos por host
- Compatibilidad de Hardware y software
- Escalabilidad.

Contenido a virtualizar Al trabajar en entornos virtualizados es necesario conocer el tipo de contenido que se debe virtualizar de manera que la elección de un hipervisor sea acorde a sus necesidades.

Aprovisionamiento de recursos. Se refiere a la escalabilidad en la asignación de recursos y la mínima reducción de rendimiento en las instancias virtuales en comparación a las implementaciones bare-metal.

Compatibilidad Hardware y software. Se refiere a un hypervisor que permita la interactividad con múltiples dispositivos y recursos para ser utilizados en las instancias virtuales, debido al tipo de implementación que se desea realizar se requiere ejecutarlo sobre un sistema operativo es decir se solicita un hypervisor tipo 2.

Escalabilidad. Esta característica se refiere a la adaptabilidad de crecimiento en base las necesidades y requerimientos que demandan nuevas aplicaciones, en este sentido el hypervisor debe permitir la creación del mayor número de instancias virtuales.

Una vez con los parámetros y requerimientos definidos se establece la tabla 10 indicando su cumplimiento por parte del software seleccionado; siendo 1 si cumple y 0 no cumple.

Tabla 9 Detalle Hipervisor

Requerimiento	Detalle	Hipervisor Lvl-1	Hipervisor Lvl-2
Contenido a Virtualizar	Virtualizar Sistemas Completos	1	1

Aprovisionamiento de recursos	Realiza la aplicación de recursos virtualizados	1	1
Compatibilidad HW/SW	Funcionalidad sobre un OS pre Instalado y Software Instalado Conjuntamente.	0	1
Escalabilidad	Permita Agregar y Manejar múltiples puntos de Virtualización	1	1

Selección Sistema Operativo Base

OpenStack puede ser instalado sobre varios SO como son: Ubuntu Server LTS, Fedora, CENTOS/RHEL, OpenSuse. OpenStack fue liberado inicialmente para sistemas RHEL (Red Hat Enterprise Linux) de la firma RED HAT, que provee sistemas Linux para empresas, una vez realizado el desarrollo y maduración del software, es ahora posible instalarlo en diferentes distribuciones de Linux, facilitando de esta manera su desarrollo, permitiendo ser utilizado para la integración de muchos servicios en ambientes que estén gestionados mediante otras distribuciones. (Openstack, 2021)

De esta manera el sistema seleccionado es Ubuntu en su edición Server LTS, por la familiaridad del sistema y su gestión facilitando el desarrollo del proyecto y garantizando la compatibilidad de sus componentes, los cuales requieren de esta distribución, de esta manera en la tabla 9 se indica las valoraciones utilizadas para la implementación del sistema seleccionado; siendo 1 si cumple y 0 no cumple.

Tabla 10 Selección Sistema Operativo

Requerimiento	Funcionalidad	Ubuntu Server LTS	RHEL/Fedora	Opensuse
OpenStack	Compatibilidad Cloud Computing	1	1	0
Onos	Controladora SDN	1	1	1
Containernet	Gestor Virtualización NFV	1	0	0

Definición de Controladora SDN

Para el control de las redes SDN internas del sistema se utilizará la controladora integrada Neutron de OpenStack, permitiendo la conectividad entre las instancias virtualizadas y el exterior.

1.21.2 Establecimiento Software en Nodos

En esta sección se establece el software a implementar como SO (Sistema Operativo) de las instancias en OpenStack, las mismas que serán utilizadas para la instalación de los servicios, cumpliendo con el requerimiento de ser muy liviano y compatible con el hypervisor.

Para la imagen que se utilizará se definió Ubuntu para el desarrollo de los nodos y sobre el cual funcionarán los servicios, ya que las utilidades y servicios a ser implementados tienen como requerimiento la utilización de esta distribución.

OpenVSwitch

Para el cumplimiento de la capa de SDN y NFV en las instancias, se requiere un software que pueda ser implementado como switch que permita su conectividad, para este fin, se estableció el software OVS, ya que es un componente de la controladora SDN de OpenStack, el cual permite la generación de interfaces virtuales con capacidad de integrar

protocolos como OpenFlow y tradicionales de manera transparente, así como puede ser utilizado por la controladora SDN a implementarse en la instancia como es ONOS.

Controladora ONOS

Con el fin de obtener la mejor interoperabilidad posible entre las aplicaciones a ser instalados en la instancia virtual, se establece como controladora SDN al software ONOS, mismo que permite la integración directa con los software OVS y Mininet, brindando su soporte y configuración de manera automatizada.

Containernet

Para la realización de prácticas SDN sobre la instancia virtual, se requiere un software que permita la integración de todos los servicios y a su vez generar las topologías requeridas, de esta manera se eligió el software Containernet el cual cumple con estos requerimientos.

Una vez finalizada la etapa de **Diseño** se procede a realizar la etapa de **Implementación** tomando en consideración todo lo elaborado en las etapas anteriores, cumpliendo de esta manera con el proceso indicado en metodología del proyecto.

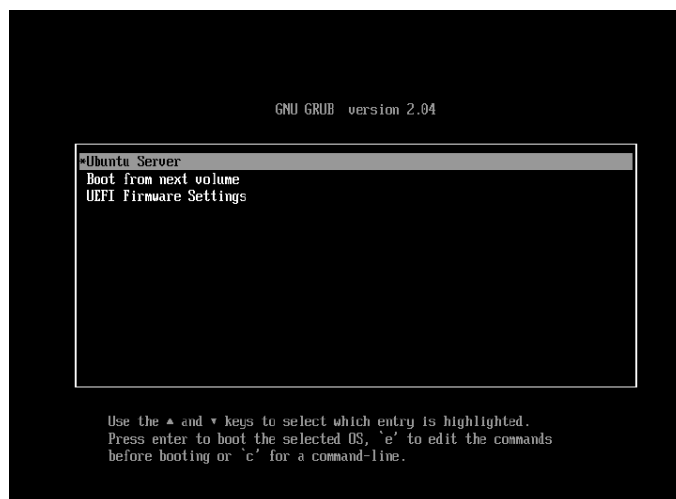
1.22 Implementación de la arquitectura.

La implementación está basada en la infraestructura presentada en la figura 36, tomando en consideración la funcionalidad de cada una de las capas de la arquitectura, la cual se presenta desarrollada en este punto. De la misma manera se continúa con el desarrollo de la metodología en su fase de **Implementación** del proyecto, cumpliendo con los requerimientos indicados en la arquitectura en sus etapas y funcionalidades.

1.22.1 Instalación SO Base

Como primer paso desarrollamos la instalación del sistema base sobre el hardware especificado, para el desarrollo del proyecto se ha elegido instalar Ubuntu Server, su instalación se detalla a continuación indicando en la figura 40 el instalador del sistema. La Instalación Completa del Sistema base está documentada en el **Anexo 1**.

Figura 44 Instalador Ubuntu Server.



Instalación KVM

Posterior a la instalación del sistema operativo base, se procede a instalar el paquete KVM, el cual es requerido por OpenStack para sus funciones de virtualización, este procedimiento se muestra en la figura 45, la cual evidencia la instalación de todas sus dependencias y paquetes; para ello es necesario ejecutar el siguiente comando: *sudo apt install -y qemu qemu-kvm libvirt-daemon libvirt-clients bridge-utils virt-manager*

Figura 45 Instalación KVM.

```

zoid@openstack:~$ sudo apt install -y qemu qemu-kvm libvirt-daemon libvirt-clients bridge-utils virt-manager
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'qemu-system-x86' instead of 'qemu-kvm'
The following additional packages will be installed:
acl dns-root-data dnsmasq-base genisoimage gir1.2-atk-1.0 gir1.2-ayatanaappindicator3-0.1
gir1.2-freedesktop gir1.2-gdkpixbuf-2.0 gir1.2-gstreamer-1.0 gir1.2-gtk-3.0
gir1.2-gtk-vnc-2.0 gir1.2-gtksource-4 gir1.2-harfBuzz-0.0 gir1.2-libosinfo-1.0
gir1.2-libvirt-glib-1.0 gir1.2-pango-1.0 gir1.2-spiceclientglib-2.0
gir1.2-spiceclientgtk-3.0 gir1.2-vte-2.91 gstreamer1.0-plugins-base
gstreamer1.0-plugins-good gstreamer1.0-x-ibverbs-providers ipxe-qemu
ipxe-qemu-256k-compact-efi-roms jq libayatana-appindicator3-1 libayatana-ido3-0.4-0
libayatana-indicator3-7 libboost-iostreams1.74.0 libboost-thread1.74.0 libbrlapi0.8
libcacard0 libcdparanoia0 libdaxctl1 libdvd libfdt1 libgiovann-common libgiovann2
libgstreamer-plugins-base1.0-0 libgstreamer-plugins-good1.0-0 libgtk-vnc-2.0-0
libgtksourceview-4-0 libgtksourceview-4-common libgvc-1.0-0 libibverbs1 libiec1803-0
libiscsi7 libjq1 libndctl6 libnss-mymachines libonig5 libosinfo-1.0-0 libpangoft2-1.0-0
libphodav-2.0-0 libphodav-2.0-common libpman1 librados2 librbd1 librdmacm1 libslirp0
libspice-client-glib-2.0-0 libspice-client-gtk-3.0-0 libspice-server1 liburing1
libusbredirhost1 libusbredirparser1 libv4l-0 libv4lconvert0 libvirglrenderer1
libvirt-daemon-config-network libvirt-daemon-config-nwfilter libvirt-daemon-driver-qemu
libvirt-daemon-system libvirt-daemon-system-systemd libvirt-glib-1.0-0 libvirt0
libvisual-0.4-0 libyajl2 mdevctl osinfo-db ovmf python3-cairo python3-gi-cairo
python3-libvirt python3-libxml2 qemu-block-extra qemu-system-common qemu-system-data
qemu-system-gui qemu-utils seabios sharutils spice-client-glib-usb-acl-helper
systemd-container virt-viewer virtinst
Suggested packages:
ifupdown wedim cdrkit doc gvfs libdvd-bin oss-compat libvisual-0.4-plugins libosinfo-l10n

```

1.22.2 Implementación IAAS.

Considerando los niveles de implementación de una IAAS indicados en la sección 2.13, se realiza la instalación del software habilitante para el efecto, el cual acorde a la selección realizada en el apartado 2.12 se utilizará OpenStack. Para ello, gracias a la herramienta de instalación de desarrollo llamada DevStack, se instala todo el repertorio de aplicaciones y módulos de Openstack desde sus códigos fuente, este método demanda más tiempo de instalación, pero de esta manera se obtiene la versión más actualizada a su vez que una mejor compatibilidad de hardware.

En este sentido, se descarga desde la página oficial <https://docs.openstack.org/devstack/latest/> y se procede con los pasos de instalación indicados a continuación.

Generación usuario stack

En primer lugar, es necesario la generación de un usuario único en este caso se crea el usuario *stack* específicamente para poder realizar funciones fuera del dominio del usuario administrador, de esta manera las funciones automatizadas no requieren su intervención, esto se lo genera con el siguiente comando, donde *sudo* permite elevar los privilegios ejecución, por otra parte *useradd* permite agregar un usuario al Sistema, utilizando la opción *-s /bin/bash* se establece el binario a utilizar para ejecución de comandos, con la opción *-d /opt/stack* es posible especificar la locación de la carpeta del usuario y finalmente *-m stack* permite establecer el nombre del usuario.

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

La asignación de sus permisos y privilegios se realiza con el siguiente comando por medio del comando *echo "stack ALL=(ALL) NOPASSWD: ALL"* el cual establece todos los permisos de ejecución para el usuario *stack*, por otra parte la sección *sudo tee /etc/sudoers.d/stack* del comando indica el archivo en el cual establecer los privilegios.

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

de esta manera se autoriza al usuario **stack** para su funcionamiento autónomo dentro del sistema.

Obtención Código Fuente.

Para obtener el código fuente para la instalación se ejecuta el comando `git clone https://opendev.org/openstack/devstack`, el cual realiza una clonación del Código desde el link indicado, su proceso se muestra en la figura 46.

Figura 46 Código Fuente.

```
stack@openstack:~$ git clone https://opendev.org/openstack/devstack
Cloning into 'devstack'...
remote: Enumerating objects: 47298, done.
remote: Counting objects: 100% (47298/47298), done.
remote: Compressing objects: 100% (21755/21755), done.
remote: Total 47298 (delta 33444), reused 38236 (delta 24828)
Receiving objects: 100% (47298/47298), 9.74 MiB | 4.99 MiB/s, done.
Resolving deltas: 100% (33444/33444), done.
stack@openstack:~$
```

Una vez obtenido el código fuente se debe establecer el Branch o versión que se desea utilizar para la instalación, como se muestra en la figura 47, para la instalación se utilizará la versión WALLABY, utilizando el comando `git checkout stable/wallaby`, observando un cambio en el Branch para la instalación

Figura 47 Selección Branch,

```
stack@openstack:~/devstack$ git checkout stable/wallaby
Branch 'stable/wallaby' set up to track remote branch 'stable/wallaby' from 'origin'.
Switched to a new branch 'stable/wallaby'
stack@openstack:~/devstack$
```

Archivo Autoconfiguración

Con el objetivo de simplificar la instalación de OpenStack, se crea un archivo llamado `local.conf`, este permite automatizar y asignar una contraseña especificada con la variable `ADMIN_PASSWORD` con un valor de `secret`, este valor puede ser reemplazado por el que el instalador prefiera, a su vez en el archivo mostrado en la figura

48, se especifica el valor de `ADMIN_PASSWORD` al resto de valores, indicando que su valor será el mismo para todos los servicios.

Figura 48 Fichero `local.conf`.

```

1  [[local|localrc]]
2  ADMIN_PASSWORD=secret
3  DATABASE_PASSWORD=$ADMIN_PASSWORD
4  RABBIT_PASSWORD=$ADMIN_PASSWORD
5  SERVICE_PASSWORD=$ADMIN_PASSWORD

```

Instalación.

Cabe recalcar que este procedimiento cambia las características y comportamiento de sistema por lo que se debe realizar en un sistema destinado para este propósito. Para realizar la instalación se ingresa al directorio Devstack, como se muestra en la figura 49, mediante el comando `cd devstack/`.

Figura 49 Directorio Devstack.

```

stack@openstack:~$ ls
devstack
stack@openstack:~$ cd devstack/
stack@openstack:~/devstack$ ls
CONTRIBUTING.rst  clean.sh  functions-common  playbooks  stack.sh
FUTURE.rst        data     gate              roles      stackrc
HACKING.rst       doc     inc              run_tests.sh  tests
LICENSE           extras.d lib             samples    tools
Makefile          files   local.conf       setup.cfg  tox.ini
README.rst        functions openrc          setup.py   unstack.sh
stack@openstack:~/devstack$

```

Una vez en la carpeta es necesario ejecutar el fichero `stack.sh`, este no requiere ser ejecutado de manera privilegiada ya que estos fueron establecidos en un paso anterior, este realizará la instalación de OpenStack en el sistema de manera automática, como se muestra en la figura 50.

Figura 50 Instalación OpenStack,

```

stack@openstack:~/devstack$ ./stack.sh
+ unset GREP_OPTIONS
+ unset LANG
+ unset LANGUAGE
+ LC_ALL=en_US.utf8
./stack.sh: line 60: warning: setlocale: LC_ALL: cannot change locale (e
n_US.utf8): No such file or directory
+ export LC_ALL
++ env
++ grep -E '^OS '
++ cut -d = -f 1
+ unset
+ umask 022
+ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr
/games:/usr/local/games:/snap/bin:/usr/local/sbin:/usr/sbin:/sbin
+++ dirname ./stack.sh
++ cd .
++ pwd
+ TOP_DIR=/opt/stack/devstack
+ NOUNSET=
+ [[ -n '' ]]
++ date +%s
+ DEVSTACK_START_TIME=1624653474
+ [[ -r /opt/stack/devstack/.stackenv ]]

```

Durante este proceso se requiere acceso a internet debido a que es necesario descargar múltiples repositorios de cada uno de sus módulos para su compilación, El tiempo estimado de instalación es de aproximadamente 45min, al culminar se mostrará un mensaje en la terminal indicando la versión que ha sido instalada, así como la ip a la cual responderán los servicios de OpenStack, juntamente con los usuarios y contraseñas para su administración como se muestra en la figura 51.

Figura 51 Instalación Completa.

```
=====
DevStack Component Timing
(times are in seconds)
=====
wait_for_service      23
pip_install           488
apt-get               45
run_process           184
dbsync                46
git_timed             574
apt-get-update        1
test_with_retry       3
async_wait            0
osc                   770
-----
Unaccounted time      1067
=====
Total runtime         3201

This is your host IP address: 192.168.1.40
This is your host IPv6 address: fe80::20c:29ff:fe25:a9fd
Horizon is now available at http://192.168.1.40/dashboard
Keystone is serving at http://192.168.1.40/identity/
The default users are: admin and demo
The password: ██████████

Services are running under systemd unit files.
For more information see:
https://docs.openstack.org/devstack/latest/systemd.html

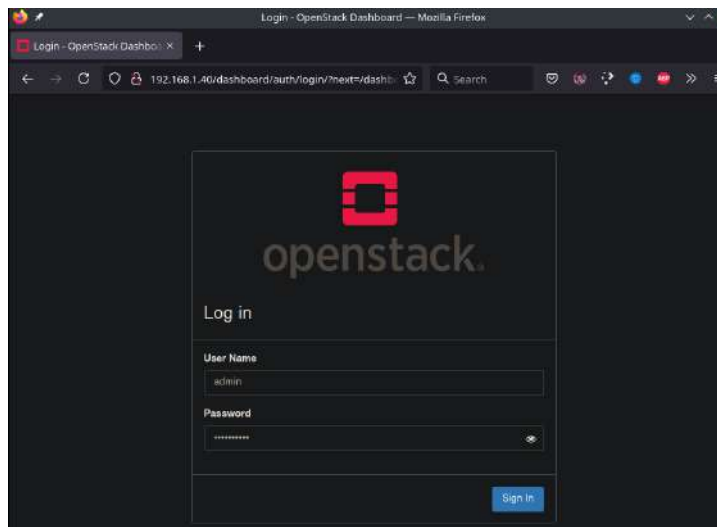
DevStack Version: wallaby
Change: ee8a227b620d2ef8b79e4b775ccc2521634a802e Move verify
OS Version: Ubuntu 20.04 focal

stack@openstack:~/devstack$ █
```

Gestión OpenStack

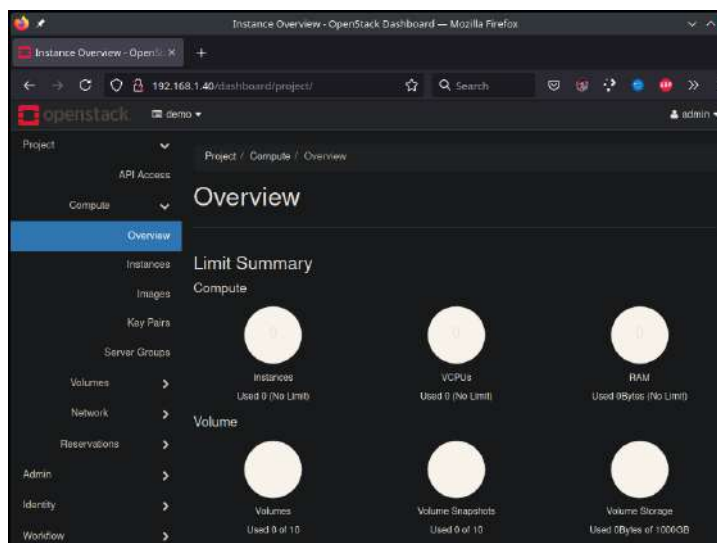
Una vez que se ha realizado la instalación es posible acceder a OpenStack mediante la interfaz gráfica, ingresando mediante la ip asignada: **192.168.1.40** obteniendo la página de login mostrada en la figura 52

Figura 52 Pantalla de Login OpenStack.



Para acceder a la interfaz de Openstack, se debe ingresar con los datos de usuario y contraseña que fueron indicados en la instalación para realizar las configuraciones pertinentes, como se muestra en la figura 53

Figura 53 Dashboard OpenStack.

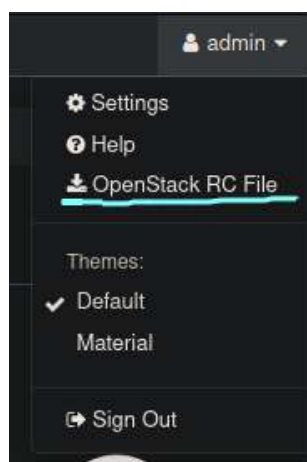


La figura 53 muestra todos los parámetros de OpenStack, su rendimiento, utilización de recursos, así también las configuraciones de nodos, red, imágenes e instancias activas.

Autenticación.

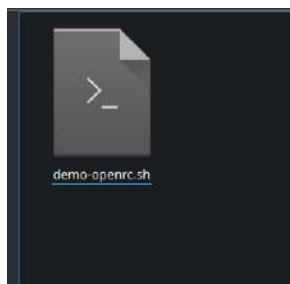
Para poder autenticarse y continuar con la configuración es necesario obtener el token de autenticación de OpenStack, para ello, en el menú del usuario se despliega la opción de “**OpenStack RC file**”, donde nos permite descargar el fichero, para poder ser ejecutado desde el computador del cual se está accediendo, de esta manera se puede desarrollar cambios dentro de OpenStack, como se muestra en la figura 54

Figura 54 Archivo Token.



Para descargar el fichero es necesario seleccionar la pestaña correspondiente en el menú de usuario, de esta manera es posible descargarlo en la computadora desde la cual se está accediendo, mostrando el fichero descargado en la figura 55

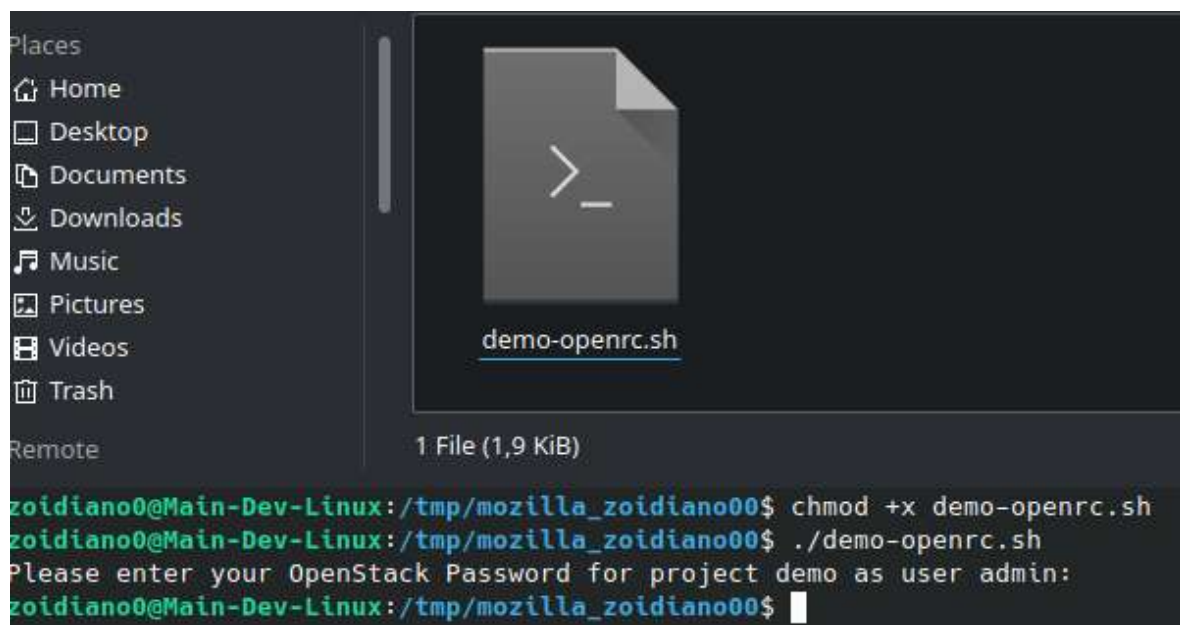
Figura 55 Documento Descargado.



Una vez descargado el fichero debe establecerse como ejecutable mediante `chmod +x <fichero>`, esto permite que pueda ser ejecutado en la terminal, donde se solicitará ingresar

las credenciales para autenticar el equipo como se aprecia en la figura 56. Una vez completado esta acción OpenStack aceptará cambios y configuraciones realizadas desde el host autenticado mediante el navegador y el fichero de token.

Figura 56 Ejecución fichero.



1.22.3 Generación Imagen de Hosts.

En esta sección se generará la imagen de sistema a utilizar dentro de la infraestructura de OpenStack para cada uno de los nodos, la cual comprende el host utilizado por los estudiantes para realizar las practicas. Este sistema como se indica en el capítulo anterior será basado en ubuntu, el cual será implementado como una imagen estandarizada para todos los hosts, teniendo las mismas capacidades y funciones entre las cuales notamos herramientas y servicios que permiten a los estudiantes poder realizar prácticas de tecnologías SDN y NFV.

Obtención Imagen BASE

En la generación es necesario que la imagen base del sistema operativo a utilizar para el proyecto sea ubuntu. Como primer paso se requiere obtener el instalador del sistema utilizando el comando `wget` que descargará el fichero `.iso` del repositorio oficial. Así como se muestra en la ilustración 57

wget <http://archive.ubuntu.com/ubuntu/dists/focal/main/installer-amd64/current/legacy-images/netboot/mini.iso>

Figura 57 Descarga Imagen Base

```
zoidiano0@Main-Dev-Linux:~/Development/BASE/iso$ wget http://archive.ubuntu.com/ubuntu/dist
--2021-06-30 15:57:24-- http://archive.ubuntu.com/ubuntu/dists/focal/main/installer-amd64/
Resolving archive.ubuntu.com (archive.ubuntu.com)... 91.189.88.142, 91.189.88.152, 2001:67c
Connecting to archive.ubuntu.com (archive.ubuntu.com)|91.189.88.142|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 77594624 (74M) [application/x-iso9660-image]
Saving to: 'mini.iso'

mini.iso                               100%[=====
2021-06-30 15:57:30 (14,6 MB/s) - 'mini.iso' saved [77594624/77594624]

zoidiano0@Main-Dev-Linux:~/Development/BASE/iso$ |
```

Una vez descargada la imagen es necesario asignar derechos de ejecución y derechos de gestión, con este propósito se utiliza el comando *chown* indicando el usuario y grupo que tiene los derechos *libvirt-qemu:kvm* que serán aplicados en el fichero *mini.iso*, los cambios son apreciables en la figura 58.

```
chown libvirt-qemu:kvm mini.iso
```

Figura 58 Cambio de derechos

```
zoidiano0@Main-Dev-Linux:~/Development/BASE/iso$ ls -l
total 75776
-rw-rw-r-- 1 libvirt-qemu kvm 77594624 abr 21 2020 mini.iso
zoidiano0@Main-Dev-Linux:~/Development/BASE/iso$ |
```

Generación Disco Base QEMU

En este paso se generará el disco virtual donde será instalado y configurado el sistema base, generado en un formato el cual OpenStack pueda reconocer y utilizar como imagen para sus nodos, su formato es *qcow2*, del sistema de emulación *Qemu*, este formato permite ser ampliado y modificado posteriormente de ser necesario y utiliza una compresión, encriptación por defecto y permite la creación de snapshots para guardar trabajos en proceso. Para generarlo se utiliza el comando *qemu-img*, que al utilizar la

opción `create` y la bandera `-f`, misma que permite especificar `qcow2` como extensión del fichero, además se establece su nombre escribiéndolo a continuación `base.qcow2`, y se termina el comando especificando el tamaño del disco en Gigabytes.

```
qemu-img create -f qcow2 base.qcow2 10G
```

Perfil KVM

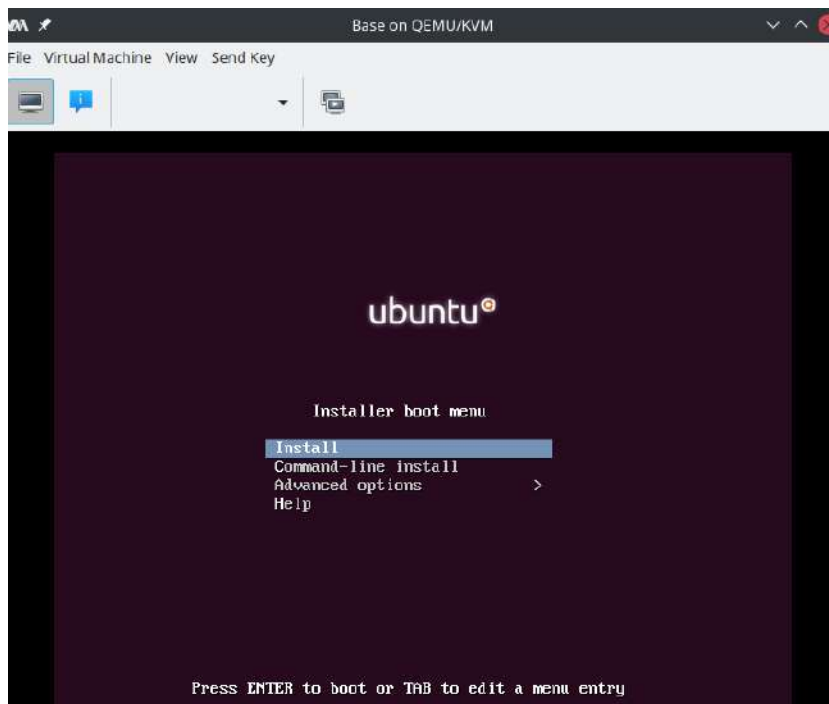
El perfil KVM permite establecer las características de la máquina virtual sobre la cual se generará la imagen del host a ser implementado, para ello se utiliza el comando `virt-install`, este permite mediante sus opciones establecer las prestaciones utilizando las opciones como el tipo de máquina virtual mediante `--virt-type kvm`, el nombre de la máquina virtual `--name Base`, la cantidad de ram `--ram 1024`, la imagen de cd a utilizar `--cdrom=iso/mini.iso`, el disco a utilizar `-disk image/base.qcow2, bus=virtio, size=15`, además de indicar el formato del mismo `format=qcow2`, la conexión a la red `--network network=default`, el tipo de visualización gráfica `--graphics vnc, listen=0.0.0.0` y opciones de sistema como son `--noautoconsole` `--os-type=Linux`, así como el sistema operativo a utilizar `--os-variant=ubuntu20.04`.

```
virt-install --virt-type kvm --name Base --ram 1024 --cdrom=iso/mini.iso -disk image/base.qcow2, bus=virtio, size=15, format=qcow2 --network network=default --graphics vnc, listen=0.0.0.0 --noautoconsole --os-type=Linux --os-variant=ubuntu20.04
```

Instalación

Una vez generado el perfil y encendiendo la máquina KVM, se obtiene una ventana como la indicada en la figura 60, dando acceso a la máquina virtual. La continuación de la instalación de la imagen de nodo se encuentra detallada en el **Anexo 2**

Figura 59 KVM Fuente: Autor



1.22.4 Instalación Servicios y Herramientas SDN y NFV.

En esta sección se realizará la instalación y configuración de los diferentes servicios y herramientas que permiten que la imagen de sistema generada en el punto 3.3.3, donde es posible realizar prácticas sobre las tecnologías SDN y NFV, de esta manera cumpliendo con las funciones indicadas en la arquitectura del proyecto. Se debe recalcar que en este punto se indica la instalación, mientras que su utilización estará demostrada en el capítulo 4.

Como una primera configuración y buena práctica se realizará la actualización del sistema por completo, como se muestra en la figura 61, de manera que se trabaje en la versión más actual, evitando problemas de compatibilidad y seguridad.

Figura 60 Actualización del sistema.

```

estudiante@Hanstap:~$ sudo apt update && sudo aptu upgrade -y
[sudo] contraseña para estudiante:
Des:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Obj:2 http://ec.archive.ubuntu.com/ubuntu focal InRelease
Des:3 http://ec.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Des:4 http://security.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadata
 [7.956 B]
Des:5 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 c-n-f Me
tadata [456 B]
Des:6 http://security.ubuntu.com/ubuntu focal-security/universe amd64 c-n-f Meta
data [11,6 kB]
Des:7 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 c-n-f Me
tadata [528 B]
Des:8 http://ec.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
0% [Trabajando]

```

Instalación OpenVswitch

El software openVswitch es un switch virtual que puede ser utilizado en implementaciones SDN como parte del plano de datos en una red SDN, así como puede ser utilizado en implementaciones NFV, al permitir su funcionamiento configurable por el usuario como una instancia NFV Intel, 2020). La instalación de este componente se realizará mediante código fuente, obteniendo de esta manera la versión con las características más actuales, y los parches de seguridad.

Para esto se ejecutan los pasos indicados en la página web del proyecto.

<https://docs.openvswitch.org/en/latest/intro/install/general/>

Para iniciar la instalación se descarga el código fuente con el comando.

```
git clone https://github.com/openvswitch/ovs.git
```

en una carpeta del sistema como se indica en la figura 62

Figura 61 Obtención Código.

```

estudiante@Hanstap:~/Dev$ git clone https://github.com/openvswitch/ovs.git
Clonando en 'ovs'...
remote: Enumerating objects: 157761, done.
remote: Counting objects: 100% (83/83), done.
remote: Compressing objects: 100% (55/55), done.
remote: Total 157761 (delta 53), reused 47 (delta 28), pack-reused 157678
Recibiendo objetos: 100% (157761/157761), 101.56 MiB | 23.18 MiB/s, listo.
Resolviendo deltas: 100% (125263/125263), listo.
estudiante@Hanstap:~/Dev$

```

Se ejecuta la secuencia de inicialización y compilación del código como indica en la página web, mostrada en la figura 63 y 64, ejecutando el fichero `boot.sh`

Figura 62 Inicialización código.

```

estudiante@Hanstap:~/Dev/ovs$ ./boot.sh
libtoolize: putting auxiliary files in AC_CONFIG_AUX_DIR, 'build-aux'.
libtoolize: copying file 'build-aux/ltmain.sh'
libtoolize: putting macros in AC_CONFIG_MACRO_DIRS, 'm4'.
libtoolize: copying file 'm4/libtool.m4'
libtoolize: copying file 'm4/ltoptions.m4'
libtoolize: copying file 'm4/ltsugar.m4'
libtoolize: copying file 'm4/ltversion.m4'
libtoolize: copying file 'm4/lt-obsolete.m4'
configure.ac:24: installing 'build-aux/compile'
configure.ac:39: installing 'build-aux/config.guess'
configure.ac:39: installing 'build-aux/config.sub'
configure.ac:22: installing 'build-aux/install-sh'
configure.ac:22: installing 'build-aux/missing'
Makefile.am: installing 'build-aux/depcomp'
estudiante@Hanstap:~/Dev/ovs$

```

Se realiza la comprobación del código y de la compatibilidad de sistema, mediante la ejecución con el comando `./configure`

Figura 63 Configuración del Compilador.

```

estudiante@Hanstap:~/Dev/ovs$ ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking how to create a pax tar archive... gnutar
checking whether make supports the include directive... yes (GNU style)
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes

```

Se realiza la configuración del código fuente, y a continuación la compilación con el comando `make -j2` como se muestra en la figura 65

Figura 64 Compilación Completa.

```

estudiante@Hanstap:~/Dev/ovs$ make -j2
make all-recursive
make[1]: se entra en el directorio '/home/estudiante/Dev/ovs'
Making all in datapath
make[2]: se entra en el directorio '/home/estudiante/Dev/ovs/datapath'
make[3]: se entra en el directorio '/home/estudiante/Dev/ovs/datapath'
make[3]: se sale del directorio '/home/estudiante/Dev/ovs/datapath'
make[2]: se sale del directorio '/home/estudiante/Dev/ovs/datapath'
make[2]: se entra en el directorio '/home/estudiante/Dev/ovs'
make[3]: se entra en el directorio '/home/estudiante/Dev/ovs/datapath'
make[3]: 'distfiles' está actualizado.
make[3]: se sale del directorio '/home/estudiante/Dev/ovs/datapath'
make[2]: se sale del directorio '/home/estudiante/Dev/ovs'
make[1]: se sale del directorio '/home/estudiante/Dev/ovs'
estudiante@Hanstap:~/Dev/ovs$

```

Una vez que la compilación ha terminado se realiza la instalación de los binarios del sistema, así como también los módulos de kernel que son requeridos para una integración completa con el sistema, mostrados en la figura 66, mediante el comando “*sudo make install*”

Figura 65 Instalación de binarios.

```

estudiante@Hanstap:~/Dev/ovs$ sudo make install
[sudo] contraseña para estudiante:
make install-recursive
make[1]: se entra en el directorio '/home/estudiante/Dev/ovs'
Making install in datapath
make[2]: se entra en el directorio '/home/estudiante/Dev/ovs/datapath'
make[3]: se entra en el directorio '/home/estudiante/Dev/ovs/datapath'
make[4]: se entra en el directorio '/home/estudiante/Dev/ovs/datapath'
make[4]: No se hace nada para 'install-exec-am'.
make[4]: No se hace nada para 'install-data-am'.
make[4]: se sale del directorio '/home/estudiante/Dev/ovs/datapath'
make[3]: se sale del directorio '/home/estudiante/Dev/ovs/datapath'
make[2]: se sale del directorio '/home/estudiante/Dev/ovs/datapath'
make[2]: se entra en el directorio '/home/estudiante/Dev/ovs'
make[3]: se entra en el directorio '/home/estudiante/Dev/ovs/datapath'

```

Inicialización OVS

Una vez que se ha realizado la instalación de los módulos dentro del SO, es posible ejecutar a los binarios para inicializar el servicio dentro del sistema, para ello se agrega al PATH de sistema la carpeta donde todos los binarios están alojados, mediante

el comando, `export PATH=$PATH:/usr/local/share/openvswitch/scripts`, de esta manera los binarios serán accesibles para el usuario y se podrá ejecutar las configuraciones, permitiendo que la controladora SDN realice llamadas de creación de instancias de OVS para las topologías totalmente separadas para una mejor administración.

Para inicializar el servicio se utiliza el comando `ovs-ctl start`, comando que inicializa el servicio del sistema a la espera de instrucciones, se verifica el estado del servicio con el comando `ovs-ctl status`, obteniendo un resultado como se muestra en la figura 67

Figura 66 Estado OVS.

```
estudiante@Hanstap:~$ ovs-ctl status
ovsdb-server is running with pid 2213
ovs-vswitchd is running with pid 2237
estudiante@Hanstap:~$
```

De esta manera se establece el servicio de OVS para el SO que será utilizado por el estudiante para desarrollar prácticas SDN y NFV, comprobando su funcionamiento con una prueba donde se observará la creación de interfaces de red virtuales pertenecientes a las instancias generadas por mininet, verificando el estado de su servicio del sistema como se indica en la figura 68.

Figura 67 Servicio OVS Activo.

```
estudiante@Hanstap:~$ systemctl status openvswitch-switch.service
● openvswitch-switch.service - Open vSwitch
   Loaded: loaded (/lib/systemd/system/openvswitch-switch.service; enabled;
   Active: active (exited) since Tue 2021-08-10 11:12:57 -05; 42s ago
   Process: 929 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 929 (code=exited, status=0/SUCCESS)

ago 10 11:12:57 Hanstap systemd[1]: Starting Open vSwitch...
ago 10 11:12:57 Hanstap systemd[1]: Finished Open vSwitch.
```

Acto seguido se verifica las interfaces existentes en el sistema previas a la creación de interfaces virtuales, esto es verificado utilizando el comando `ifconfig` que lista las interfaces existentes en el equipo como se aprecia en la figura 69.

Figura 68 Interfaces Básicas.

```

estudiante@Hanstap:~$ ifconfig
enp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.122.114 netmask 255.255.255.0 broadcast 192.168.122.255
    inet6 fe80::5054:ff:fee7:52ea prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:e7:52:ea txqueuelen 1000 (Ethernet)
    RX packets 807 bytes 919099 (919.0 KB)
    RX errors 0 dropped 139 overruns 0 frame 0
    TX packets 271 bytes 23307 (23.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Bucle local)
    RX packets 205 bytes 14974 (14.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 205 bytes 14974 (14.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

estudiante@Hanstap:~$

```

En la figura 69 es posible apreciar las 2 únicas interfaces existentes en el sistema, mientras que en la figura 70 se muestran las nuevas interfaces creadas por OVS

Figura 69 Interfaces virtuales OVS.

```

estudiante@Hanstap:~$ ifconfig
enp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.122.114 netmask 255.255.255.0 broadcast 192.168.122.255
    inet6 fe80::5054:ff:fee7:52ea prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:e7:52:ea txqueuelen 1000 (Ethernet)
    RX packets 1667 bytes 975798 (975.7 KB)
    RX errors 0 dropped 964 overruns 0 frame 0
    TX packets 304 bytes 26774 (26.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Bucle local)
    RX packets 641 bytes 37548 (37.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 641 bytes 37548 (37.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::885d:40ff:fef1:2811 prefixlen 64 scopeid 0x20<link>
    ether 8a:5d:40:f1:28:11 txqueuelen 1000 (Ethernet)
    RX packets 7 bytes 586 (586.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 15 bytes 2168 (2.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::bc24:ffff:fe17:7fb0 prefixlen 64 scopeid 0x20<link>
    ether be:24:ff:17:7f:b0 txqueuelen 1000 (Ethernet)
    RX packets 7 bytes 586 (586.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 15 bytes 2168 (2.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

estudiante@Hanstap:~$

```

Instalación Controladora SDN

La Controladora SDN ONOS estará a cargo de la administración de las topologías , siendo esta la controladora de las redes en el plano de control y utilizando instancias de OVS dando instrucciones de gestión y administración de la red en su plano de datos. Para la instalación de ONOS se requiere además sus dependencias con el siguiente comando.

```
Sudo apt install curl wget git openjdk-11-jdk
```

Una vez instaladas estas dependencias es necesario establecer los directorios por defecto de Java, editando el fichero ubicado en `/etc/environment` agregando las siguientes líneas al final del archivo.

```
JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
JRE_HOME=/usr/lib/jvm/java-11-openjdk-amd64/jre
```

Seguido se accede a la carpeta `/opt` donde se realiza la instalación de ONOS, descargando el fichero con el comando.

```
wget https://repo1.maven.org/maven2/org/onosproject/onos-releases/2.6.0/onos-2.6.0.tar.gz
```

El mismo que debe ser extraído, una vez realizado, se accede a la carpeta y observaremos los ficheros a ser compilados, como se muestra en la figura 71.

Figura 70 Ficheros ONOS,

```
estudiante@Hanstap:/opt/onos-2.6.0$ ls
apache-karaf-4.2.9  apps  bin  lib  VERSION
estudiante@Hanstap:/opt/onos-2.6.0$
```

A continuación, se iniciará el servicio de ONOS ingresando a la carpeta `bin` y ejecutando el siguiente comando, tal como se muestra en la figura 72 obteniendo el arranque de la controladora utilizando el comando: `./onos-service start`. Una vez que el servicio ha iniciado se comprueba su funcionamiento observando los puertos abiertos como se muestra en la figura 73.

Figura 71 Arranque Servicio ONOS.

```
estudiante@Hanstap:/opt/onos/bin$ sudo ./onos-service start
jul. 25, 2021 4:30:54 P. M. org.apache.karaf.main.Main launch
INFORMACIÓN: Installing and starting initial bundles
jul. 25, 2021 4:30:54 P. M. org.apache.karaf.main.Main launch
INFORMACIÓN: All initial bundles installed and set to start
jul. 25, 2021 4:30:54 P. M. org.apache.karaf.main.lock.SimpleFileLock lock
INFORMACIÓN: Trying to lock /opt/onos/apache-karaf-4.2.9/lock
jul. 25, 2021 4:30:54 P. M. org.apache.karaf.main.lock.SimpleFileLock lock
INFORMACIÓN: Lock acquired
jul. 25, 2021 4:30:54 P. M. org.apache.karaf.main.Main$KarafLockCallback lockAcquired
INFORMACIÓN: Lock acquired. Setting startlevel to 100
```

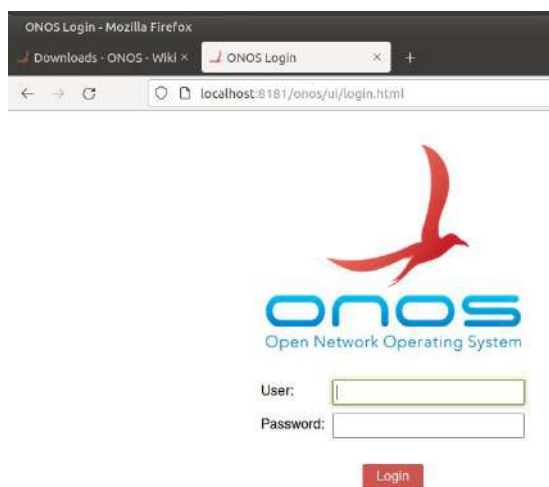
Figura 72 Puertos de escucha del Host.

```
estudiante@Hanstap:~$ sudo netstat -tulpn | grep ESCUCHAR
tcp        0      0 127.0.0.53:53          0.0.0.0:*           ESCUCHAR    564/systemd-resolve
tcp        0      0 0.0.0.0:22            0.0.0.0:*           ESCUCHAR    936/sshd: /usr/sbin
tcp        0      0 127.0.0.1:631         0.0.0.0:*           ESCUCHAR    615/cupsd
tcp6       0      0 :::1099               :::*                 ESCUCHAR    2433/java
tcp6       0      0 127.0.0.1:42383      :::*                 ESCUCHAR    2433/java
tcp6       0      0 :::9876               :::*                 ESCUCHAR    2433/java
tcp6       0      0 :::8181               :::*                 ESCUCHAR    2433/java
tcp6       0      0 :::22                 :::*                 ESCUCHAR    936/sshd: /usr/sbin
tcp6       0      0 :::1631               :::*                 ESCUCHAR    615/cupsd
tcp6       0      0 :::8101               :::*                 ESCUCHAR    2433/java
tcp6       0      0 :::44549              :::*                 ESCUCHAR    2433/java
estudiante@Hanstap:~$
```

En la figura 73 se muestran los puertos de sistema que se encuentran escuchando por una conexión, entre ellos se encuentran los puertos utilizados por la controladora ONOS, los mismos que son listados a continuación, permitiendo observar el funcionamiento de la controladora ONOS en el sistema, confirmando mediante el acceso web en la dirección `http://localhost/onos/ui`. tal como se muestra en la figura 74, la cual se ingresa mediante las credenciales por defecto: User: onos Password: rocks

- 8181 Configuraciones REST y acceso API vía GUI
 - 8101 Acceso a La terminal CLI de ONOS
 - 9876 Comunicación de Clúster ONOS
 - 6653 Puerto OpenFlow
 - 6640 Puerto OpenvSwitch

Figura 73 Interfaz Web ONOS.



Para obtener un funcionamiento correcto de la controladora con el resto de los servicios y softwares a instalar, se requiere la instalación de ciertas aplicaciones internas de la controladora, esto se realiza mediante la terminal *cli* de la controladora, la cual es accesible mediante una conexión hacia el puerto 8101, utilizando las credenciales por defecto, es decir *User: karaf* y *Password: karaf* de manera que se obtiene acceso hacia la terminal de comandos de la controladora mostrada en la figura 75, una vez en la terminal se realiza la instalación de cada aplicación individualmente con los comandos indicados a continuación, su instalación será confirmada tal y como se muestra en la figura 76.

Figura 74 ONOS CLI,

```

estudiante@Hanstap:~$ ssh karaf@localhost -p 8101
The authenticity of host '[localhost]:8101 ([::1]:8101)' can't be established.
RSA key fingerprint is SHA256:QHNN7oSjELytn5m/PQpidqkjDQcB30BS9H3UT9MDq9A.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:8101' (RSA) to the list of known hosts.
Password authentication
Password:
Welcome to Open Network Operating System (ONOS)!

  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.

karaf@root >

```

Aplicaciones.

```

app activate org.onosproject.openflow # Permite La comunicación mediante el
                                     protocolo Openflow
app activate org.onosproject.openflow-message # Permite el envío de mensajes mediante
                                               el protocolo Openflow
app activate org.onosproject.fwd # Permite el reenvío de paquetes en la controladora
                                  SDN
app activate org.onosproject.ofagent # Establece un agente monitor del protocolo
                                     Openflow

```

*app activate org.onosproject.workflow #Permite la activación de menús y diálogos para
La integración de Openflow*

app activate org.onosproject.ovsdb #Permite la conectividad con OpenvSwitch

*app activate org.onosproject.drivers.ovsdb #Establece un agente de trabajo con
OpenvSwitch*

*app activate org.onosproject.mfwd #Permite el reenvío de paquetes entre módulos de la
controladora SDN*

*app activate org.onosproject.workflow.ofoverlay #Presenta estadísticas del protocolo
de Openflow*

Figura 75 Activación Aplicaciones ONOS,

```

karaf@root > app activate org.onosproject.openflow
Activated org.onosproject.openflow
karaf@root > app activate org.onosproject.openflow-message
> app activate org.onosproject.fwd
> app activate org.onosproject.ofagent
> app activate org.onosproject.workflow
> app activate org.onosproject.ovsdb
> app activate org.onosproject.drivers.ovsdb
> app activate org.onosproject.mfwd
> app activate org.onosproject.workflow.ofover
Activated org.onosproject.openflow-message
Activated org.onosproject.fwd
Activated org.onosproject.ofagent
Activated org.onosproject.workflow
Activated org.onosproject.ovsdb
Activated org.onosproject.drivers.ovsdb
Activated org.onosproject.mfwd
No such application: org.onosproject.workflow.ofover
karaf@root > app activate org.onosproject.workflow.ofoverlay
Activated org.onosproject.workflow.ofoverlay
karaf@root >

```

Para comprobar el funcionamiento de la controladora se utiliza mininet, programa que fue previamente instalado. Para la ejecución de una topología básica que permita observar el funcionamiento y compatibilidad del software, se utiliza el siguiente comando:

```
mn -topo single,3 -mac -switch ovsk -controller remote,ip=localhost
```

Del comando anterior, el parámetro “-topo single,3” permite generar una topología de un solo Switch al cual están conectados 3 host. El parámetro “-switch ovsk”

permite utilizar *ovs* como switch en la topología, y finalmente el parámetro “*-controller remote, ip=localhost*” permite que se conecta a la controladora ubicada en localhost

Instalación Containernet

Containernet es una aplicación que permite generar topologías de redes SDN con aplicativos NFV de manera rápida, permitiendo utilizar como controladora el software ONOS y el software OVS para implementaciones SDN como parte del plano de datos, así como una instancia NFV de ser requerido. Para la instalación, se seguirá los pasos indicados en la documentación en línea

<https://github.com/containernet/containernet>

Donde se obtendrá los comandos de instalación, estos son desarrollados a continuación, indicando cada paso iniciando desde la instalación de los requerimientos base mediante el comando `sudo apt-get install ansible git aptitude`, del cual su resultado se presenta en la figura 77, acto seguido se obtiene el repositorio del software, como se muestra en la figura 78.

Figura 76 Instalación Requerimientos Containernet. Fuente: Autor

```
estudiante@Hanstap:~$ sudo apt install ansible git aptitude
[sudo] contraseña para estudiante:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
git ya está en su versión más reciente (1:2.25.1-1ubuntu3.1).
Se instalarán los siguientes paquetes adicionales:
  aptitude-common ieee-data libboost-iostreams1.71.0 libcbi-fast-perl
  libcgi-pm-perl libclass-accessor-perl libcwidget4 libfcgi-perl
  libparse-debianchange-log-perl python3-argcomplete python3-dnspython
  python3-jmespath python3-kerberos python3-libcloud python3-lockfile
  python3-netaddr python3-ntlm-auth python3-requests-kerberos
  python3-requests-ntlm python3-selinux python3-winrm python3-xmltodict
```

Figura 77 Obtención Repositorio Containernet.

```

estudiante@Hanstap:~$ git clone https://github.com/containernet/containernet.git
Clonando en 'containernet'...
remote: Enumerating objects: 10425, done.
remote: Counting objects: 100% (52/52), done.
remote: Compressing objects: 100% (42/42), done.
remote: Total 10425 (delta 21), reused 21 (delta 9), pack-reused 10373
Recibiendo objetos: 100% (10425/10425), 3.25 MiB | 5.50 MiB/s, listo.
Resolviendo deltas: 100% (6898/6898), listo.
estudiante@Hanstap:~$

```

Para la compilación del software ejecutamos el comando `sudo ansible-playbook -i "localhost," -c local install.yml` el cual configura el software al nuevo host, indicándolo mediante el argumento `-i "localhost," -c local install.yml` como se muestra en la figura 79, una vez terminada la compilación aparecerá un mensaje donde se indica que a instalación ha sido completada, como se aprecia en la figura 80

Figura 78 Compilación Containernet.

```

estudiante@Hanstap:~/containernet/ansible$ sudo ansible-playbook -i "localhost,"
-c local install.yml

PLAY [localhost] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [updates apt] *****

```

Figura 79 Compilación Containernet Completa.

```

localhost : ok=16  changed=14  unreachable=0  failed=0
skipped=0  rescued=0  ignored=0
estudiante@Hanstap:~/containernet/ansible$ █

```

Para realizar la instalación de los binarios compilados se ingresa el comando `sudo make develOp`, de esta manera es posible ejecutar comandos de forma global en el host.

La comprobación del funcionamiento del software se realiza ejecutando uno de los ejemplos desde la terminal, utilizando el comando `sudo python3 examples/containernet_example.py`, especificando el ejemplo mediante el argumento `examples/containernet_example.py` obteniendo lo mostrado en la figura 81 obtenido el funcionamiento de este componente del proyecto.

Figura 80 Prueba Containernet.

```

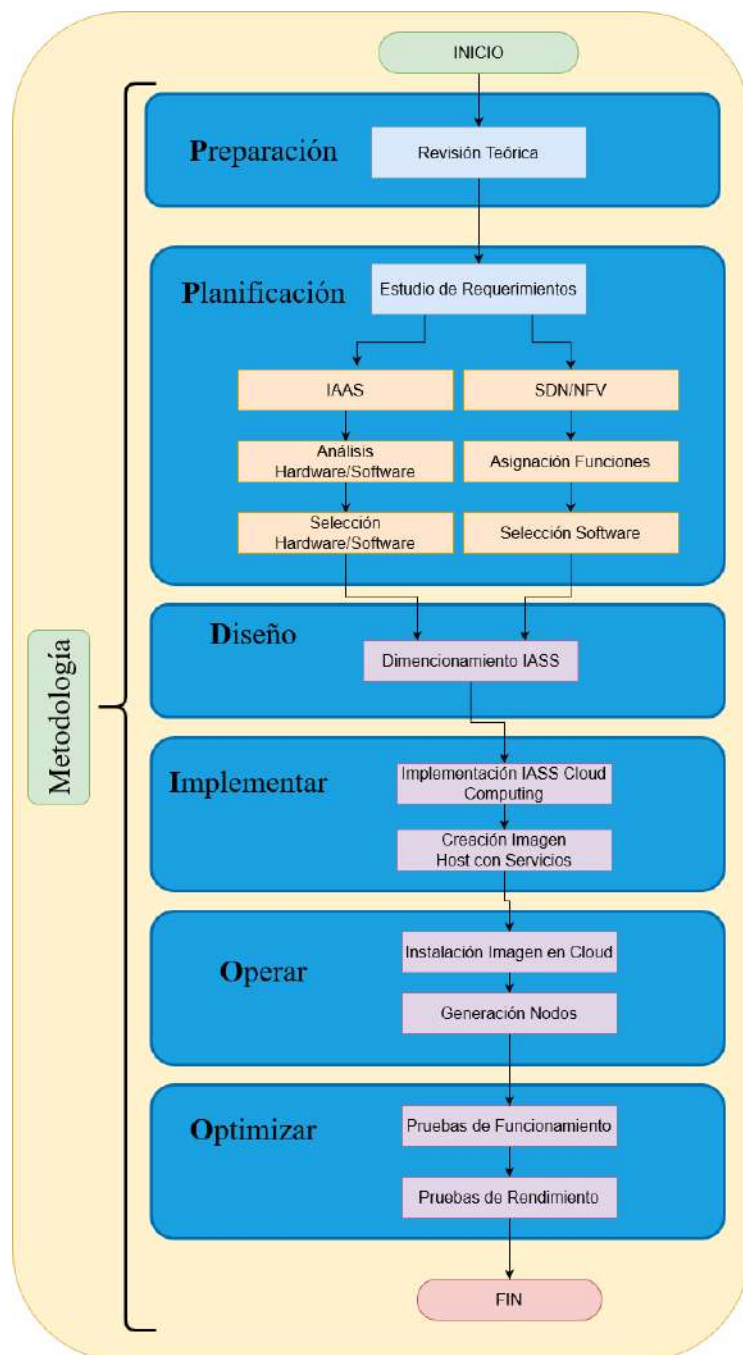
1:
d1: kwargs {'ip': '10.0.0.251'}
d1: update resources {'cpu_quota': -1}
1:
d2: kwargs {'ip': '10.0.0.252'}
d2: update resources {'cpu_quota': -1}
*** Adding switches
*** Creating links
(1.00Mbit 100ms delay) (1.00Mbit 100ms delay)
*** Configuring hosts
d1 d2
*** Starting controller
c0
*** Starting 2 switches
s1 (1.00Mbit 100ms delay) s2 (1.00Mbit 100ms delay)
*** Testing connectivity
d1 -> d2
d2 -> d1
*** Results: 0% dropped (2/2 received)
*** Running CLI
*** Starting CLI:
containernet> ping all

```

1.23 Guía de Proceso

Esta guía presenta el resumen de la implementación realizada y los pasos que se siguieron para el desarrollo del proyecto de una manera gráfica y clara del procedimiento, mostrado en la figura 82.

Figura 81 Diagrama de flujo estructura del proyecto.



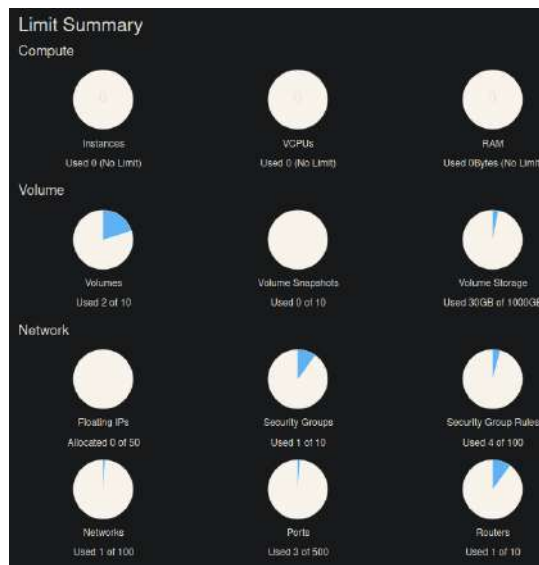
CAPITULO IV Operación y Resultados

EL presente capítulo se enfoca en la **Operación** y **Optimización** contemplados en la metodología propuesta para el desarrollo del presente trabajo, se presentará la funcionalidad de cada una de sus partes, incluyendo pruebas de funcionamiento individual y general, siendo estas basadas en las recomendaciones indicadas en el RFC 2544 (IETF, 2020), las cuales especifican, latencia, tramas erradas y velocidad de enlace.

1.24 Funcionamiento IAAS

Como primer punto se presenta la **Operación** de la IAAS establecida en el apartado 3.3.2 implementando la plataforma de Cloud Computing OpenStack, la misma que se utiliza como gestor de la infraestructura, así como de las instancias virtuales que se establecerán sobre la misma, de esta manera, se demuestra su funcionamiento accediendo a la dirección IP asignada *192.168.1.40* utilizando las credenciales respectivas, permitiendo observar los parámetros de monitoreo que cuenta por defecto esta plataforma como son el número de instancias en funcionamiento, observando la utilización de VCPU y RAM, volúmenes asignados, almacenamiento del sistema, así como información referente a las redes, todo esto indicado en la figura 82.

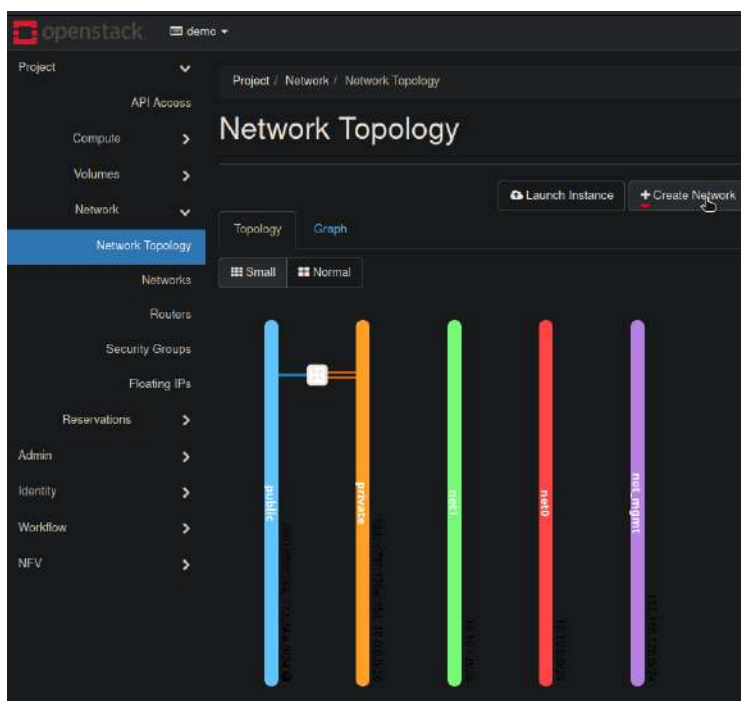
Figura 82 Monitoreo Básico OpenStack.



Redes SDN internas de la IAAS para nodos

La plataforma de OpenStack permite generar redes SDN, las cuáles sirven para la interconexión de las instancias virtuales y a su vez su conexión hacia el internet mediante las redes *public* y *private* (generadas por defecto), permitiendo separar instancias en segmentos de red individuales, de esta manera, tal como se muestra en la Figura 83, haciendo click sobre la opción *+Create network*, es posible generar las redes *net0* y *net1* mismas que servirán para separar 2 grupos de instancias, además se establece una red *mgmt*, misma que se utilizará, de ser requerida, para configuración y acceso administrativo desde la plataforma hacia la instancia virtual.

Figura 83 Infraestructura SDN Interna



Instalación Imagen de Host

Uno de los requerimientos del proyecto es la capacidad de poder desplegar instancias virtuales para que los estudiantes puedan desarrollar sus prácticas, para esto, se utiliza la imagen estandarizada generada en el punto 3.3.3, la cual fue establecida utilizando los requerimientos especificados en el punto 3.1.1 así como los servicios y

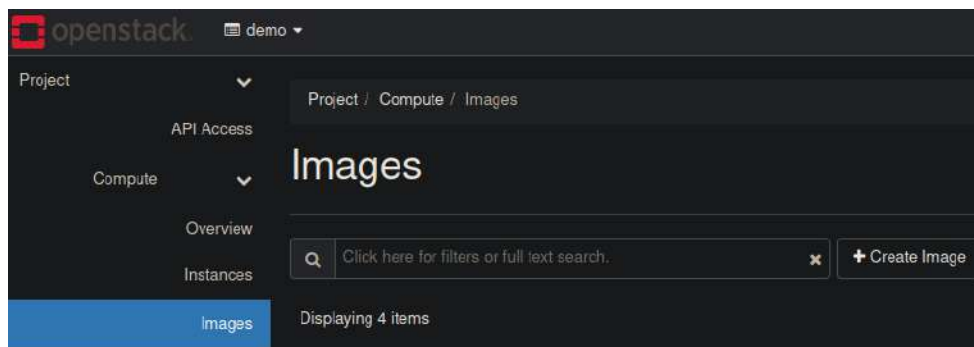
softwares habilitantes de tecnologías SDN y NFV en el punto 3.3.4 de esta manera fue posible obtener una imagen de sistema la cual cumple con las características requeridas para el proyecto indicadas en la tabla 11 presentada a continuación.

Tabla 11 Características imagen de instancia

Requerimientos	Valor
Sistema Operativo	Ubuntu
CPU (cores)	2
RAM (GB)	2
Almacenamiento (GB)	15
Controladora SDN	ONOS
Switich SDN/NFV	OVS
Software Topologías	Mininet

Para que la imagen pueda ser utilizada por OpenStack, esta debe ser instalada en la plataforma mediante la interfaz web ingresando a la sección de *Compute > Images*, mediante el botón de *create Image*, como se muestra en la figura 84

Figura 84 Gestor Volúmenes OpenStack

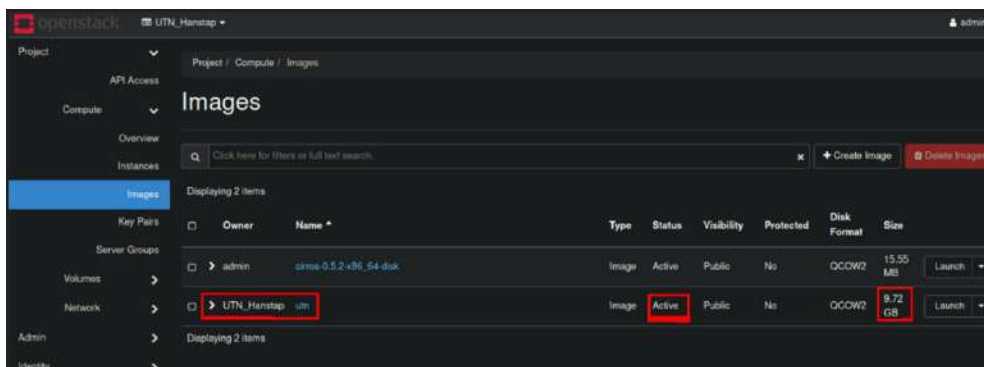


Con esto se obtiene un asistente en el proceso de instalación, se llena los campos indicados como son: *Nombre* y *Descripción*, tal y como se muestra en la figura 85, con los valores de *almacenamiento* y *memoria RAM* mínimas indicadas en la tabla 11.

Figura 85 Información Imagen.

Al completarse el proceso de instalación de la imagen, esta se muestra en la sección de *Images*, indicando su estado activo para ser utilizada, además se presenta su tamaño de imagen, esto es apreciable en la figura 86.

Figura 86 Imagen HOST Instalada.



Generación de instancias virtuales

Continuando con el funcionamiento de las partes de la infraestructura, se muestra la sección en la cual las instancias son creadas sobre la plataforma OpenStack, las mismas que serán asociadas hacia una de las redes indicadas en el punto 4.1.1 de manera que puedan tener acceso a internet, para esto, es necesario navegar hacia la pestaña *Instances*, donde se selecciona el botón de *Launch Instance*, posteriormente aparecerá un asistente en el cual como primer valor se debe ingresar los detalles de identificación y el número de instancias que se desea generar, en este caso será de 1, como se muestra en la figura 87.

Figura 87 Detalles Instancia.

Launch Instance

Details

Sources

Flavor *

Networks *

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Please provide the initial hostname for the instance, the availability zone where it is launched, and the count of instances to create. Increase the Count to create multiple instances with the same settings.

Project Name

Instance Name *

Description

Availability Zone

Count *

A continuación, se selecciona el *Flavor* (Características físicas) que se asignarán a las instancias para su funcionamiento, indicando los valores de *Memoria RAM*, *CPU* y *almacenamiento* indicados en la tabla 11 como se muestra en la ilustración 88, de la misma manera se establecerá la conexión con la red *public* para su conectividad, como se muestra en la figura 89

Figura 88 Características Físicas Instancia.

Flavors manage the sizing for the compute, memory and storage capacity of the instance.

Allocated

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
> UTN	4	2 GB	15 GB	15 GB	0 GB	Yes

Available 4/2

Click here for filters or full text search.

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
> m1.nano	1	128 MB	1 GB	1 GB	0 GB	Yes
> m1.micro	1	192 MB	1 GB	1 GB	0 GB	Yes
> cirros256	1	256 MB	1 GB	1 GB	0 GB	Yes
> m1.tiny	1	512 MB	1 GB	1 GB	0 GB	Yes
> ds512M	1	512 MB	5 GB	5 GB	0 GB	Yes

Figura 89 Conexión a red.

Launch Instance

Networks provide the communication channels for instances in the cloud.

Allocated 1 Select networks from those

Network	Subnets Associated	Shared	Admin State	Status
↕ 1 > public	ipv6-public-subnet public-subnet	Yes	Up	Active

Available 1 Select at least

Click here for filters or full text search.

Network	Subnets Associated	Shared	Admin State	Status
> shared	shared-subnet	Yes	Up	Active

Se selecciona la imagen que será utilizada en la instancia, indicada por el nombre *utn* como se muestra en la figura 90, una vez generada la instancia esta será automáticamente inicializada, la verificación de su funcionamiento es posible mediante la pestaña de instancias, como se muestra en la figura 91.

Figura 90 Selección Imágen.

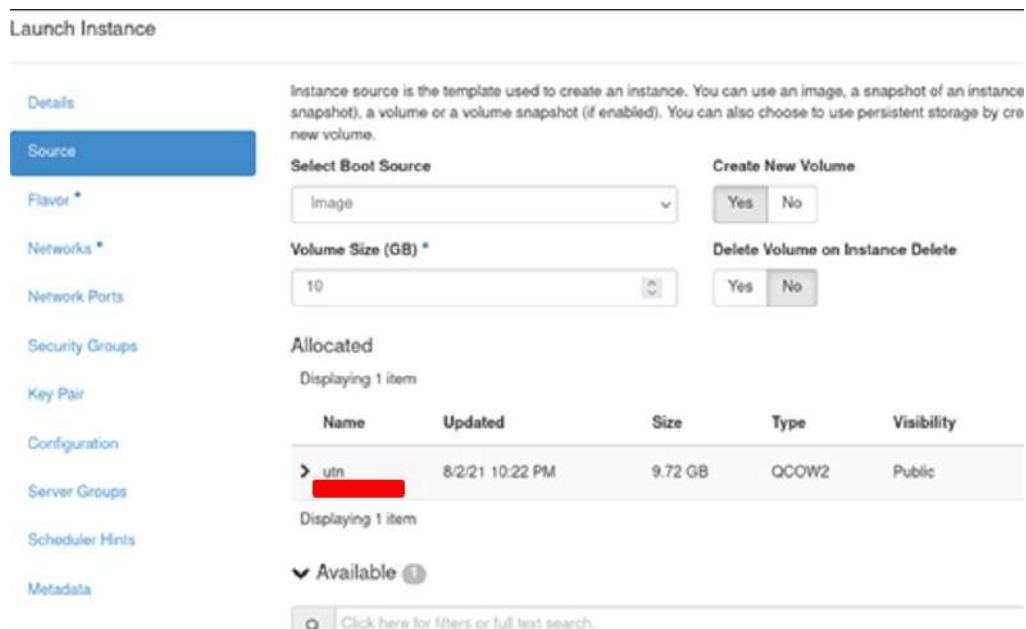
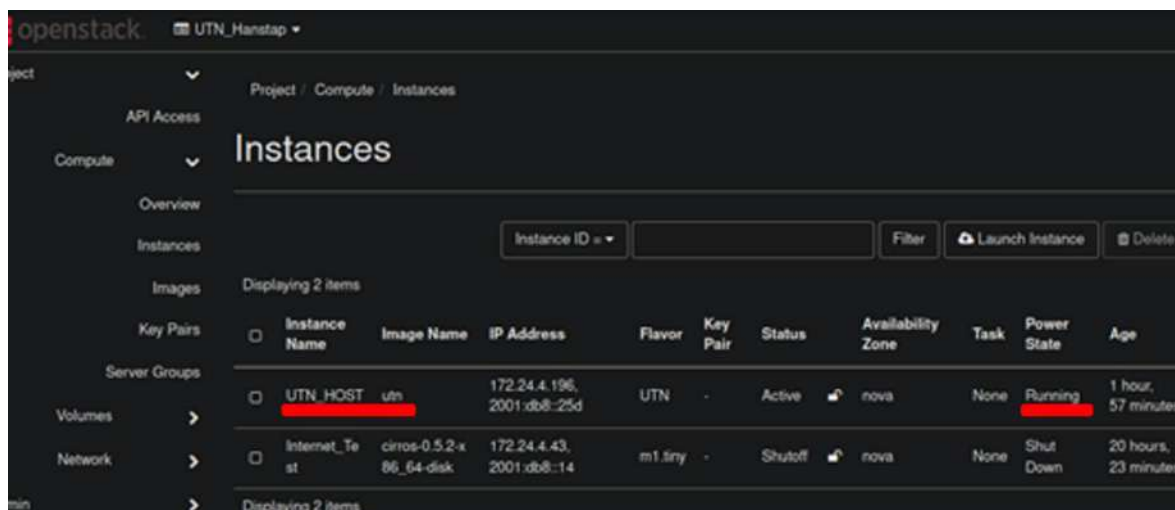
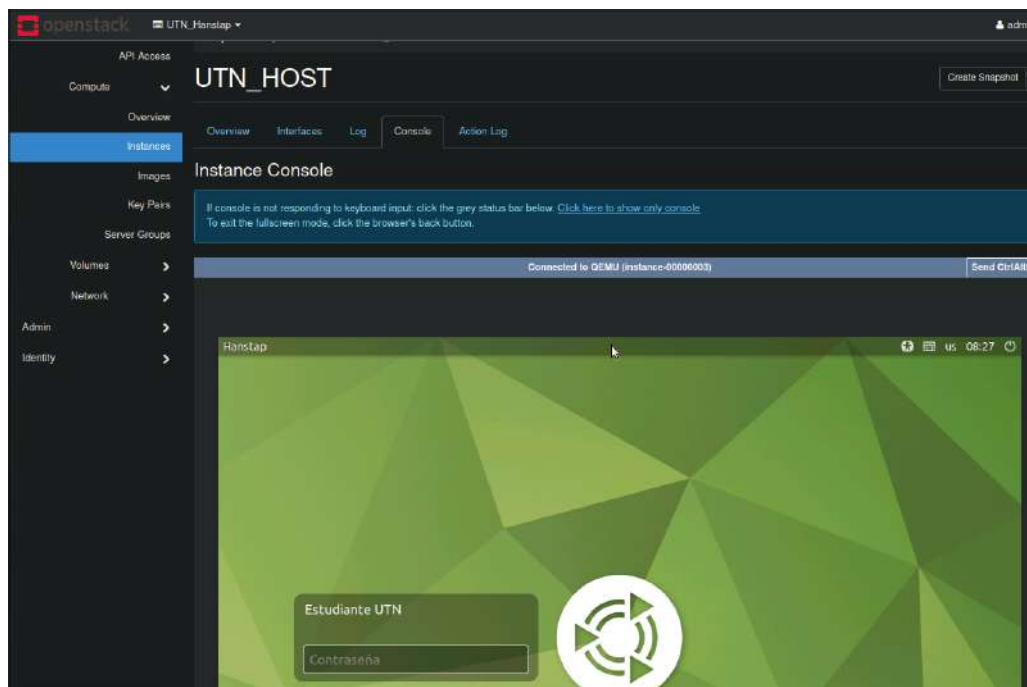


Figura 91 Instancia Funcionando.



De esta manera se observa el funcionamiento de la instancia creada la cual puede asignarse a un estudiante para realizar las prácticas, utilizando el escritorio virtual mostrado en la figura 92, mediante la pestaña “Console”, donde es apreciable la interfaz gráfica del SO.

Figura 92 Escritorio Virtual.



1.25 Pruebas de rendimiento

Esta sección presenta el desarrollo de las pruebas de rendimiento tanto de los recursos físicos como virtuales y de red integrados en la infraestructura, de esta manera, es posible realizar un análisis de rendimiento y consumo de recursos comparado al análisis realizado en el punto 3.1.1, así como los tiempos de respuesta y de arranque de las instancias, utilizando los siguientes parámetros:

- Utilización de recursos físicos # consumo de RAM y CPU de host físico
- Utilización de recursos por instancia # Consumo vRAM y vCPU de instancia virtual
- Velocidad transferencia datos entre instancias # Transmisión de red
- Rendimiento instancias simultáneas # Funcionamiento simultáneo de instancias Virtuales
- Rendimiento nodos mininet # Número de nodos posibles en topologías mininet.

1.25.1 Utilización de Recursos Físicos.

Los análisis de recursos tanto físicos como virtuales se realizaron en los dos escenarios mediante las mismas herramientas, previamente determinadas las cuales son: `htop` y `stress`, entre los recursos medidos están CPU, RAM y procesos activos en el sistema; indicados en la figura 93, datos tomados del host principal y como resultado final se muestra la utilización media² del sistema que es de 1.4% de CPU, además la utilización de memoria RAM es de 8.94GB, datos que se indican la tabla 12.

Figura 93 Recursos HOST Principal.

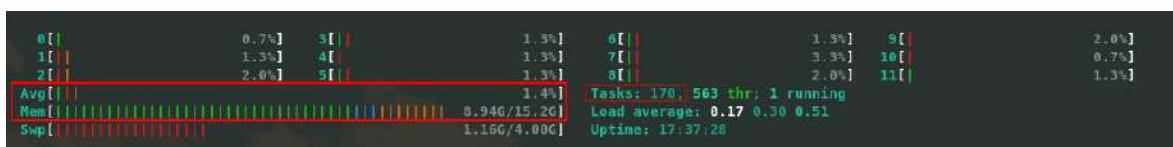


Tabla 12 Utilización Recursos Host Principal.

Carga Media CPU	Utilización RAM	# Procesos
1.4%	8.94GB	170
Carga media dimensionada CPU	RAM dimensionada	
6%	9GB	

Los resultados obtenidos en la tabla 12 son tomados del host físico sobre el cual se tiene una instancia virtual con una asignación de 2 cores de cpu y 2GB de RAM, por lo que se aprecia que la utilización de CPU y RAM es baja y no se acerca a la utilización que se considera en el dimensionamiento en el punto 3.1.1. donde nos indica que la utilización requerida es de 9GB de RAM y una utilización de CPU del 6% por los

² La herramienta `htop` puede no mostrar por defecto la carga media del CPU, para activarla es necesario presionar la tecla `F10`, mostrando un menú del cual se puede seleccionar `cpu average`, y se lo activa presionando la tecla `espacio`.

requerimientos propios del SO base y el sistema de Cloud computing OpenStack, lo que nos indica un mejor rendimiento al esperado.

Pruebas host virtual.

De la misma manera se realizan las pruebas en la instancia virtual, donde se observa una relación entre la utilización de sus recursos con los del host físico, obteniendo que la utilización media es de 0.3% y de memoria RAM es de 582MB, así como 90 procesos activos en el sistema, valores que son tomados de la figura 94 y mostrados en la tabla 13, Como resultado de las pruebas se verifica que los requerimientos de a instancia virtual establecidos en el punto 3.1.1. donde se indicaba que la instancia tendría una utilización en idle del 10%, así como una utilización de 1GB basándose en los requerimientos de sistema mismos que no fueron necesarios, lo que ofrece un mejor rendimiento al esperado

Figura 94 Utilización Recursos Host Virtual.

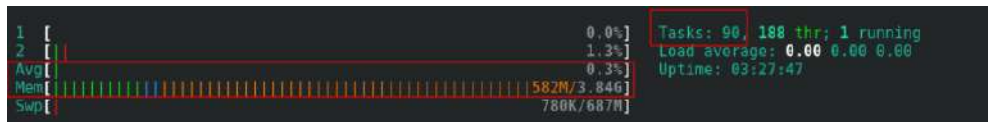


Tabla 13 Utilización Recursos Host Virtual.

Carga Média CPU	Utilización RAM	# Procesos
0.3%	1GB	90
Carga dimensionada CPU	Utilización RAM dimensionada	
10%	2GB	

Prueba en carga.

Las pruebas en carga son realizadas para una instancia virtual, mediante la aplicación de una carga sintética en el CPU, es decir se establece una tarea sin fin a cada uno de los cores del cpu, utilizando el comando “*stress -cpu 2*”, de esta manera se puede simular que un sistema operativo exige el 100% del rendimiento de su hardware. Al ser una instancia virtual, la prueba puede ser observada en el host físico mediante el comando *htop*, obteniendo la utilización de los 12 cores físicos los cuales conforman el CPU físico tal como se muestra en la figura 95

Figura 95 Utilización Host físico en carga.

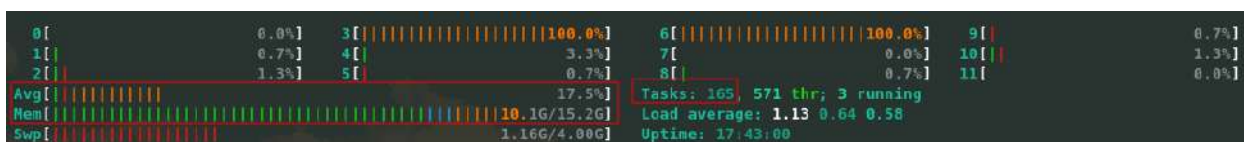
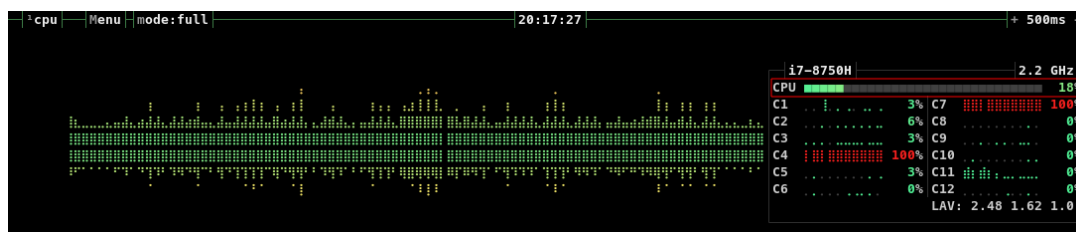


Figura 96 Histograma prueba en carga



Por otra parte, la figura 96 muestra el histograma de utilización de cada uno de los cores durante un lapso de tiempo de 30 segundos desde que la prueba fue ejecutada; en donde se evidencia que no todos están siendo utilizados debido a que el sistema operativo tiene incorporado un esquema de balanceo de carga. Es apreciable entonces que la carga de CPU promedio es del 18% y la utilización de memoria RAM es de 10.16GB, valores presentados en la tabla 14, mismos que permiten demostrar que el dimensionamiento realizado en el punto 3.1.1. fue correcto dividiendo la utilización media del CPU entre la instancia virtual y la instancia virtual, de la misma manera se observa un aumento de la utilización de la RAM debido a la instancia virtual como se aprecia en la tabla 14

Tabla 14 Utilización Host físico en Carga.

Carga Média CPU	Utilización RAM	# Procesos
18%	10.16GB	165
Carca Dimensionada CPU	Utilización Estimada RAM	Utilización RAM sistema
10%	1.22GB	8.94GB

1.25.2 Pruebas de Red

Las pruebas de red se basan en el RFC-2544 (IETF, 2020), el cual determina los siguientes parámetros a medir: latencia, tasa de transferencia, y tramas perdidas; el software a utilizar para obtener estos datos es *iperf3*, mismo que funciona en un esquema cliente-servidor para lo cual se ejecuta comando con el argumento -s para activarlo en modo servidor y con el argumento -c <ip server> para establecer la conexión desde el cliente, al realizar esto la herramienta brinda la información requerida para su posterior análisis

- Latencia

Este valor representa el retardo de tiempo que existe entre el envío de un paquete entre 2 host, los valores referenciales para la prueba se indican en la tabla 15

Tabla 15 Tabla Latencia

Métrica	Latencia (ms)
Estándar/Aceptable	5-40ms
Alta/No Aceptable	150ms

(Rogier, 2016)

- Reintentos (tramas perdidas)

Este valor representa el número de tramas perdidas en una transmisión que deben ser reenviadas desde el transmisor hacia el receptor, los valores de referencia para la prueba son indicados en la tabla 16.

Tabla 16 Tabla Retardo

Perdida de tramas	Valor Estimado.
Normal/Aceptable	<2%
Alta/No Aceptable	>5%

Fuente: (Schoenfelder, 2021)

- Tasa de transferencia efectiva (Throughput)

Este valor representa la velocidad a la cual pueden ser enviados los paquetes entre los 2 host, para esto, se requiere establecer un hosts como servidor y otro como cliente el cual establece la conexión, la prueba será generada mediante el comando *iperf3*, su resultado será verificado mediante la fórmula de *throughput* mostrada en la Ec. 8, la que considera el tamaño máximo del segmento *MSS(Maximun Segment Size)*, la latencia o *RTT(Round Trip Time)* y el porcentaje de paquetes perdidos *p*.

$$Throughput = \left(\frac{\frac{MSS*8}{RTT} * \frac{1}{\sqrt{p*10^{-6}}}}{2} \right) \quad \text{Ec. 8}$$

Mediante a la Ec.8 y considerando los valores indicados de latencia (*5ms*) en la tabla 15, el porcentaje de paquetes errados equivalente al (2%) tomados de la tabla 16 y al conocer que el tamaño de trama utilizado por *iperf3* es de 1500 bytes, se realiza el remplazo de valores respectivos, se obtiene su resultado en Mbps

$$Throughput = \left(\frac{1500(bytes)*8}{5ms} * \frac{1}{\sqrt{2\%*10^{-6}}} \right) \quad \text{Ec. 9}$$

$$Throughput = (809219.49Mbps) \quad \text{Ec. 10}$$

Este valor expresado en Gbps da como resultado un enlace de 0.8 Gbps, mismo está dentro del rango de trabajo del estándar de la IEEE 802.3z, por lo que se tomara

como referencia para las pruebas siguientes, estos valores son indicados en la tabla 17 para su posterior utilización,

Tabla 17 Valores referenciales para pruebas de red

Valores referenciales		
Latencia (RTT)	% Tramas Perdidas (p)	Throughput
5 ms	<2%	0.8Gbps

Throughput entre Instancias Virtuales.

La prueba de throughput se realiza entre las instancias generadas sobre la infraestructura desplegada en la presente tesis, con el fin de identificar su rendimiento y tasa de transición máxima alcanzable entre ellas, se establece a la máquina con Dirección IP 172.24.4.116 como servidor tal como se muestra en la figura 97 y la estación cliente mostrada en la figura 98.

Figura 97 Instancia definida como Servidor.

```
estudiante@Hanstap:~$ iperf3 -s
-----
Server listening on 5201
-----
```

Figura 98 Instancia definida como Cliente.

```
estudiante@Hanstap:~$ iperf3 -c 172.24.4.116
Connecting to host 172.24.4.116, port 5201
[ 5] local 172.24.4.140 port 34552 connected to 172.24.4.116 port 5201
[ ID] Interval          Transfer      Bitrate      Retr  Cwnd
[ 5]  0.00-1.00    sec   367 MBytes  3.07 Gbits/sec  719  1.29 MBytes
```

Una vez realizada la prueba se obtienen los resultados como se muestra en la figura 99 y tabla 18, donde es posible verificar las métricas requeridas para su análisis como son throughput, reintentos y transferencia total, su latencia puede ser observada en la figura 100 mediante un ping. Los resultados indican que la latencia es de 1.242ms, valor que está dentro de los parámetros mínimos aceptados para transmisión de datos, la prueba también indica que existieron errores en la transmisión donde fue requerido el reenvío de 125 paquetes, indicando un comportamiento normal de una red, la conectividad entre interfaces es de 1Gbps; al analizar su throughput mediante la ecuación 8, se obtiene un valor teórico de 3.25Gbps, y al compararlos con los valores de la tabla 17 se concluye que su latencia y pérdida de paquetes se encuentra en el rango aceptado mientras que el rendimiento es menor al esperado pero está dentro de los valores referenciales del estándar IEE-802.3z de 1Gbps.

Figura 99 Prueba Rendimiento Red entre Instancias Virtuales.

```

estudiante@Hanstap:~$ iperf3 -c 172.24.4.35
Connecting to host 172.24.4.35, port 5201
[ 5] local 172.24.4.114 port 47590 connected to 172.24.4.35 port 5201
[ ID] Interval      Transfer    Bitrate    Retr    Cwnd
[ 5]  0.00-1.00  sec    140 MBytes  1.17 Gbits/sec  68    1.41 MBytes
[ 5]  1.00-2.00  sec    149 MBytes  1.25 Gbits/sec   0    1.51 MBytes
[ 5]  2.00-3.00  sec    142 MBytes  1.19 Gbits/sec   0    1.65 MBytes
[ 5]  3.00-4.00  sec    124 MBytes  1.04 Gbits/sec   0    1.73 MBytes
[ 5]  4.00-5.00  sec    138 MBytes  1.15 Gbits/sec  57    1.28 MBytes
[ 5]  5.00-6.00  sec    139 MBytes  1.10 Gbits/sec   0    1.30 MBytes
[ 5]  6.00-7.00  sec    138 MBytes  1.15 Gbits/sec   0    1.41 MBytes
[ 5]  7.00-8.00  sec    140 MBytes  1.17 Gbits/sec   0    1.44 MBytes
[ 5]  8.00-9.00  sec    142 MBytes  1.20 Gbits/sec   0    1.48 MBytes
[ 5]  9.00-10.00 sec    139 MBytes  1.16 Gbits/sec   0    1.49 MBytes
[ ID] Interval      Transfer    Bitrate    Retr
[ 5]  0.00-10.00  sec    1.36 GBytes  1.17 Gbits/sec  125
[ 5]  0.00-10.01  sec    1.35 GBytes  1.16 Gbits/sec
iperf Done.
  
```

Figura 100 Prueba Latencia

```

estudiante@Hanstap:~$ ping 172.24.4.35
PING 172.24.4.35 (172.24.4.35) 56(84) bytes of data:
64 bytes from 172.24.4.35: icmp_seq=1 ttl=64 time=2.02 ms
64 bytes from 172.24.4.35: icmp_seq=2 ttl=64 time=1.27 ms
64 bytes from 172.24.4.35: icmp_seq=3 ttl=64 time=1.13 ms
64 bytes from 172.24.4.35: icmp_seq=4 ttl=64 time=1.07 ms
64 bytes from 172.24.4.35: icmp_seq=5 ttl=64 time=0.947 ms
64 bytes from 172.24.4.35: icmp_seq=6 ttl=64 time=1.08 ms
64 bytes from 172.24.4.35: icmp_seq=7 ttl=64 time=1.19 ms
^C
--- 172.24.4.35 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6006ms
rtt min/avg/max/mdev = 0.947/1.242/2.020/0.330 ms
estudiante@Hanstap:~$
  
```

Rendimiento entre Instancia Virtual y el Host físico

En esta prueba se desarrolló el mismo procedimiento realizado en el punto anterior, con la diferencia que el enlace se encuentra entre la instancia y el host físico, obteniendo los resultados mostrados en la figura 101 sobre la cantidad de datos transmitidos, la tasa de transmisión y reintentos ocurridos, mismos que son presentados en la tabla 18, mientras que la latencia se muestra en la figura 102 mediante un ping.

Figura 101 Rendimiento Virtual-Físico Red.

```

estudiante@Hanstap:~$ lperf3 -c 172.24.4.1
Connecting to host 172.24.4.1, port 5201
[ 5] local 172.24.4.114 port 52440 connected to 172.24.4.1 port 5201
[ ID] Interval      Transfer    Bitrate    Retr    Cwnd
[ 5] 0.00-1.00    sec 174 MBytes 1.46 Gbits/sec 0 1.51 MBytes
[ 5] 1.00-2.00    sec 190 MBytes 1.59 Gbits/sec 0 2.01 MBytes
[ 5] 2.00-3.00    sec 191 MBytes 1.00 Gbits/sec 0 2.33 MBytes
[ 5] 3.00-4.00    sec 194 MBytes 1.02 Gbits/sec 0 2.71 MBytes
[ 5] 4.00-5.00    sec 171 MBytes 1.44 Gbits/sec 0 2.87 MBytes
[ 5] 5.00-6.00    sec 204 MBytes 1.71 Gbits/sec 0 2.87 MBytes
[ 5] 6.00-7.00    sec 158 MBytes 1.32 Gbits/sec 0 2.87 MBytes
[ 5] 7.00-8.00    sec 185 MBytes 1.55 Gbits/sec 0 3.02 MBytes
[ 5] 8.00-9.00    sec 178 MBytes 1.49 Gbits/sec 0 3.02 MBytes
[ 5] 9.00-10.00   sec 201 MBytes 1.09 Gbits/sec 0 3.02 MBytes
[ ID] Interval      Transfer    Bitrate    Retr
[ 5] 0.00-10.00   sec 1.80 GBytes 1.55 Gbits/sec 0
[ 5] 0.00-10.01   sec 1.80 GBytes 1.55 Gbits/sec 0
iperf Done.
  
```

```

estudiante@Hanstap:~$ ping 172.24.4.1
PING 172.24.4.1 (172.24.4.1) 56(84) bytes of data:
64 bytes from 172.24.4.1: icmp_seq=1 ttl=64 time=1.65 ms
64 bytes from 172.24.4.1: icmp_seq=2 ttl=64 time=0.737 ms
64 bytes from 172.24.4.1: icmp_seq=3 ttl=64 time=0.516 ms
64 bytes from 172.24.4.1: icmp_seq=4 ttl=64 time=0.706 ms
64 bytes from 172.24.4.1: icmp_seq=5 ttl=64 time=0.693 ms
64 bytes from 172.24.4.1: icmp_seq=6 ttl=64 time=0.522 ms
64 bytes from 172.24.4.1: icmp_seq=7 ttl=64 time=0.634 ms
64 bytes from 172.24.4.1: icmp_seq=8 ttl=64 time=0.566 ms
^C
--- 172.24.4.1 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7150ms
rtt min/avg/max/mdev = 0.516/0.752/1.648/0.347 ms
estudiante@Hanstap:~$
  
```

Figura 102 Latencia Prueba

El análisis de los resultados presentes en la tabla 18, indican un enlace de 1.55 Gbps, Un retardo muy bajo siendo de 0.752ms, de la misma manera en la transmisión existieron cero reintentos perteneciendo a <2% de paquetes perdidos, valores comparados con la tabla 17, de la misma manera al aplicar estos datos en la ecuación 8 se obtiene una transmisión teórica de 5.38Gbps dando como resultado que la velocidad es mucho menor a la esperada, pero aún supera la velocidad del standard IEEE-802.3z.

Tabla 18 Pruebas de Red

Prueba	Throughput Calculado	Throughput Medido	Retardo	Reintentos
Virtual-Virtual	3.25Gbps	1.17Gbps	1.242ms	125
Físico – Virtual	5.38Gbps	1.55Gbps	0.752ms	0

1.26 Rendimiento Instancias Simultáneas.

En esta etapa se verifica el rendimiento del sistema al momento de generar instancias simultáneamente, la capacidad máxima utilizable del sistema fue establecida en el punto 3.1.1, la cual especifica que se tiene un rendimiento máximo al llegar a 10, para ello, se realizarán 2 pruebas con un número incremental de 5 y 10 instancias, todas utilizarán el Flavor UTN de manera que sus especificaciones sean las mismas, para poder obtener el tiempo que al sistema le toma iniciar, se utiliza el comando *systemd-analyze*. El procedimiento de generación será mediante la interfaz web, como se muestran en la figura 103, esta medición será tomada habiendo esperado un lapso de 5 minutos en los cuales el sistema habrá iniciado correctamente. Los resultados de las pruebas de 5 y 10 instancias son mostrados en las tablas 19 y 20 respectivamente.

Figura 103 Instancias Simultaneas

Launch Instance

Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Project Name
UTN_Handlap

Instance Name *
UTN_TEST

Description

Availability Zone
nova

Count *
5

Total Instances (10 Max)
50%

0 Current Usage
5 Added
5 Remaining

Cancel Back Next Launch Instance

Tabla 19 Rendimiento 5 Instancias.

Instancia	Tiempo (s)
1	34.332s
2	48.042s
3	39.896s
4	36.474s
5	44.411s
Promedio	40.631s

Tabla 20 10 Instancias Simultaneas

Instancia	Tiempo (s)
1	48.112s
2	39.866s
3	36.471s

4	44.371s
5	40.711s
6	31.013
7	40.242
8	22.141
9	40.769
10	46.336
Promedio	42.03

Al observar los tiempos de ejecución de las instancias se aprecia un leve incremento en el tiempo requerido para ejecutar un número mayor, pasando de 40 a 42 segundos en promedio, de esta manera, es posible concluir el sistema permite un máximo de 10 instancias simultáneas, sin experimentar reducción perceptible en su rendimiento confirmando el dimensionamiento realizado en el punto 3.1.1.

1.27 Rendimiento Nodos Mininet.

En esta sección se realiza un análisis de las capacidades de simulación de las instancias, al momento de generar hosts virtuales pertenecientes a la red SDN, mediante la aplicación de mininet. Se utiliza su interfaz CLI por terminal, la cual permite generar topologías con un número determinado de host de manera rápida, con el objetivo de mantener los resultados constantes y únicamente verificar el número de host, se utilizará la misma topología “*Tree*”(árbol) en la cual se cambiará el número de puntos intermedios en la red *profundidad* y *Abanico* el cual indica el número de ramificaciones presentes en cada punto remoto de la red, estos serán indicados en cada prueba para posterior recreación. Una vez ingresados los valores de *Topologia=Tree*, *Profundidad=2* *Abanico=4* en el comando, se obtiene una visualización en la controladora ONOS como se muestra en la figura 104, el reporte sobre la topología está indicada figura 105 donde

indica el número de host creados, de la misma manera la utilización de recursos es apreciable en la figura 106

Figura 104 Prueba I

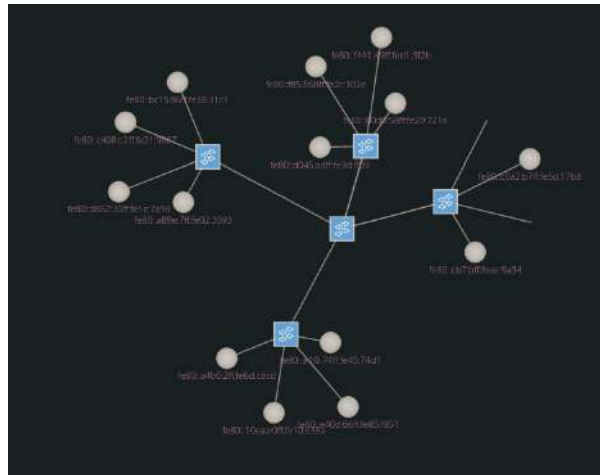
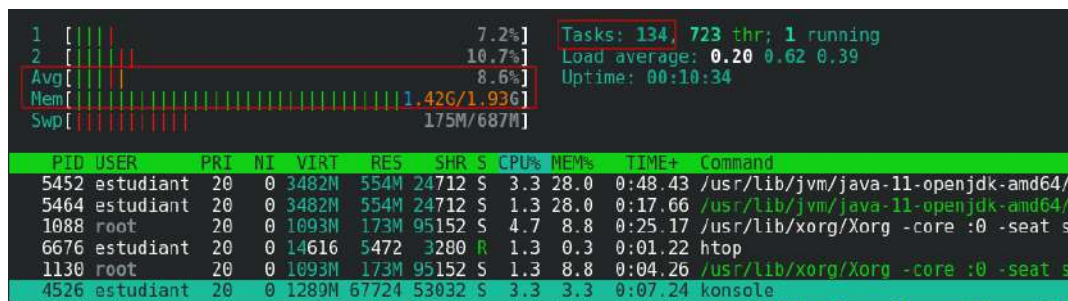


Figura 105 Reporte Prueba 1 ONOS

Sumario de ONOS	
Versión	2.5.1
Dispositivos	5
Enlaces	8
Hosts	16
SCCs de la topología	1
Intents	0
Flujos	25

Figura 106 Utilización prueba 1



Como resultado, la instancia no presentó ningún inconveniente al poder ejecutar un número de host, enlaces y flujos al simular esta topología, en la cual se pudo comprobar la convergencia de esta topología mediante un ping entre todos sus hosts, como se muestra en la figura 106, presentando que los 240 paquetes fueron entregados correctamente, de la misma manera la utilización de CPU fue de 8.6% mientras que la de memoria RAM fue de 1.42GB, manteniendo 134 procesos activos.

Figura 107 Prueba I Conectividad.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12 h13 h14 h15 h16
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12 h13 h14 h15 h16
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12 h13 h14 h15 h16
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12 h13 h14 h15 h16
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12 h13 h14 h15 h16
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h13 h14 h15 h16
h13 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h14 h15 h16
h14 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h15 h16
h15 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h16
h16 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
*** Results: 0% dropped (240/240 received)
```

La segunda prueba se desarrolla con los valores de *Topologia=Tree*, *Profundidad=3* *Abanico=4* obteniendo una visualización en la controladora ONOS como se muestra en la figura 108 el reporte sobre la topología está indicada figura 109 donde indica el número de host creados, mientras que en la figura 110 se observa la utilización de recursos del sistema durante la prueba.

Figura 108 Prueba 1

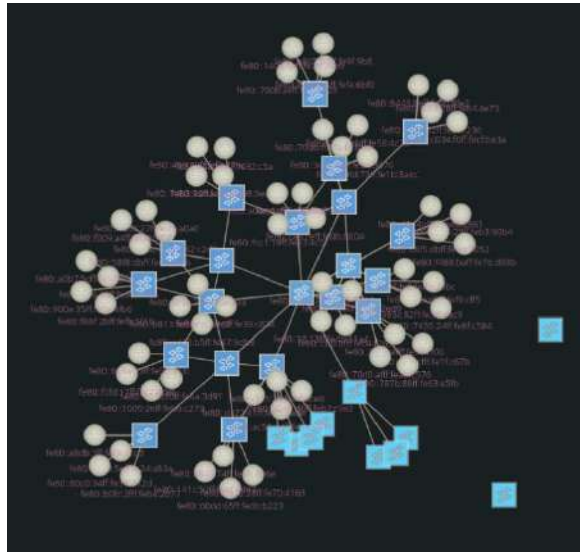
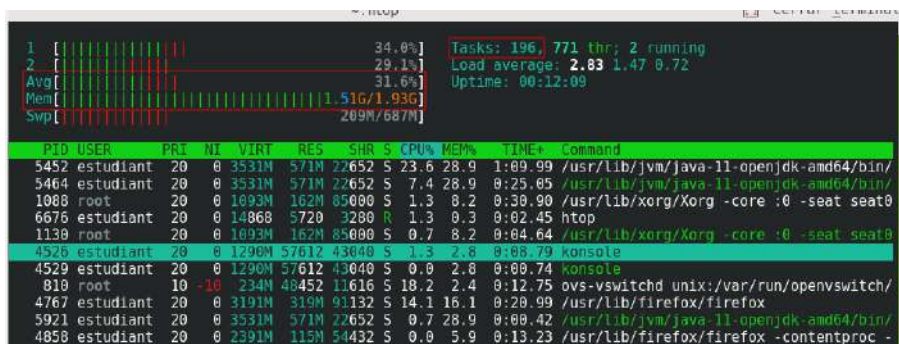


Figura 109 Reporte Prueba 2 ONOS

Sumario de ONOS	
Versión	2.5.1
Dispositivos	31
Enlaces	40
Hosts	64
SCCs de la topología	1
Intents	0
Flujos	155

Figura 110 Rendimiento prueba 2



Como resultado, la instancia no presento ningún inconveniente al poder ejecutar 64 host, 40 enlaces y 155 flujos al simular la topología indicada, en la cual es posible comprobar la convergencia mediante un ping entre todos sus hosts, como se muestra en la figura 111, donde se indica que 4027/4032 paquetes fueron entregados correctamente, de la misma manera la utilización de CPU fue de 31.6% mientras que la de memoria RAM fue de 1.51GB, manteniendo 196 procesos activos.

Figura 111 Prueba I Conectividad.

```

h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
h55 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
h56 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
h57 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
h58 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
h59 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
h60 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
h61 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
h62 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
h63 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
h64 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13
h54 h55 h56 h57 h58 h59 h60 h61 h62 h63
*** Results: 0% dropped (4027/4032 received)
wtnlnt>

```

La tercera prueba se desarrolla con los valores de *Topologia=Tree*, *Profundidad=5* *Abanico=4* obteniendo una visualización en la controladora ONOS como se muestra en la figura 112 el reporte sobre la topología está indicada figura 113 donde indica el número de host creados, mientras que en la figura 114 se observa la utilización de recursos del sistema durante la prueba.

Figura 112 Prueba I

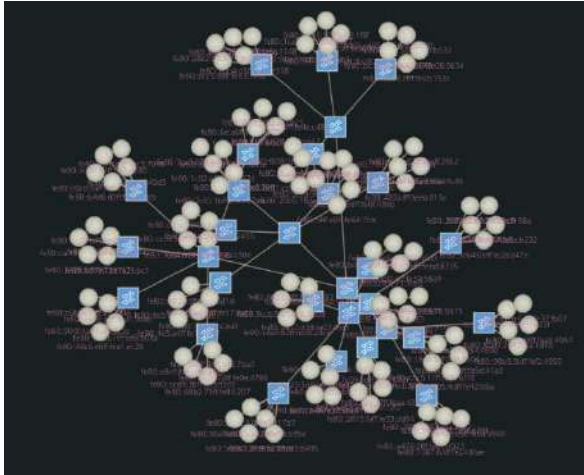
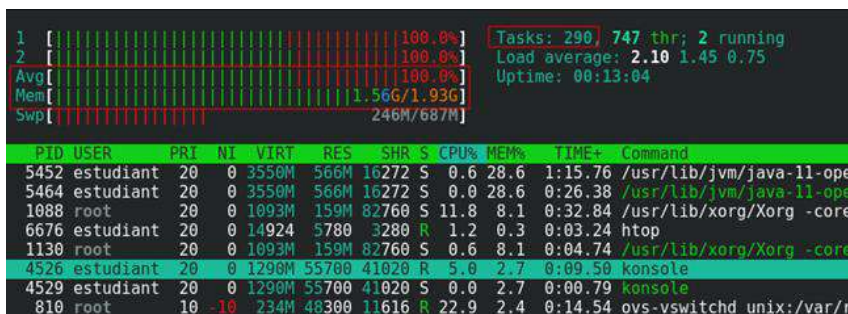


Figura 113 Reporte Prueba 1 ONOS

Sumario de ONOS	
Versión	2.5.1
Dispositivos	31
Enlaces	47
Hosts	125
SCCs de la topología	13
Intents	0
Flujos	155

Figura 114 Rendimiento prueba 3



La instancia presentó sobrecarga con el número de host que tenía a su disposición, por lo que se establece que 125 host sería un número de host no aplicable para las simulaciones, debido a la gran utilización del CPU llegando a un 100% mientras que la memoria RAM presenta 1.56GB y 290 procesos activos. La tabla 22 presenta una recopilación de los datos obtenidos en las pruebas realizadas.

Tabla 21 Utilización recursos host SDN

Topología	Profundidad	Abanico	Host	CPU	RAM	procesos
Tree	2	4	16	8.6%	1.42GB	134
Tree	3	4	64	31.6	1.51GB	196
Tree	5	4	125	100%	1.56	290

1.28 Funciones SASS

Este punto muestra las funcionalidades, servicios y capacidades de la imagen de host que corre sobre OpenStack, esta permite la creación de topologías de red experimentales donde será posible la utilización de servicios como OVS para gestión SDN y de implementaciones NFV, permitiendo un control sobre cada uno de los componentes de la topología mediante interfaz gráfica.

1.28.1 Ejecución Mininet.

Al ser un comando global, *mn* (mininet) este puede ser ejecutado directamente y generar cualquier topología mediante su sintaxis o implementación con un script. Para la demostración de funcionamiento, se utilizará un script con sintaxis de mininet desarrollando una topología en árbol con 2 puntos fuente y 2 ramificaciones mediante el argumento `--topo tree,2,2`, y utilizando `sudo mn --switch=default,protocols=OpenFlow10` para definir el protocolo Openflow para los switch de la topología

```
sudo mn --switch=default,protocols=OpenFlow10 --topo tree,2,2
```

Como resultado se obtiene la creación de la topología y se accede a la terminal de mininet donde será posible la configuración y verificación del funcionamiento de los dispositivos, el resultado se muestra en la figura 114, donde es apreciable los 4 host y los 3 switches establecidos para la topología.

Figura 115 Funcionamiento mininet cli

```
estudiante@Hanstap:~$ sudo mn --switch=default,protocols=openFlow10 --topo tree,2,2
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, s2) (s1, s3) (s2, h1) (s2, h2) (s3, h3) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```

1.28.2 Funcionamiento OVS

Como parte de su funcionamiento mininet genera y establece instancias virtuales de OVS generando interfaces, de manera que se puede verificar las conexiones utilizando el comando **ip link show** para listar las interfaces de su respectivo switch, identificado por las iniciales *s1*, *s2* y *s3* al inicio de su nombre de interfaz tal y como se aprecia en la figura 115

Figura 116 Interfaces Virtuales mininet-ovs

```

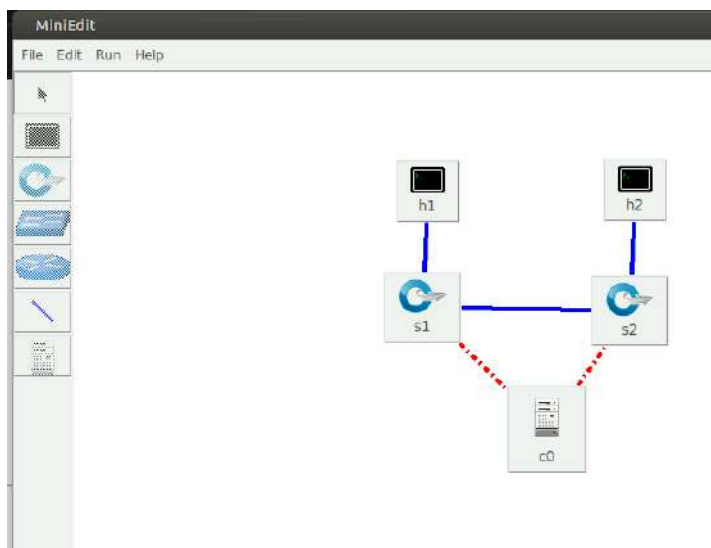
estudiante@hanslab:~$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether fa:16:3e:bf:a8:a4 brd ff:ff:ff:ff:ff:ff
    altname enp8s3
3: s2-eth1g1-eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether 26:4b:56:74:58:ba brd ff:ff:ff:ff:ff:ff
4: s1-eth1g2-eth: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether 4a:51:9e:e6:47:6f brd ff:ff:ff:ff:ff:ff
5: s3-eth3g1-eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether ba:3e:fb:15:d4:71 brd ff:ff:ff:ff:ff:ff
6: s1-eth2g3-eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether ae:b7:a4:24:e1:a8 brd ff:ff:ff:ff:ff:ff
7: s2-eth1g1f2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether 2a:ef:a9:ac:e9:b1 brd ff:ff:ff:ff:ff:ff link-netnsid 0
8: s2-eth2g1f2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether 62:d9:d5:6a:09:c3 brd ff:ff:ff:ff:ff:ff link-netnsid 1
9: s3-eth1g1f2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether 9a:b1:08:c3:d7:ac brd ff:ff:ff:ff:ff:ff link-netnsid 2
10: s3-eth2g1f2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether 0a:79:b1:7c:85:7e brd ff:ff:ff:ff:ff:ff link-netnsid 3
11: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether f6:dd:2f:33:20:d3 brd ff:ff:ff:ff:ff:ff
12: s1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether b2:c0:3b:4c:aa:43 brd ff:ff:ff:ff:ff:ff
13: s3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether a6:f4:4c:e0:9e:4c brd ff:ff:ff:ff:ff:ff
14: s2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 32:33:e7:cd:14:40 brd ff:ff:ff:ff:ff:ff
estudiante@hanslab:~$

```

Generación topologías mediante Mininet

Mininet contiene además una interfaz gráfica, su presentación es más amigable con el usuario, permitiendo una comprensión de su funcionamiento, como ejemplo se genera una topología simple en su ambiente gráfico, mismo que puede ser observado mediante la controladora ONOS instalada en el sistema, la topología consta de una controladora, 2 switch-ovs y 2 host virtuales como se muestra en la figura 116.

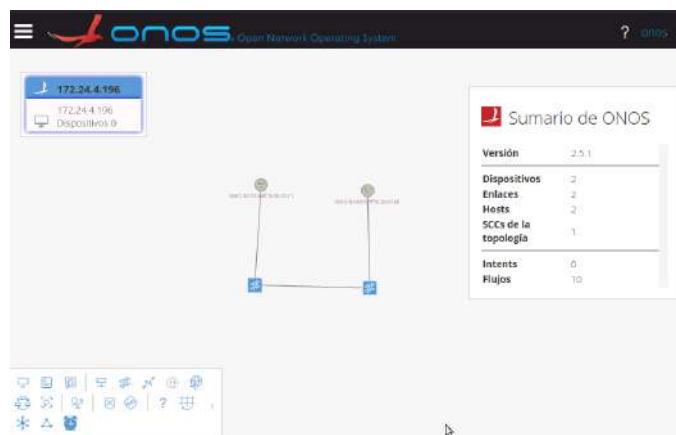
Figura 117 Mininet GUI.



1.28.3 Funcionamiento ONOS

La topología simple generada es mostrada en la controladora, donde se aprecia las conexiones existentes entre cada switch y host de la topología, comprobando su funcionamiento en la interfaz gráfica mostrada en la figura 117.

Figura 118 Controladora ONOS.



Verificación Conectividad

Como parte de la demostración de funcionamiento se realiza un ping entre las 2 estaciones, el cual es capturado mediante el software wireshark, apreciable en la figura

119, confirmando el funcionamiento de todos los servicios que permiten la generación de topologías SDN.

Figura 119 Transferencia Paquetes.

Time	Source	Destination	Protocol	Length	Info
844	10.0.0.2	10.0.0.1	ICMP	802	Echo (ping) reply id=0x1188, seq=1/0/45880, ttl=64 (request in 830)
845	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x118b, seq=1/256, ttl=64 (reply in 846)
846	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x118b, seq=1/256, ttl=64 (request in 845)
847	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x118b, seq=2/512, ttl=64 (reply in 848)
848	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x118b, seq=2/512, ttl=64 (request in 847)


```

Frame 844: 802 bytes on wire (6416 bits), 802 bytes captured (6416 bits) on interface si-eth1, id 0
Ethernet II, Src: 48:7d:a8:36:dc:21 (48:7d:a8:36:dc:21), Dst: 86:46:b7:2e:d1:a8 (86:46:b7:2e:d1:a8)
  Destination: 86:46:b7:2e:d1:a8 (86:46:b7:2e:d1:a8)
  Source: 48:7d:a8:36:dc:21 (48:7d:a8:36:dc:21)
  Type: IPv4 (0x0800)
  Internet Protocol Version 4, Src: 10.0.0.2, Dst: 10.0.0.1
    0100 ... = Version: 4
    ... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 788
    Identification: 0x16f5 (5877)
    Flags: 0x8965
    Fragment offset: 19248
    Time to live: 64
    Protocol: ICMP [1]
    Header checksum: 0x438d [validation disabled]
    [Header checksum status: Unverified]
  
```

CAPITULO V Guías.

Este capítulo muestra el desarrollo de las guías de prácticas indicadas como parte de la metodología cumpliendo con la parte **Operativa** y de **Optimización** del trabajo de grado. Las guías están desarrolladas en un orden progresivo de aprendizaje, permitiendo al usuario poder familiarizarte con el sistema, permitiéndole desarrollar tareas y administración básica de la infraestructura, ofreciendo una mejor comprensión de la misma.

1.29 Guía I – Acceso y Reconocimiento del Sistema.

Esta guía presentará los métodos de ingreso al sistema y el reconocimiento de los campos de administración dentro de la infraestructura.

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN ELECTRÓNICA Y REDES
DE COMUNICACIÓN



TEMA: Guía Acceso y Reconocimiento IAAS

AUTOR: SANTIAGO ESTÉVEZ.

Esta guía esta enfocada en mostrar las formas de acceso hacia el sistema IAAS por sus diferentes métodos de autenticación, así como navegación hacia cada una de sus principales aspectos y características para una mejor administración.

Objetivos

- Reconocer métodos de conectividad hacia la plataforma OpenStack mediante sus interfaces Web y CLI.
- Desarrollar el proceso de autenticación requerido para realizar modificaciones en la plataforma.
- Realizar un reconocimiento de las características y recursos que son accesibles en OpenStack mediante sus diferentes interfaces.

Interfaz Web.

Uno de los métodos de autenticación con el cual es posible acceder al sistema es mediante su interfaz web, esta interfaz permite ingreso tanto de administrador como de operador al sistema, permitiendo configurar privilegios u permisos de acuerdo con el tipo de usuario, para su ingreso es necesario la utilización de un navegador web. Una vez abierto en navegador ingresamos la ip en la cual la infraestructura se encuentra, la cual para este caso es: **192.168.1.40**. Tal y como se aprecia en la figura 120, al acceder a la interfaz web permitiendo un *Login* donde deben colocarse las credenciales indicadas por el administrador del sistema, como se indica en la figura 121.

Figura 120 Ingreso Gui Web

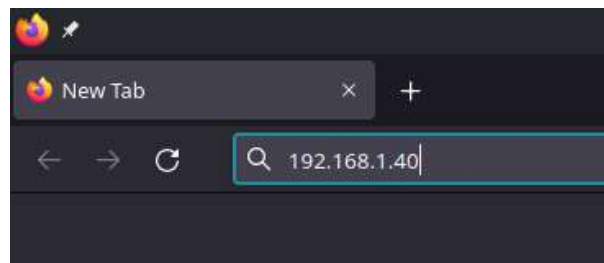
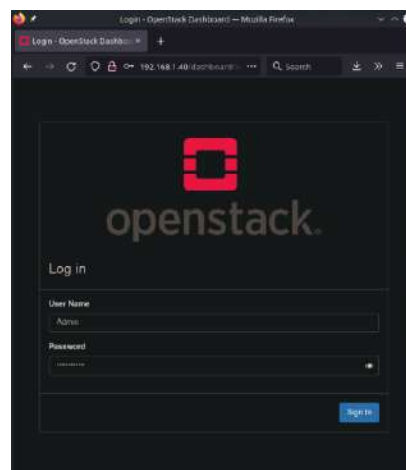
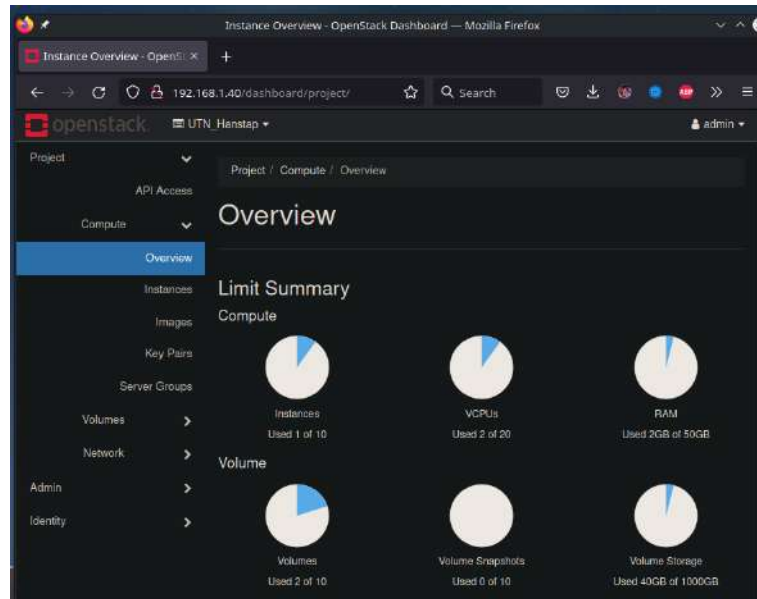


Figura 121 Login Web



Una vez Ingresadas las credenciales proporcionadas por el administrador se tiene acceso al *core* de configuraciones de la infraestructura, como se aprecia en la figura 122

Figura 122 Acceso Administración



De esta manera es posible ingresar a la interfaz web de la aplicación y poder desarrollar una exploración de las características y funcionalidades del sistema.

1.29.1 Reconocimiento Interfaz Web

Una vez que ha sido posible ingresar a la interfaz web es necesario conocer como acceder a puntos importantes sobre la infraestructura como son:

- *Images* # Imágenes de sistemas pre configurados para Instancias.
- *Instancias* # Instancias virtuales creadas.
- *Redes* # *Redes virtuales internas de La IASS.*
- *Proyectos* # Proyectos generados dentro de la IASS.
- *Flavor* # Características virtuales pre-configuradas.

Puntos que serán indicados a continuación.

Acceso Imágenes.

Para el Acceso hacia las Imágenes de sistema que se utilizan para las instancias, además del panel de funciones se accede de la siguiente forma. Se sigue la siguiente secuencia para poder ingresar a las imágenes *Compute* → *Images*, seleccionando este último para desplegar el menú completo de opciones. Mostrado en la figura 123, como se muestra en la figura 124 se ha ingresado hacia el menú de Imágenes donde se realiza la administración de las imágenes de sistema que utilizará la infraestructura para generar las instancias requeridas.

Figura 123 Selección Menú Imágenes.

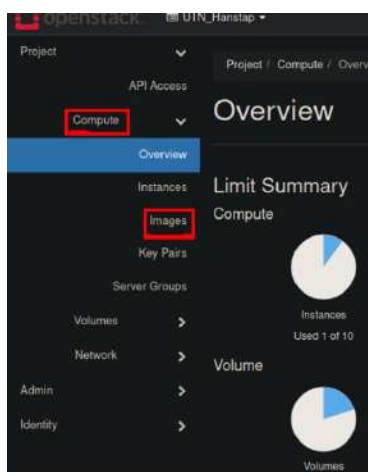
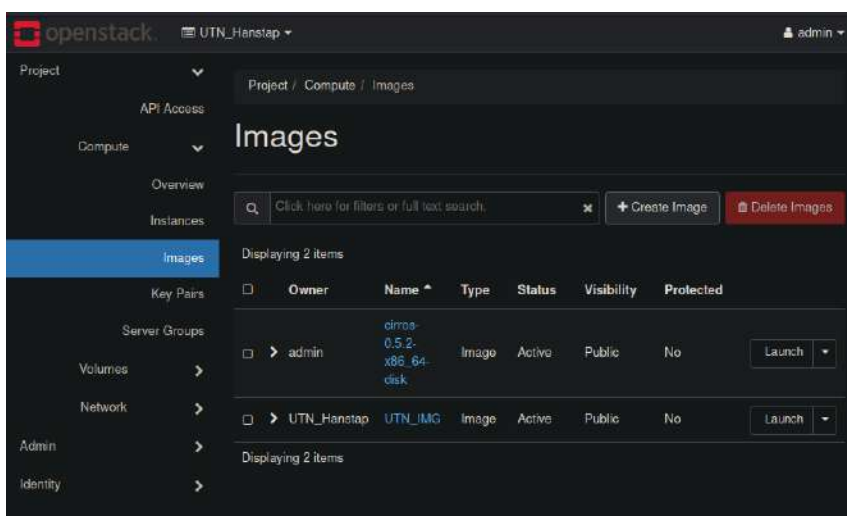


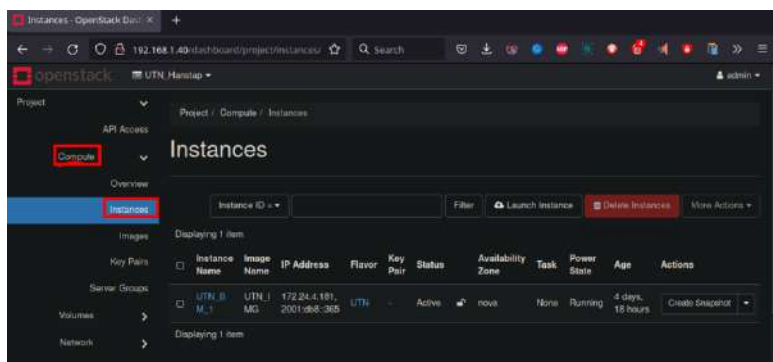
Figura 124 Menú Imágenes.



Acceso a Instancias.

Para ingresar al menú de Instancias se puede ingresar mediante, el siguiente orden *Compute* → *Instances*. Desplegando de esta manera el menú de instancias permitiendo la gestión de estas, como se indica en la figura 125.

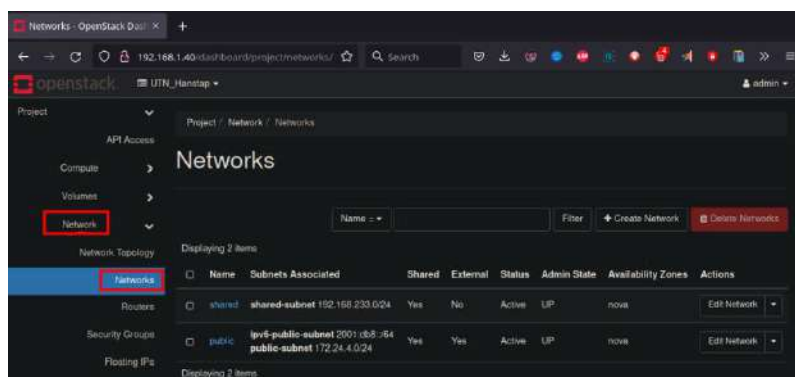
Figura 125 Acceso Menú Instancias.



Acceso a Redes.

Para el ingreso hacia el menú de configuración de redes se accede de la siguiente manera: *Network* → *Networks*. De esta manera se puede acceder a al panel y realizar la gestión de las redes. Como se muestra en la figura 126, se observa en el menú de *Network* existe múltiples ítems que permiten realizar la administración de múltiples aspectos de la red.

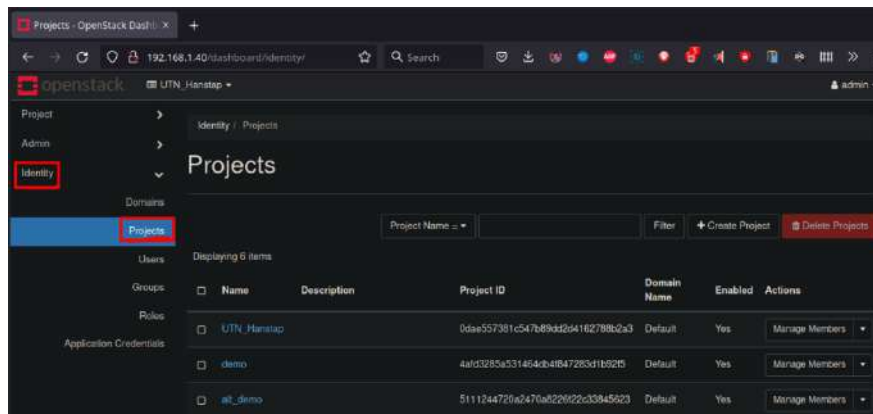
Figura 126 Redes IAAS.



Acceso a Proyectos.

Para el acceso a proyectos donde se puede realizar la administración de múltiples entornos de la IAAS permitiendo organizar sus funcionalidades y servicios. Su acceso se realiza en *Identity* → *Projects*, como se muestra en la figura 127 donde se aprecia los proyectos existentes en la infraestructura, como ejemplo el proyecto UTN_Hanstap.

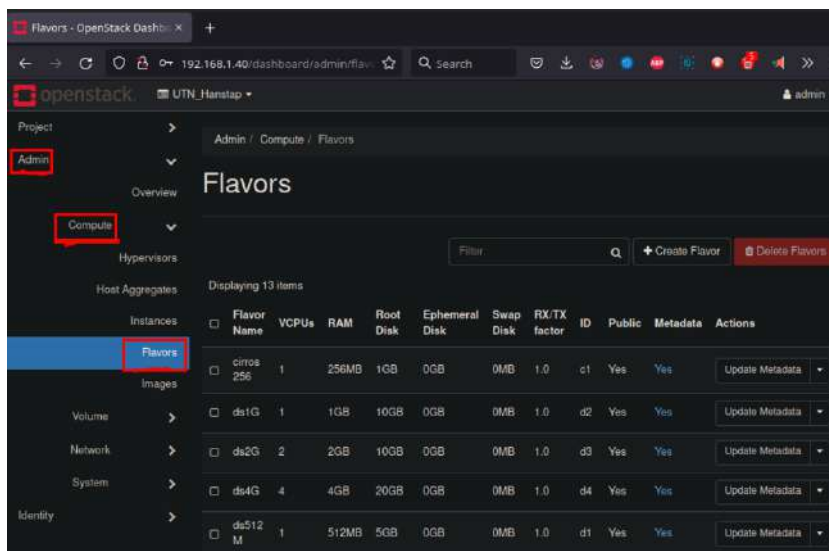
Figura 127 Acceso Proyectos.



Acceso Flavors

Para el ingreso al menú de *Flavors* se ingresa en el siguiente orden. *Admin* → *Compute* → *Flavors*. De esta manera es posible ingresar al control de *flavors* que podemos utilizar para manipular las características que podemos asignar a las instancias en la IAAS, tal y como se muestra en la figura 128

Figura 128 Menú Flavors



De esta manera se ha podido indicar los principales menús que podrán ser utilizados por los usuarios que ingresen a la IAAS mediante la interfaz Web.

1.29.2 Acceso Interfaz CLI

Para acceder a la administración de la IAAS, existen varios métodos que pueden ser utilizados para este propósito, Conexión mediante protocolo SSH, u obtener una sección mediante la interfaz propia del sistema. Para obtener el acceso mediante SSH se muestra el proceso en la figura 129

Figura 129 Acceso SSH

```

> ssh -XY zoid@192.168.1.40
zoid@192.168.1.40's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-80-generic x86_64)

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

System information as of Sat 25 Sep 2021 07:07:34 PM CEST

System load:  2.97           Users logged in:      1
Usage of /:   74.8% of 107.28GB IPv4 address for br-ex: 172.24.4.1
Memory usage: 53%           IPv6 address for br-ex: 2001:db8::2
Swap usage:   62%           IPv4 address for ens33: 192.168.1.40
Processes:   484            IPv4 address for virbr0: 192.168.122.1

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

98 updates can be applied immediately.
46 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

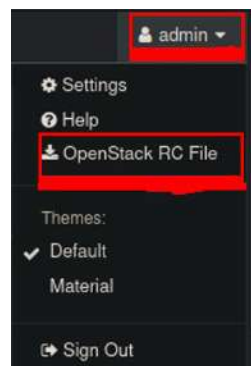
Last login: Sat Sep 25 12:02:30 2021 from 192.168.1.111
zoid@openstack:~$

```

Una vez que se tiene acceso a una terminal es necesario obtener los permisos para lograr una autenticación en la administración de la IAAS ya que es un sistema separado al sistema operativo donde esta instalada. Para esto se requiere que se ejecute un script cual habilita al usuario para poder gestionar la IAAS.

Para esto ingresamos al directorio donde tenemos el script de autenticación correspondiente al usuario y al proyecto, que se desea administrar. El fichero puede ser descargado de la interfaz web tal y como se muestra en la figura 130

Figura 130 Descarga RC File.



Una vez Descargado y transferido el Fichero hacia la máquina se ejecuta como se muestra en la figura 131, habilitando a la sesión poder realizar cambios en la IAAS. el Comando utilizar es el siguiente: **source <Fichero RC>**

Figura 131 Script RC.

```
zoid@openstack:~/Downloads$ source UTN_Hanstap-openrc.sh
Please enter your OpenStack Password for project UTN_Hanstap as user admin:
zoid@openstack:~/Downloads$
```

Una vez habilitado el usuario es posible verificar las mismas configuraciones revisadas en la interfaz Web.

Acceso imágenes.

Para verificar las imágenes y observar las opciones que se tiene para la gestión de las imágenes se utiliza el siguiente comando. **Openstack image list** como se presenta en la figura 132

Figura 132 imágenes CLI.

```
zoid@openstack:~/Downloads$ openstack image list
/usr/lib/python3/dist-packages/secretstorage/dhcrypto.py:15: CryptographyDeprecationWarning:
from cryptography.utils import int_from_bytes
/usr/lib/python3/dist-packages/secretstorage/util.py:19: CryptographyDeprecationWarning:
from cryptography.utils import int_from_bytes
+-----+-----+-----+
| ID                | Name                | Status |
+-----+-----+-----+
| d8153859-42b5-490f-ac04-d00cb53b1e43 | UTN_IMG             | active |
| 2361ab8c-102c-4848-9ce9-6c8aaaaed47c | cirros-0.5.2-x86_64-disk | active |
+-----+-----+-----+
```

De la misma manera si verificamos las opciones del comando **openstack image** , es posible verificar las opciones que se tiene para la gestión de las imágenes como se aprecia en la figura 133.

Figura 133 Opciones imágenes.

```
zoid@openstack:~/Downloads$ openstack image
add      create  delete  list    member  remove  save    set     show    unset
```

Acceso Instancias.

Para acceder a las instancias se utiliza el siguiente comando. **Openstack server list**, y se obtiene un resultado como el que se aprecia en la figura 134.

Figura 134 Lista Instancias.

```
zoid@openstack:~/Downloads$ openstack server list
/usr/lib/python3/dist-packages/secretstorage/dhcrypto.py:15: CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes
  from cryptography.utils import int_from_bytes
/usr/lib/python3/dist-packages/secretstorage/util.py:19: CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes
  from cryptography.utils import int_from_bytes
```

ID	Name	Status	Networks	Image	Flavor
80cb4791-8d5e-407e-92b7-7d865867ec98	UTN_BM_1	ACTIVE	public=172.24.4.181, 2001:db8::365	UTN_IMG	UTN

De esta manera es posible apreciar las configuraciones que tiene cada una de las instancias existentes, y las opciones que permite realizar se visualizan mediante el comando **openstack server <tab>**, como se observa en la figura 135.

Figura 135 Gestión Instancias.

```
zoid@openstack:~/Downloads$ openstack server
add          dump         image        migrate_confirm  reboot      resize       resume       ssh           unlock       unshelve
backup      evacuate    list         migrate_revert   rebuild    resize_confirm set          start        unpause     volume
create      event       lock         migration        remove     resize_revert shelve      stop         unrescue
delete     group       migrate      pause            rescue     restore      show        suspend     unset
```

Así se observa todos los comandos y opciones para aplicar con las instancias.

Acceso Redes

Para visualizar las redes se utiliza el comando, **openstack network list**, donde se muestran las redes indicadas en la figura 136

Figura 136 Redes IAAS.

```
zoid@openstack:~/Downloads$ openstack network list
/usr/lib/python3/dist-packages/secretstorage/dhcrypto.py:15: CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes
  from cryptography.utils import int_from_bytes
/usr/lib/python3/dist-packages/secretstorage/util.py:19: CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes
  from cryptography.utils import int_from_bytes
```

ID	Name	Subnets
01b506a3-bcd1-4687-aa56-1ea94170ba5a	shared	f26b11e0-1737-49c9-ae8-718e5602b021
786b7be7-6acc-4c9d-9789-baf93b2d7928	private	54c2de3f-17a7-496e-b05d-561d0dbf9b61, cb714966-4618-40e9-a348-a1c3bd3cb29c
814a9144-9328-4eba-b59d-482b2d2d24b1	public	72c932ee-06b4-4eb0-bc11-1ac69986222c, eb95a3e9-8588-4c0d-ac7b-e29d27ee6c25

Visualización Proyectos.

Para la visualización de los proyectos existentes en el sistema se utiliza el comando, **openstack projects list**. Como se muestra en la figura 137

Figura 137 Proyectos.

```
zoid@openstack:~/Downloads$ openstack project list
/usr/lib/python3/dist-packages/secretstorage/dhcrypto.py:1
from cryptography.utils import int_from_bytes
/usr/lib/python3/dist-packages/secretstorage/util.py:19: C
from cryptography.utils import int_from_bytes
+-----+
| ID | Name |
+-----+
| 0dae557381c547b89dd2d4162788b2a3 | UTN_Hanstap |
| 4afd3285a531464db4f847283d1b92f5 | demo |
| 5111244720a2470a8226f22c33845623 | alt_demo |
| 96c28e68db324fa9ac55a3a99db7fb49 | admin |
| 9cc8f77fe0754d11ad80df61cb0ad203 | invisible_to_admin |
| b0f21f3f473a4830b6cbc7e990bae093 | service |
+-----+
zoid@openstack:~/Downloads$
```

Para observar los comandos posibles para la gestión de proyectos ejecutamos el comando, **openstack project <tab>**. Mostrando los comandos posibles para la administración de proyectos, como se muestran en la figura 138

Figura 1388 Comandos Proyectos.

```
zoid@openstack:~/Downloads$ openstack project
cleanup create delete list purge set show
zoid@openstack:~/Downloads$ openstack project
```

Visualización Flavors

Para visualizar los Flavors que están activos en el sistema se utiliza el comando, **openstack flavor list**, como se muestra en la figura 139.

Figura 13939 Flavors.


```

zoid@openstack:~/Downloads$ openstack flavor list
/usr/lib/python3/dist-packages/secretstorage/dhcrypto.py:15: CryptographyDeprecationWarning:
from cryptography.utils import int_from_bytes
/usr/lib/python3/dist-packages/secretstorage/util.py:19: CryptographyDeprecationWarning:
from cryptography.utils import int_from_bytes
+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | RAM | Disk | Ephemeral | VCPUs | Is Public |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | m1.tiny | 512 | 1 | 0 | 1 | True |
| 2 | m1.small | 2048 | 20 | 0 | 1 | True |
| 3 | m1.medium | 4096 | 40 | 0 | 2 | True |
| 4 | m1.large | 8192 | 80 | 0 | 4 | True |
| 42 | m1.nano | 128 | 1 | 0 | 1 | True |
| 5 | m1.xlarge | 16384 | 160 | 0 | 8 | True |
| 84 | m1.micro | 192 | 1 | 0 | 1 | True |
| c1 | cirros256 | 256 | 1 | 0 | 1 | True |
| d1 | ds512M | 512 | 5 | 0 | 1 | True |
| d2 | ds1G | 1024 | 10 | 0 | 1 | True |
| d3 | ds2G | 2048 | 10 | 0 | 2 | True |
| d4 | ds4G | 4096 | 20 | 0 | 4 | True |
| utn | UTN | 2048 | 15 | 0 | 2 | True |
+-----+-----+-----+-----+-----+-----+-----+

```

De esta manera se ha mostrado las diferentes formas de realizar la administración por parte de los usuarios considerando las principales características de la IAAS por medio de sus métodos de conexión.

1.30 Guía 2 Creación Flavor e Instancias

La presente guía da a conocer el procedimiento para la creación de una flavor de características de una máquina para ser utilizado en la creación de una posterior instancia.

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN ELECTRÓNICA Y REDES
DE COMUNICACIÓN



TEMA: Guía Creación Flavors e Instancias.

AUTOR: SANTIAGO ESTÉVEZ.

En la presente práctica se indicará el proceso de generación de Flavor y creación de una instancia en la IAAS. Indicando paso a paso su generación mediante Interfaz Web.

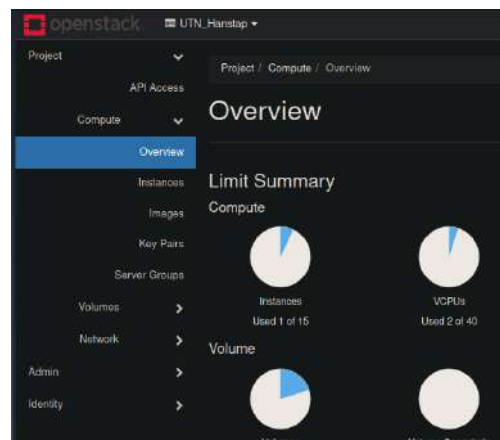
Objetivos

- Generación de un flavor (Características virtuales) de una instancia en la plataforma OpenStack para su posterior utilización.
- Creación de una instancia virtual utilizando el flavor generado anteriormente.

1.30.1 Creación Flavor

Paso 1. Se ingresa a la Interfaz Web de la IAAS. Con las credenciales indicadas por el Administrador, como se muestra en la figura 140, de esta manera se tiene los permisos para poder desarrollar las funciones administrativas.

Figura 140 Ingreso IAAS.



Paso 2. En el menú de *Flavors* Ubicado en: *Admin* → *Compute* → *Flavors*, mostrado en la figura 141, se selecciona el botón “*Create Flavor*”, el cual abrirá el asistente de creación de flavor mostrado en la figura 142, se establecen las características deseadas, para la presente guía se utilizarán los valores

Figura 141 Flavors.

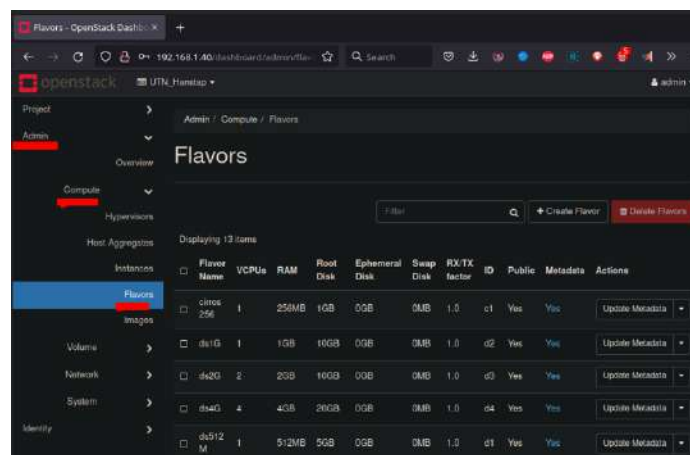


Figura 142 Asistente Flavor.

Una vez determinados los valores deseados para el Flavor, se da click en el botón “*Create Flavor*”, Se confirma la creación observando que se agregó a la lista de Flavors como se muestra en la Figura 143.

Figura 143 Creación Flavor.

Name	VCPUs	RAM (MB)	Root Disk (GB)	Ephemeral Disk (GB)	Swap Disk (MB)	RX/TX Factor	ID	Access
m1.nano	1	192MB	1GB	0GB	0MB	1.0	84	Yes
m1.nano	1	128MB	1GB	0GB	0MB	1.0	42	Yes
m1.small	1	2GB	20GB	0GB	0MB	1.0	2	Yes
m1.tiny	1	512MB	1GB	0GB	0MB	1.0	1	Yes
m1.xlarge	8	16GB	160GB	0GB	0MB	1.0	5	Yes
UTN	2	2GB	15GB	0GB	0MB	1.0	utn	Yes
UTN_Flavor	4	2.4GB	15GB	0GB	0MB	1.0	b605b48b-ea54-498f-bf79-5b6d809dbb05	No

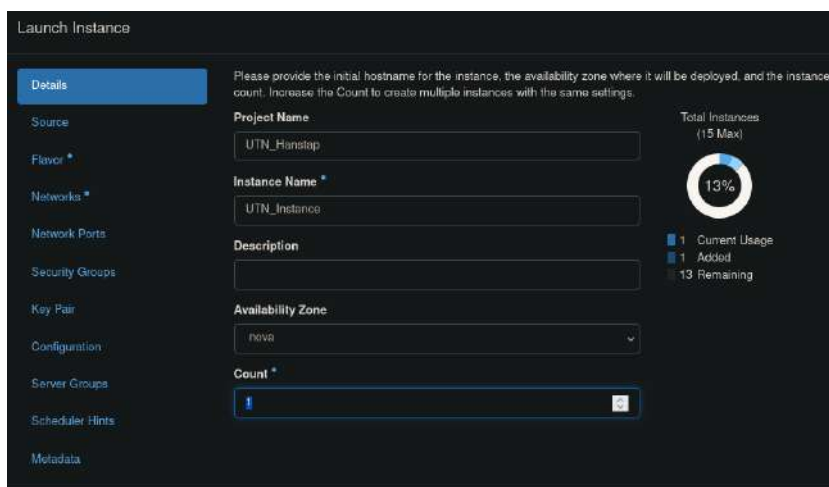
Displaying 14 items

Con lo indicado es posible crear un flavor para utilizarlo en la generación de instancias. Como se desarrollarán a continuación.

1.30.2 Creación Instancia.

Paso 1. Es necesario dirigirse hacia el menú de instancias en *Project* → *compute* → *Instances*, para de esta manera utilizar el asistente como se muestra en la figura 144, en la cual se agregan del nombre de la instancia, así como la cantidad de instancias que se requiere crear. en la figura 145 se muestra la selección del Flavor a utilizar para la misma.

Figura 144 Creación Instancia.



The screenshot displays the 'Launch Instance' wizard interface. On the left, a sidebar lists various configuration options: Details (selected), Source, Flavor, Networks, Network Ports, Security Groups, Key Pair, Configuration, Server Groups, Scheduler Hints, and Metadata. The main area contains the following fields and controls:

- Project Name:** A text input field containing 'UTN_Horistap'.
- Instance Name:** A text input field containing 'UTN_Instance'.
- Description:** An empty text input field.
- Availability Zone:** A dropdown menu currently set to 'nova'.
- Count:** A text input field containing '1', with a small icon to its right.

On the right side of the main area, there is a circular progress indicator showing '13%' completion. Below it, a legend indicates: '1 Current Usage' (blue square), '1 Added' (light blue square), and '13 Remaining' (grey square). Above the progress indicator, the text reads 'Total Instances (15 Max)'.

Figura 1455 Selección Flavor

Launch Instance

Flavors manage the sizing for the compute, memory and storage capacity of the instance.

Allocated

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
UTN_Flavor	4	2.44 GB	15 GB	15 GB	0 GB	No

Available (13)

Select one

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
m1.nano	1	128 MB	1 GB	1 GB	0 GB	Yes
m1.micro	1	192 MB	1 GB	1 GB	0 GB	Yes
cirros256	1	256 MB	1 GB	1 GB	0 GB	Yes
m1.tiny	1	512 MB	1 GB	1 GB	0 GB	Yes
ds512M	1	512 MB	5 GB	5 GB	0 GB	Yes

Paso 2. De esta manera es posible utilizar cualquier Flavor que se tiene en la lista, para esta instancia se utilizará el Flavor creado anteriormente, pero no está limitado a utilizar únicamente este. A continuación, se realiza la selección de la red a la cual la instancia generada estará conectada. Como se aprecia en la figura 146

Figura 1466 Selección de Red

Launch Instance

Networks provide the communication channels for instances in the cloud.

Allocated (1)

Select networks from those listed below.

Network	Subnets Associated	Shared	Admin State	Status
public	ipv6-public-subnet public-subnet	Yes	Up	Active

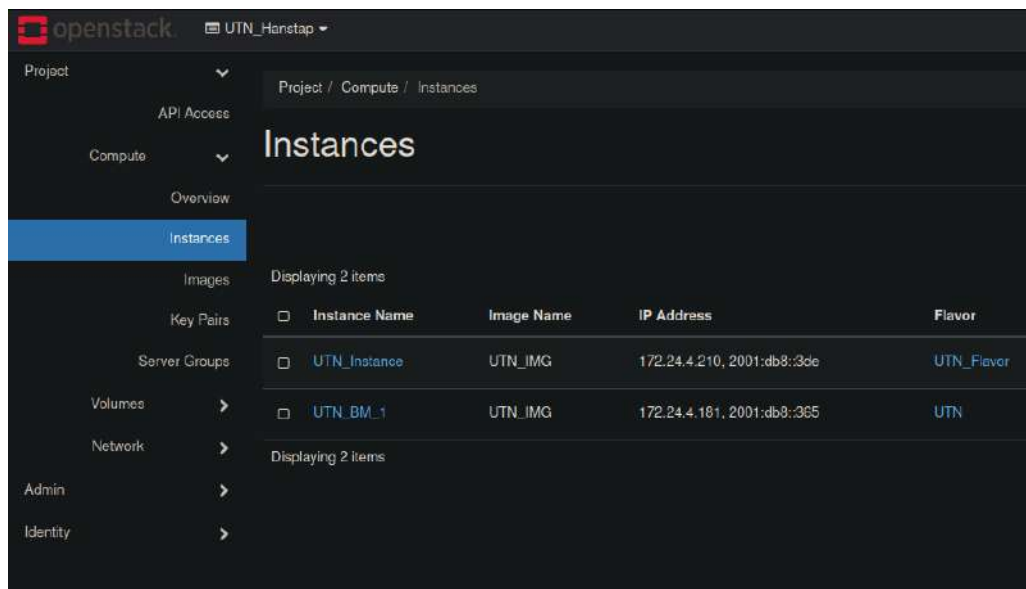
Available (1)

Select at least one network

Network	Subnets Associated	Shared	Admin State	Status
shared	shared-subnet	Yes	Up	Active

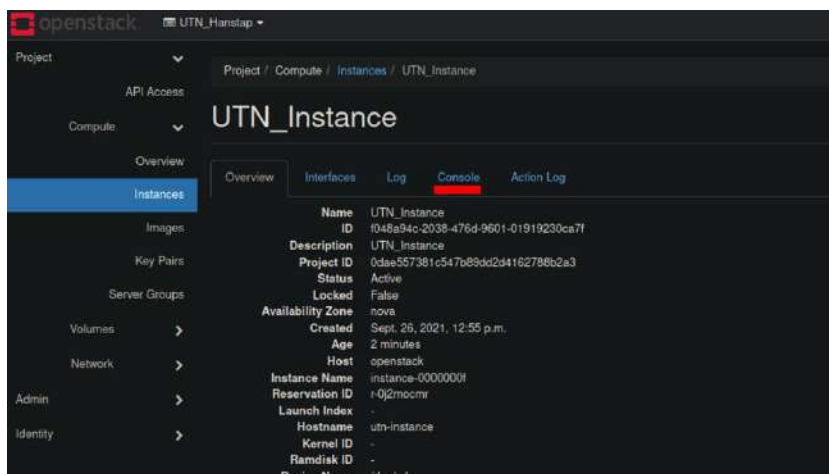
Paso 3. El resto de las configuraciones pueden dejarse por defecto y se da en el botón de “**Create Instance**”, permitiendo la creación de la instancia y listándose en el menú como se muestra en la figura 147.

Figura 147 Creación Instancia.



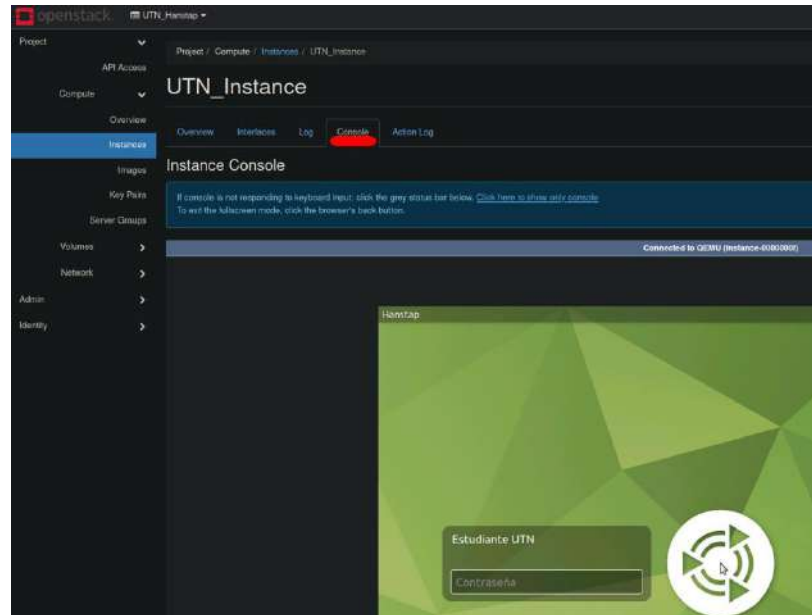
Paso 4. Para obtener acceso a la instancia generada podemos dar click en el nombre de la instancia obteniendo un menú como el que se muestra en la Figura 148

Figura 148 Opciones Instancia.



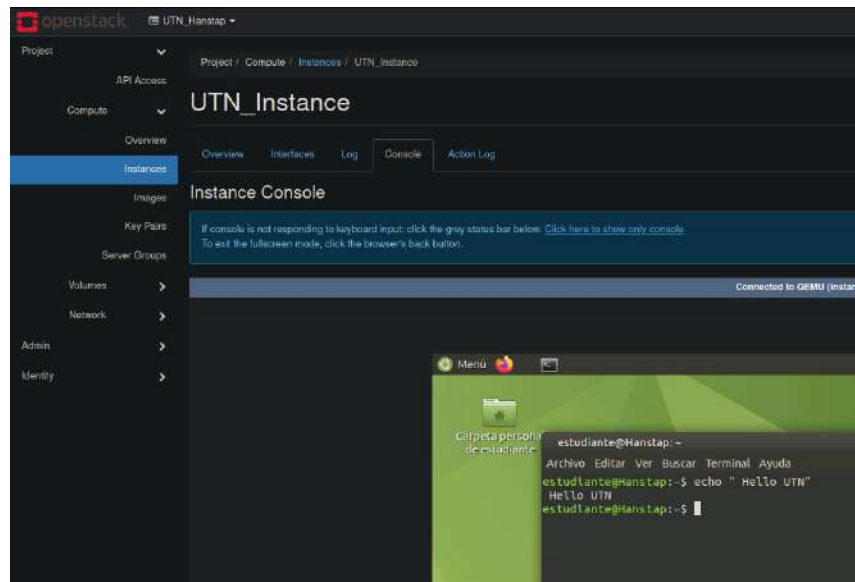
Paso 5. Una vez en el menú es posible seleccionar la opción “**Console**” la cual permite acceder a la interfaz de la Instancia y Continuar con su administración. Como se aprecia en la Figura 149

Figura 149 Instancia Activa.



De esta manera ingresando las credenciales proporcionadas por el administrador es posible ingresar a la instancia. Tal y como se muestra en la figura 150

Figura 1500 Ingreso Instancia



1.31 Guia 3 Desarrollo SDN/NFV Instancia.

La presente guía muestra el procedimiento para realizar una práctica sobre tecnologías SDN y NFV, en la cual se utilizará las herramientas y servicios instalados en la instancia generada sobre la plataforma OpenStack.

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN ELECTRÓNICA Y REDES
DE COMUNICACIÓN



TEMA: Práctica SDN/NFV.

AUTOR: SANTIAGO ESTÉVEZ.

La presente guía presenta el desarrollo de una práctica en las tecnologías SDN/NFV utilizando una instancia presente en la IAAS.

Para el desarrollo de la práctica se requiere haber obtenido las credenciales otorgadas por el administrador de la IAAS, para poder acceder a la instancia.

Objetivos.

- Familiarizarse con la inicialización de los servicios y aplicaciones ofrecidas en la instancia
- Generar topología SDN utilizando instancias NFV de OVS mediante mininet.
- Verificación de comunicación y de convergencia de la red utilizando ONOS

1.31.1 Acceso

Paso 1. Para ingresar a la instancia se ingresa mediante la ip asignada por el administrador de esta manera se obtiene una ventana como la que se muestra en la figura 151, una vez que se obtiene la interfaz se debe acceder mediante las credenciales dadas por el administrador. Garantizando el acceso a la interfaz gráfica del sistema como se aprecia en la figura 152.

Figura 1511 Instancia Estudiante

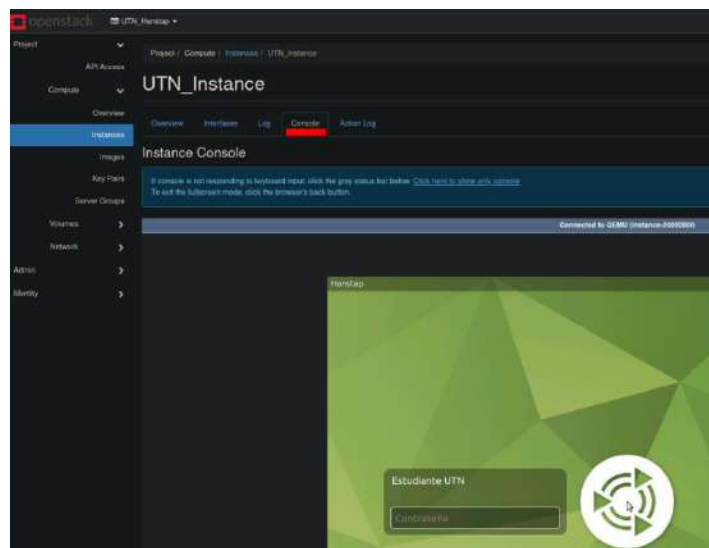


Figura 1522 Interfaz Gráfica Instancia.



Paso 2. Es necesario activar los servicios a utilizar dentro del sistema para realizar la práctica. En primer lugar, se enciende el servicio de OVS mediante la inicialización del script del demonio como se muestra en la figura 152, de esta manera mininet podrá generar las instancias NFV de OVS a ser utilizadas como switch virtuales en cualquier topología. Se enciende el servicio de la controladora de SDN como se muestra en la figura 153, a continuación, es necesario ingresar a la interfaz web de la controladora ONOS ingresando al link <http://localhost:8181/onos/ui> como se muestra en la figura 154

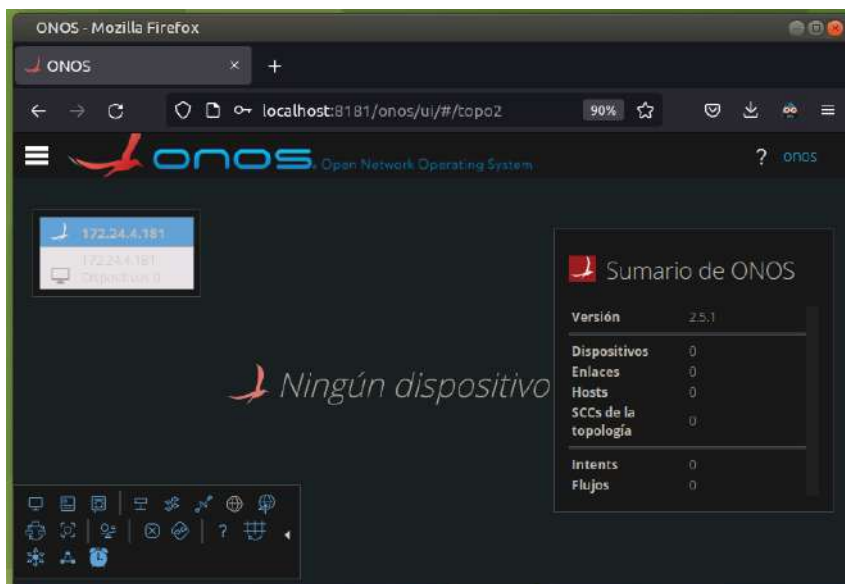
Figura 1532 Inicialización OVS

```
estudiante@Hanstap:/usr/local/share/openvswitch/scripts$ sudo ./ovs-ctl start
* ovsdb-server is already running
* Starting ovs-vswitchd
* Enabling remote OVSDB managers
```

Figura 1543 Inicialización ONOS.

```
estudiante@Hanstap:~/ONOS/onos-2.5.1/bin$ ./onos-service start
sep. 26, 2021 2:16:38 P. M. org.apache.karaf.main.lock.SimpleFileLock lock
INFORMACIÓN: Trying to lock /home/estudiante/ONOS/onos-2.5.1/apache-karaf-4.2.9/lock
sep. 26, 2021 2:16:38 P. M. org.apache.karaf.main.lock.SimpleFileLock lock
INFORMACIÓN: Lock acquired
sep. 26, 2021 2:16:38 P. M. org.apache.karaf.main.Main$KarafLockCallback lockAcquired
INFORMACIÓN: Lock acquired. Setting startlevel to 100
14:16:38.831 INFO [EventAdminConfigurationNotifier] Sending Event Admin notification
successful) to org/ops4j/pax/logging/Configuration
```

Figura 155 Interfaz ONOS.



Paso3. Con los servicios debidamente activados es posible continuar con el desarrollo de la práctica, para esto se utilizará el software minnet, el cual generará una topología en árbol mediante el comando `mn -controller=remote -switch=default,protocols=openflow13 -topo=tree,2,2.`, especificando el tipo de topología con el argumento `topo=tree,2,2`, utilizando como protocolo de comunicación Openflow mediante el argumento `-switch=default,protocols=openflow13`, al ser ejecutado el resultado es mostrado en la figura 155, misma que puede ser observada gráficamente en la controladora ONOS apreciable en la figura 156, en la cual se aprecia las 3 instancias NFV de los switch virtuales OVS, y los host asociados a cada uno para comprobar su convergencia.

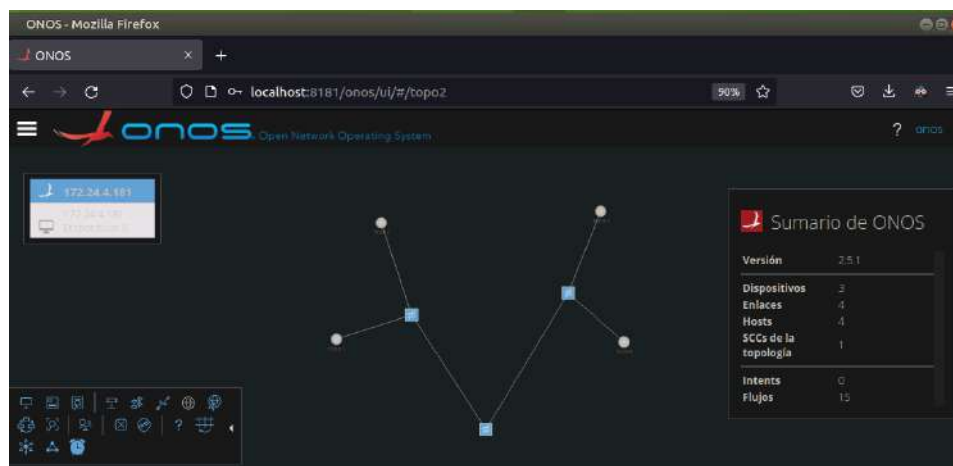
Figura 15655 Resultado Script.

```

estudiante@Hanstap:~/ONOS/onos-2.5.1/bin$ sudo mn --controller=remote,ip=127.0.0.1 --switch=
tch=default,protocols=OpenFlow13 --topo tree,2,2
[sudo] contraseña para estudiante:
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, s2) (s1, s3) (s2, h1) (s2, h2) (s3, h3) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>

```

Figura 157 Topología.



Paso 4. Una vez la topología es establecida, se realiza un análisis de las tramas enviadas SDN al establecer un ping entre todos los dispositivos, tal y como se muestran en la figura 157, donde es posible apreciar el tratamiento de tramas en el plano de control mediante la existencia de tramas OpenFlow, mismas que indican el correcto funcionamiento de la controladora SDN, mediante las banderas activas de

OFPT_PACKET_IN y *OFPT_PACKET_OUT*, de esta manera la controladora puede tomar la decisión y comandar al plano de control la ruta que el paquete debe tomar para llegar a su destino. Como es apreciable en la figura 159.

Figura 158 Comunicación Openflow

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	OpenFlow	218	Type: OFPT_PACKET_OUT
2	0.000143448	127.0.0.1	127.0.0.1	TCP	68	6653 → 6653 [ACK] Seq=3 Ack=500 Win=86 Len=0 TSval=249679515...
3	0.000307423	02:eb:8f:07:c9:42	127.0.0.1	LLDP	141	MA/08:00:08:00:00:01 PC/21 129
4	0.000312661	02:eb:8f:07:c9:42	127.0.0.1	LLDP	141	MA/08:00:08:00:00:01 PC/21 129
5	0.000364618	127.0.0.1	127.0.0.1	OpenFlow	249	Type: OFPT_PACKET_IN
6	0.0009231970	127.0.0.1	127.0.0.1	TCP	68	6653 → 36158 [ACK] Seq=1 Ack=182 Win=86 Len=0 TSval=249679515...
7	0.000768182	127.0.0.1	127.0.0.1	OpenFlow	247	Type: OFPT_PACKET_OUT
8	0.001049126	127.0.0.1	127.0.0.1	TCP	68	6654 → 6653 [ACK] Seq=1 Ack=359 Win=86 Len=0 TSval=249679515...
9	0.002174679	02:eb:8f:07:c9:42	127.0.0.1	0x8942	141	Sent by us
10	0.002175246	02:eb:8f:07:c9:42	127.0.0.1	0x8942	141	Broadcast
11	0.002262658	127.0.0.1	127.0.0.1	OpenFlow	247	Type: OFPT_PACKET_OUT
12	0.002286705	127.0.0.1	127.0.0.1	TCP	68	6654 → 6653 [ACK] Seq=1 Ack=538 Win=86 Len=0 TSval=249679515...
13	0.002323319	127.0.0.1	127.0.0.1	OpenFlow	249	Type: OFPT_PACKET_IN
14	0.002329761	127.0.0.1	127.0.0.1	TCP	68	6653 → 36158 [ACK] Seq=1 Ack=353 Win=86 Len=0 TSval=249679515...
15	0.002451895	02:eb:8f:07:c9:42	127.0.0.1	LLDP	141	MA/08:00:08:00:00:01 PC/22 129
16	0.002483658	02:eb:8f:07:c9:42	127.0.0.1	LLDP	141	MA/08:00:08:00:00:01 PC/22 129

Figura 159 Gestión de paquetes openflow mediante la controladora



De la misma manera en el plano de datos el cual tiene a cargo de las instancias NFV de OVS, permite el reenvío de paquetes dentro de la red como es apreciable en la figura 160 donde se aprecian paquetes *ICMP*, mismos que son intercambiados entre los 2 hosts de la topología.

Figura 160 Secuencia Ping

159	4.0810611420	127.0.0.1	10.0.0.2	ICMP	100 Echo (ping) request id=0x3f7a, seq=1/256, ttl=64 (no respons...
160	4.083607642	127.0.0.1	127.0.0.1	OpenFlow	206 Type: OFPT_PACKET_IN
161	4.083619103	127.0.0.1	127.0.0.1	TCP	68 6653 → 36158 [ACK] Seq=2180 Ack=7441 Win=86 Len=0 TSval=24967...
162	4.103406819	127.0.0.1	127.0.0.1	OpenFlow	206 Type: OFPT_PACKET_OUT
163	4.103422593	127.0.0.1	127.0.0.1	TCP	68 36158 → 6653 [ACK] Seq=7441 Ack=2327 Win=80 Len=0 TSval=24907...
164	4.103066761	10.0.0.1	10.0.0.2	ICMP	100 Echo (ping) request id=0x3f7a, seq=1/256, ttl=64 (reply in L...
165	4.104755893	10.0.0.2	10.0.0.1	ICMP	100 Echo (ping) reply id=0x3f7a, seq=1/256, ttl=64 (request in...
166	4.108483716	127.0.0.1	127.0.0.1	OpenFlow	208 Type: OFPT_PACKET_IN
167	4.108496299	127.0.0.1	127.0.0.1	TCP	68 6653 → 36158 [ACK] Seq=2327 Ack=7581 Win=86 Len=0 TSval=24967...
168	4.122088366	127.0.0.1	127.0.0.1	OpenFlow	206 Type: OFPT_PACKET_OUT
169	4.122155761	127.0.0.1	127.0.0.1	TCP	68 36158 → 6653 [ACK] Seq=7581 Ack=2465 Win=86 Len=0 TSval=24967...
170	4.122495963	10.0.0.2	10.0.0.1	ICMP	100 Echo (ping) reply id=0x3f7a, seq=1/256, ttl=64

1.32 Guia 4 Práctica NFV - OPNSense

La presente guía muestra la integración de una instancia NFV de firewall OPNSense e instancias NFV OVS en una topología SDN generada mediante mininet.

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN ELECTRÓNICA Y REDES
DE COMUNICACIÓN



TEMA: Integración Servicios NFV

AUTOR: SANTIAGO ESTÉVEZ.

La presente guía desarrolla de una práctica en la cual se utilizará instancias NFV tanto para de firewall OPNSense como de Switch OVS, en una topología SDN generada mediante mininet.

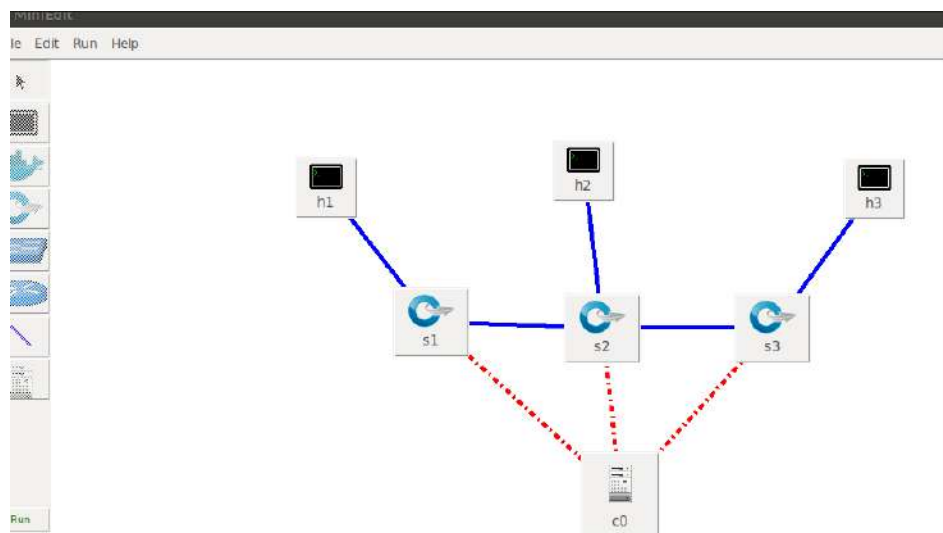
Objetivos.

- Generar una topología utilizando mininet
- Integración de instancia NFV Firewall
- Obtener información sobre la topología mediante la controladora onos

1.32.1 Generación Topología

Una vez ingresado a la instancia virtual, se ejecuta el editor gráfico de mininet mediante el comando, `sudo python3 ~/SDN/Containernet/examples/miniedit.py` mismo que permite generación gráfica de la topología, obteniendo su interfaz gráfica, misma que consta de 3 instancias NFV de OVS, una controladora SDN y 3 host virtuales asociados a cada uno de los switches como se muestra en la figura 161

Figura 161 Topología de Integración



Para realizar la integración de la instancia NFV de firewall OPNSense, es necesario generar una interfaz *tap* en el sistema misma a la que se generará un *bridge* para poder ser conectada al switch 1. El procedimiento es el siguiente; generar la interfaz

mediante el comando `ip tuntap add tap0 mode tap` mediante el cual se genera la interfaz.

La interfaz debe ser activada mediante el comando `ip link set dev tap0 up`,

Para establecer una conexión entre hacia la instancia OVS, se requiere una interfaz bridge misma que se crea mediante el comando `ip link add virbr1 type bridge`,

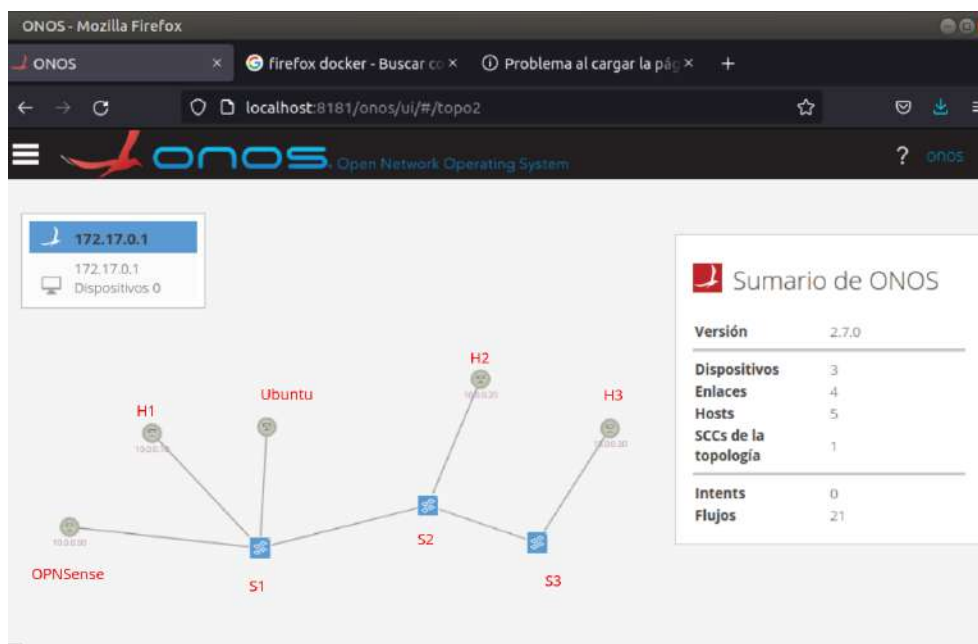
Para agregar este puente al switch OVS se el comando `sudo ovs-vsctl add-port s1 virbr1`, asociando el puerto en el Switch S1 para acceder a la red SDN ,su verificación se confirma mediante el comando `ovs-vsctl show`, el cual muestra las interfaces de cada switch OVS generado y constatamos la presencia del bridge virbr1 como puerto del switch S1 como se aprecia en la figura 162. Este mismo proceso puede ser utilizado para agregar interfaces de red USB para conexión a sistemas remotos.

Figura 162 Configuración OVS

```
OVS Summary (como superusuario)
2e038040-7c47-46e2-b090-7e790578a0d3
Bridge s1
  Controller "tcp:172.17.0.1:6633"
  is_connected: true
  fail_mode: secure
  Port virbr1
  Interface virbr1
  Port s1-eth2
  Interface s1-eth2
  Port s1-eth1
  Interface s1-eth1
  Port s1
  Interface s1
  type: internal
Bridge s2
  Controller "tcp:172.17.0.1:6633"
  is_connected: true
  fail_mode: secure
  Port s2-eth2
  Interface s2-eth2
  Port s2-eth1
  Interface s2-eth1
  Port s2-eth3
  Interface s2-eth3
  Port s2
  Interface s2
  type: internal
Bridge s3
  Controller "tcp:172.17.0.1:6633"
  is_connected: true
  fail_mode: secure
  Port s3-eth2
  Interface s3-eth2
  Port s3
  Interface s3
  type: internal
  Port s3-eth1
  Interface s3-eth1
  ovs_version: "2.13.5"
Press Enter to close
```

Al observar la topología SDN en la controladora SDN ONOS indicada en la figura 163, se puede verificar la presencia de los dispositivos host, Firewall y los switches OVS, demostrando la convergencia e integración del servicio NFV de la red.

Figura 163 Convergencia red SDN y firewall



1.32.2 Integración SDN

Para comprobar el funcionamiento se realiza un ping entre los hosts internos de la red SDN desde la instancia NFV de firewall localizado en el switch S1 hacia el host H3 en la ip $10.0.0.30$ en el switch S3, obteniendo una respuesta correcta como se indica en la figura 164

Figura 164 Ping Firewall Host H3

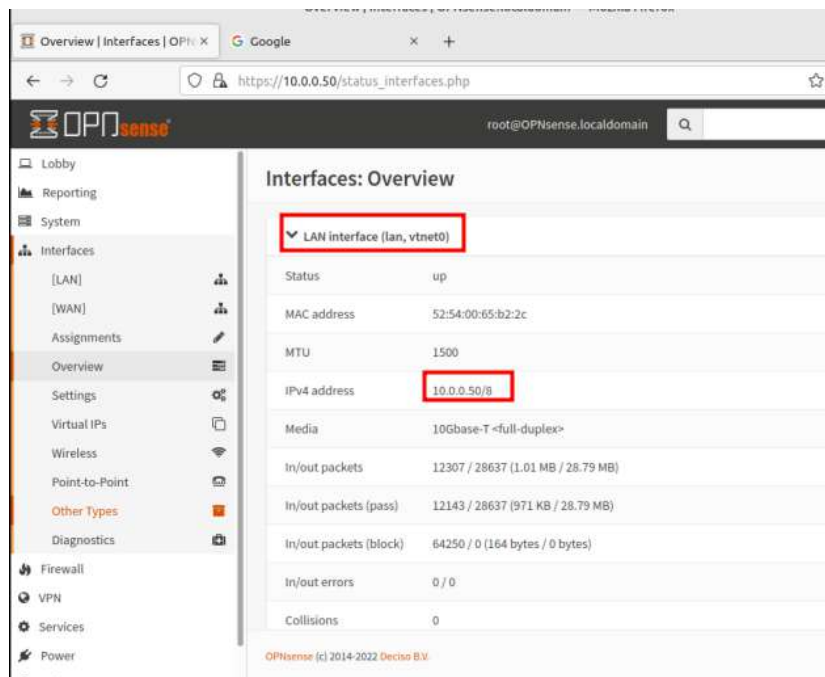
```
root@OPNsense:~ # ping 10.0.0.30
PING 10.0.0.30 (10.0.0.30): 56 data bytes
64 bytes from 10.0.0.30: icmp_seq=0 ttl=64 time=0.475 ms
64 bytes from 10.0.0.30: icmp_seq=1 ttl=64 time=0.548 ms
64 bytes from 10.0.0.30: icmp_seq=2 ttl=64 time=0.478 ms
64 bytes from 10.0.0.30: icmp_seq=3 ttl=64 time=0.497 ms
```

Comprobando el funcionamiento y configuración del firewall en la red, se verifica las interfaces conectadas al firewall, obteniendo como red WAN la red interna de la instancia virtual sobre la cual esta generado el firewall de subred $192.168.122.0/24$ y como red LAN la red interna SDN de subred $10.0.0.0/8$ apreciables en las figuras 165 y 166 respectivamente.

Figura 165 red WAN del OPNSense

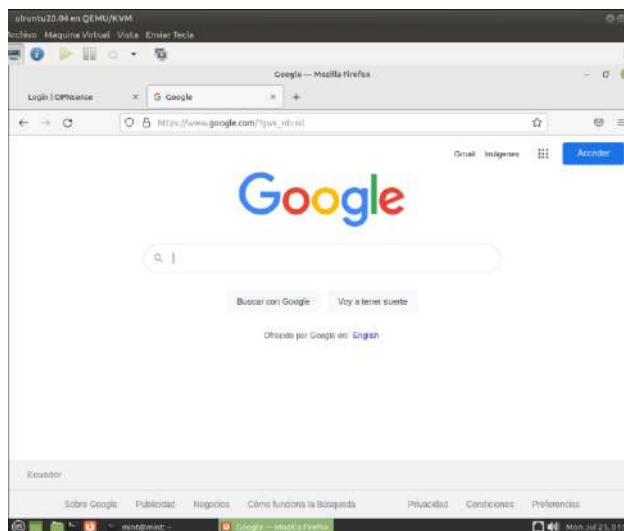
▼ WAN interface (wan, vtnet1)	
Status	up
DHCP	up <input type="button" value="Reload"/> <input type="button" value="Release"/>
MAC address	52:54:00:8e:d4:e9
MTU	1500
IPv4 address	192.168.122.124/24
IPv4 gateway	192.168.122.1

Figura 166 Red LAN OPNSense



Comprobando el funcionamiento del firewall en la topología se realiza una solicitud desde uno del host hacia el dominio www.google.com en internet, obteniendo la página de búsqueda como se aprecia en la figura 167, esto permite desarrollar un análisis del comportamiento de la red SDN al obtener paquetes desde un firewall y su tratamiento de estos al llegar a la red SDN.

Figura 167 Solicitud internet en la red SDN



Es posible observar el comportamiento de la red siendo la controladora SDN la que dirige el plano de datos mediante las banderas de *OFPT_PACKET_IN* y *OFPT_PACKET_OUT* las cuales informan a la controladora del tráfico existente en la red y determina su camino a seguir en el plano de datos para llegar a su destino como se muestra en la figura 168 y 169 donde envía las órdenes a los swiches del plano de datos utilizando un *OFPT_STATS_REPLY* la cual indica el direccionamiento por parte de la controladora..

Figura 168 Gestión tráfico WAN en Controladora ONOS

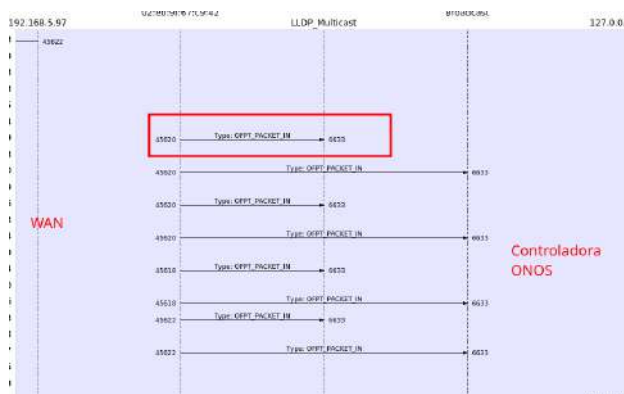
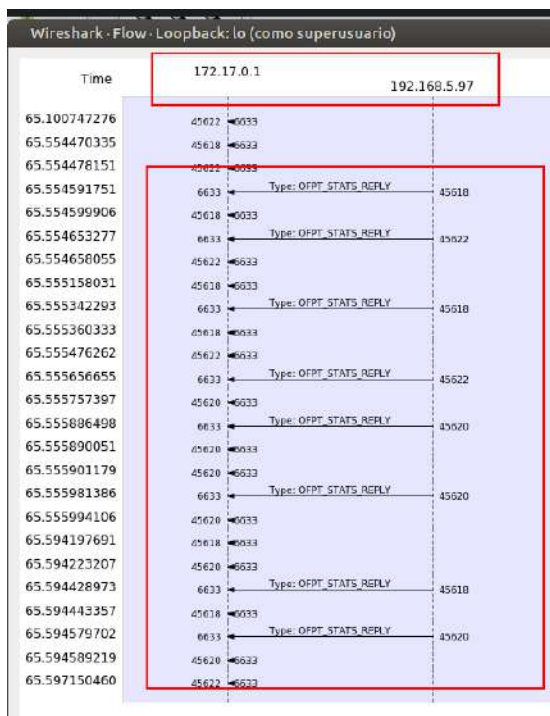


Figura 169 Respuesta de controladora



1.33 Guia 4 Estadísticas

La presente guía muestra la información y estadísticas de la red, las cuales pueden ser obtenidas mediante la controladora ONOS mientras esta controla una topología.

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN ELECTRÓNICA Y REDES
DE COMUNICACIÓN



TEMA: Estadísticas.

AUTOR: SANTIAGO ESTÉVEZ.

La presente guía desarrolla de una práctica utilizando opciones avanzadas de la controladora ONOS en su funcionamiento de monitoreo de tráfico dentro de la red establecida.

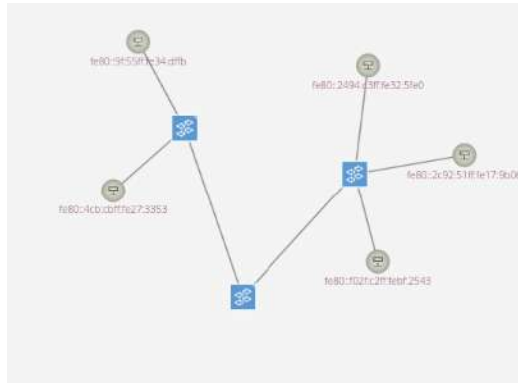
Objetivos.

- Generar una topología utilizando mininet
- Instalar aplicaciones de Estadísticas en la controladora ONOS
- Obtener información sobre la topología mediante la controladora onos

Para el desarrollo de la práctica se requiere haber obtenido las credenciales otorgadas por el administrador de la IAAS, para poder acceder a la instancia., se utiliza el comando *mininet* con los argumentos `--controller=remote, ip=127.0.0.1` para establecer la conexión hacia la controladora ONOS, para establecer el protocolo a utilizar

para la comunicación entre *ovs* y controladora se utiliza el argumento `--switch=default,protocols=OpenFlow10`, finalmente para generar una topología de árbol se utiliza el argumento `--topo tree,2,2`, de esta manera el resultado se muestra en la figura 159

Figura Topología Pruebas



1.33.1 Instalación Aplicaciones necesarias

Una vez se tiene la topología reconocida por la controladora ONOS se procede a activar las aplicaciones que se utilizarán para la monitorización en la red. Para esto en la interfaz Web de ONOS accedemos al menú aplicaciones. Mostrado en la figura 160, en el campo de búsqueda ingresamos “*monitor*” y de la lista resultante escogemos los siguientes elementos y los activamos

Figura 170 menú opciones ONOS



En el campo de búsqueda ingresamos “*monitor*” y de la lista resultante escogemos los siguientes elementos y los activamos dando click al ícono de play en su pantalla, como se muestra en la figura 161.

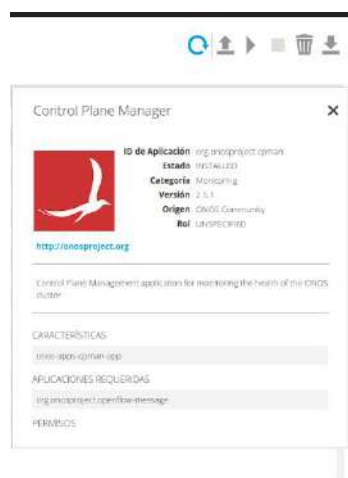
Control Plane Manager # Gestor de plano de control

Controller Monitor Application # Aplicación de control del sistema de monitoreo

Flow Space Analysis # Analizador de rutas y flujos

Topology & Intent Metrics # Gestión de topologías y métricas.

Figura 171 Activación Aplicación ONOS

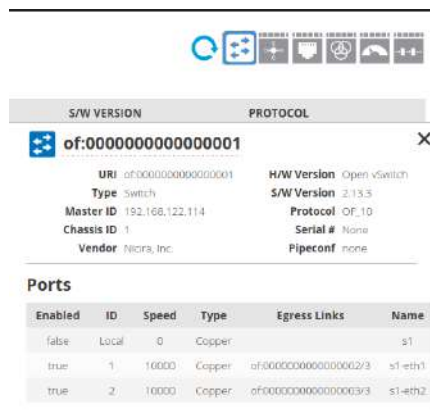


1.33.2 Revisión de estadísticas

Una vez activadas todas las aplicaciones necesarias regresamos al menú principal, se realiza un click en uno de los switches centrales y podemos acceder a sus características especiales y de monitoreo. Mostrando primeramente las características del switch como se muestra en la figura 160, especificando su protocolo de funcionamiento, en este caso OpenFlow 10, y el tipo de hardware, para este ejemplo es una instancia de OVS, además es apreciable los puertos que se encuentran activos y sus características como son id, velocidad de enlace entre otros, para

acceder a sus estadísticas de tráfico es posible acceder mediante su ítem de estadísticas, como se muestran en la figura 163

Figura 172 Características Switch



Enabled	ID	Speed	Type	Egress Links	Name
false	Local	0	Copper		s1
true	1	10000	Copper	of:0000000000000002/3	s1-eth1
true	2	10000	Copper	of:0000000000000003/3	s1-eth2

Figura 173 Estadísticas Switch y Flujos.



ESTADO	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	4500	5	0	ETH_TYPE=eth IPV4_DST=234.0.0.0	mm:OUTPUT_CONTROLLER, clearflow	*none
Added	16	4500	5	0	eth_type=arp	mm:OUTPUT_CONTROLLER, clearflow	*none
Added	34	4500	40000	0	eth_type=arp	mm:OUTPUT_CONTROLLER, clearflow	*none
Added	1450	4500	40000	0	ETH_TYPE=arp	mm:OUTPUT_CONTROLLER, clearflow	*none
Added	1450	4500	40000	0	ETH_TYPE=arp	mm:OUTPUT_CONTROLLER, clearflow	*none

De esta manera se observa el tipo de flujos, así como la configuración interna del switch de como toma sus decisiones basadas en las ordenes de la controladora. De la misma manera es posible analizar el tráfico de datos con un simple escaneo de tramas en las interfaces de cada Switch o nodo permitiendo identificar el flujo y control que tienen los paquetes bajo la controladora, así como se muestra en la figura 164, donde se puede apreciar los paquetes son enviados a negociación acorde a las órdenes de la controladora ONOS y esta dispone los mensajes de aprobación de cómo realizar los envíos de paquetes. Para aprendizaje es necesario que los estudiantes sepan reconocer comportamientos de

control y procedimientos dados por los protocolos como es apreciable se tiene las banderas de OFPT_PACKET_IN como paquete de entrada y OFPT_PACKET_OUT representando paquete de salida que se tiene en los paquetes en la comunicación entre la controladora y los switch ovs.

Figura 174 Negociación paquetes hacia destino.

The screenshot shows a Wireshark interface with a packet capture list. The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help) and a toolbar. A display filter is applied: <Ctrl-F>. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
1173	2.346264638	10.0.0.4	10.0.0.5	OpenFlow	188	Type: OFPT_PACKET_OUT
1174	2.346565548	10.0.0.4	10.0.0.5	OpenFlow	182	Type: OFPT_PACKET_IN
1175	2.346680153	10.0.0.4	10.0.0.5	OpenFlow	182	Type: OFPT_PACKET_IN
1176	2.347167814	10.0.0.4	10.0.0.5	OpenFlow	188	Type: OFPT_PACKET_OUT
1177	2.347134628	10.0.0.4	10.0.0.5	OpenFlow	188	Type: OFPT_PACKET_OUT
1178	2.347415278	10.0.0.4	10.0.0.5	OpenFlow	182	Type: OFPT_PACKET_IN
1179	2.349021698	10.0.0.4	10.0.0.5	OpenFlow	188	Type: OFPT_PACKET_OUT
1180	2.349385768	10.0.0.4	10.0.0.5	OpenFlow	182	Type: OFPT_PACKET_IN
1181	2.349394677	10.0.0.4	10.0.0.5	OpenFlow	182	Type: OFPT_PACKET_IN
1182	2.349613888	127.0.0.1	127.0.0.1	TCP	74	8181 → 56216 [PSH, ACK] Seq=1092 Ack=23 Win=86 Len=8 TSval=41...
1183	2.349891671	127.0.0.1	127.0.0.1	TCP	74	8181 → 56216 [PSH, ACK] Seq=1180 Ack=23 Win=86 Len=8 TSval=41...
1184	2.349918649	127.0.0.1	127.0.0.1	TCP	66	56216 → 8181 [ACK] Seq=23 Ack=1108 Win=86 Len=0 TSval=4183728...
1185	2.349949950	10.0.0.4	10.0.0.5	OpenFlow	188	Type: OFPT_PACKET_OUT
1186	2.349992498	10.0.0.4	10.0.0.5	OpenFlow	188	Type: OFPT_PACKET_OUT
1187	2.351969172	10.0.0.5	10.0.0.6	OpenFlow	182	Type: OFPT_PACKET_IN
1188	2.352473622	10.0.0.5	10.0.0.6	OpenFlow	188	Type: OFPT_PACKET_OUT
1189	2.352743776	10.0.0.5	10.0.0.6	OpenFlow	182	Type: OFPT_PACKET_IN
1190	2.352813524	10.0.0.6	10.0.0.5	OpenFlow	182	Type: OFPT_PACKET_IN
1191	2.353169106	10.0.0.6	10.0.0.5	OpenFlow	188	Type: OFPT_PACKET_OUT
1192	2.353374901	10.0.0.6	10.0.0.5	OpenFlow	182	Type: OFPT_PACKET_IN
1193	2.353651599	127.0.0.1	127.0.0.1	TCP	66	6053 → 36676 [ACK] Seq=7673 Ack=24637 Win=86 Len=0 TSval=4183...
1194	2.353764883	10.0.0.5	10.0.0.6	OpenFlow	188	Type: OFPT_PACKET_OUT
1195	2.353889382	10.0.0.5	10.0.0.6	OpenFlow	182	Type: OFPT_PACKET_IN
1196	2.353993936	10.0.0.6	10.0.0.5	OpenFlow	188	Type: OFPT_PACKET_OUT
1197	2.354021429	10.0.0.5	10.0.0.6	OpenFlow	182	Type: OFPT_PACKET_IN
1198	2.354062937	127.0.0.1	127.0.0.1	TCP	66	36676 → 6053 [ACK] Seq=24637 Ack=7917 Win=86 Len=0 TSval=4183...
1199	2.354156093	10.0.0.6	10.0.0.5	OpenFlow	182	Type: OFPT_PACKET_IN
1200	2.354183098	10.0.0.6	10.0.0.5	OpenFlow	182	Type: OFPT_PACKET_IN
1201	2.354276663	127.0.0.1	127.0.0.1	TCP	66	6053 → 36658 [ACK] Seq=6260 Ack=24289 Win=86 Len=0 TSval=4183...

Conclusiones.

- El desarrollo del proyecto permitió obtener una metodología diferente en el aprendizaje de los estudiantes, ofreciendo una plataforma la cual cumple cada uno de sus actividades y requerimientos establecidos en su etapa de planificación mediante la investigación de las tecnologías involucradas.
- La infraestructura permite el desarrollo de prácticas en tecnologías SDN y la implementación de servicios NFV de manera manual, brindando al estudiante un control completo sobre los servicios, tecnologías e integración.
- El dimensionamiento de hardware para la infraestructura fué altamente influenciado por los servicios y funcionalidades requeridas en las instancias virtualizadas, con el objetivo de presentar al usuario una plataforma capaz de soportar las demandas de estas tecnologías en un modelo de estudio.
- Las guías realizadas, presentan al estudiante una base de conocimiento sobre el funcionamiento, configuración e implementación de las herramientas y servicios, facilitando su aprendizaje.

Recomendaciones

- Realizar un correcto dimensionamiento de los servicios y funcionalidades, antes de realizar la selección de hardware, realizar una sobre estimación de requerimientos y crecimiento de un 30% considerando valores indicados en el capítulo 4
- Utilizar siempre la última versión del software de Cloud Computing OpenStack por motivos de estabilidad y compatibilidad.
- El desarrollo de una implementación de este tipo realizar una investigación y pruebas previas al montaje de la misma, con el motivo de conocer el comportamiento y utilización de los recursos, del sistema.

Anexos

Instalación SO Base

Como primer paso desarrollamos la instalación del sistema base sobre el hardware especificado. Para este proyecto se ha elegido instalar Ubuntu Server el cual desarrollamos su instalación, indicado en la figura 163

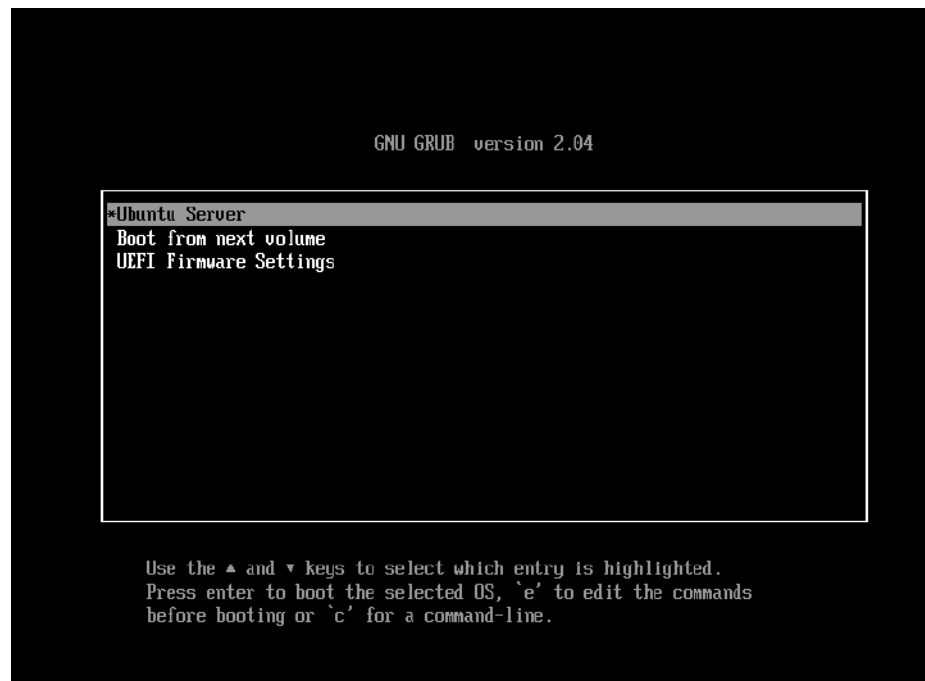


Figura 175 Boot Instalador Ubuntu Server.

La Instalación Completa del Sistema base está documentada en el Anexo 1

Al ingresar la imagen de instalación nos indicara el menú de opciones en la cual seleccionamos Ubuntu Server, indicada en la figura 164

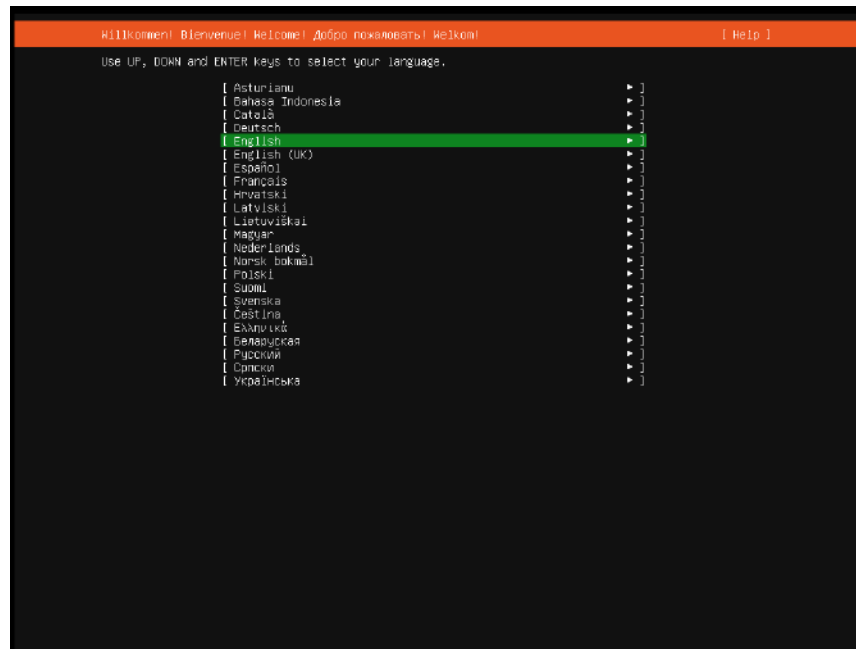


Figura 176 Selección de Idioma.

Una vez cargada la imagen nos mostrará el menú para seleccionar el idioma de instalación, indicada en la figura 165

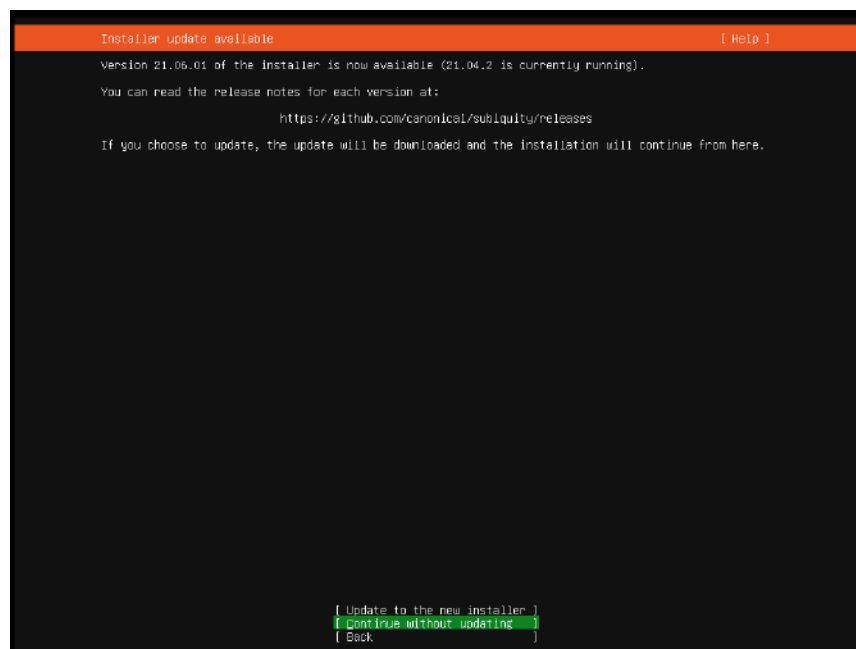


Figura 177 Selección de Instalador.

Si existiese una nueva versión del instalador de sistema seleccionamos la opción de instalación, lo que ejecutará la actualización de este instalador. Indicada en la figura 166

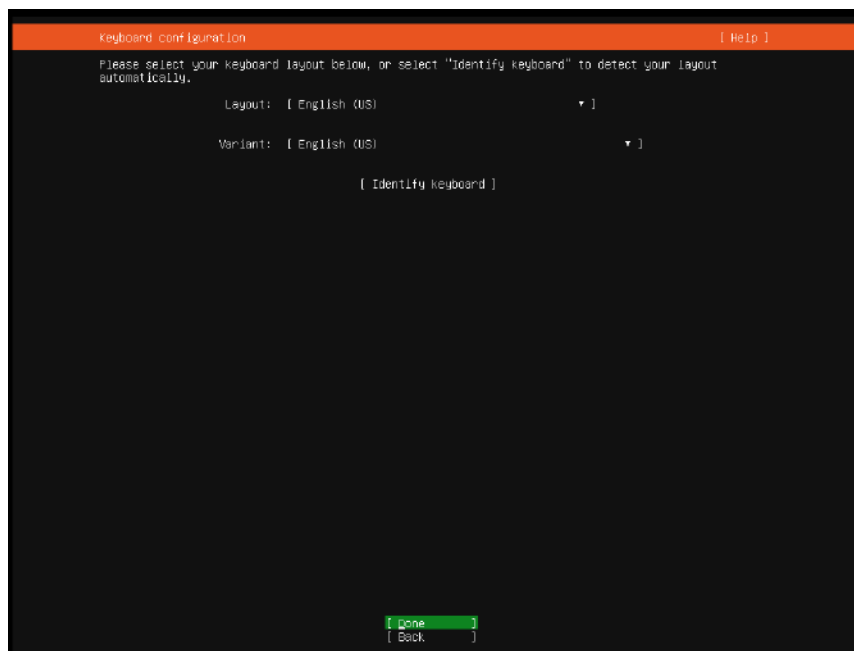


Figura 178 Selección Teclado.

Seleccionamos el teclado que deseemos utilizar en el sistema. Figura167

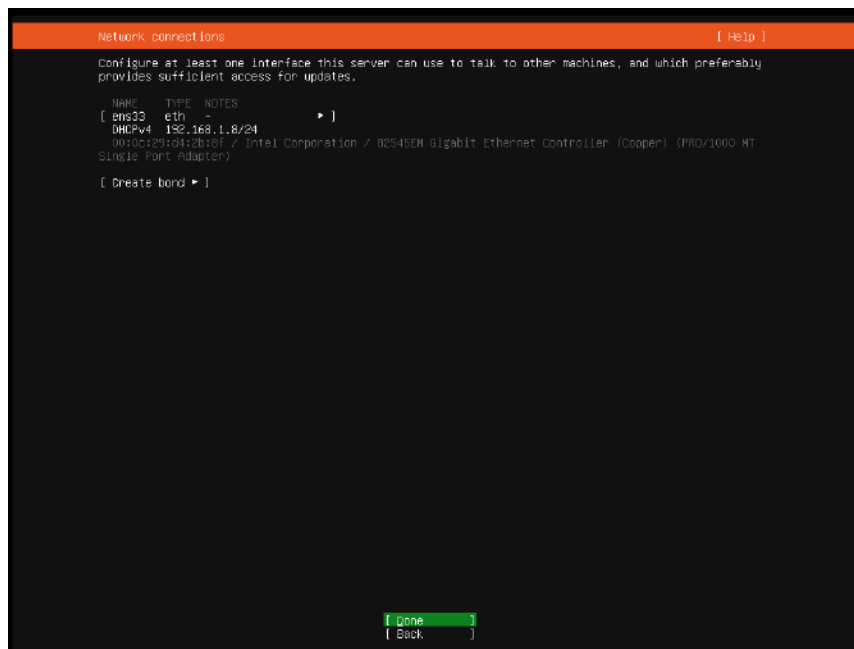


Figura 179 Configuración Red.

Realizará automáticamente la configuración de direccionamiento ip, pero para la aplicación es necesario realizar las configuraciones de manera manual ya que la ip debe ser fija para el funcionamiento de Openstack. Mostrado en la figura 41

Para esto establecemos la interfaz en Modo Manual con el direccionamiento ip 192.168.1.40 como se indica en la figura 168

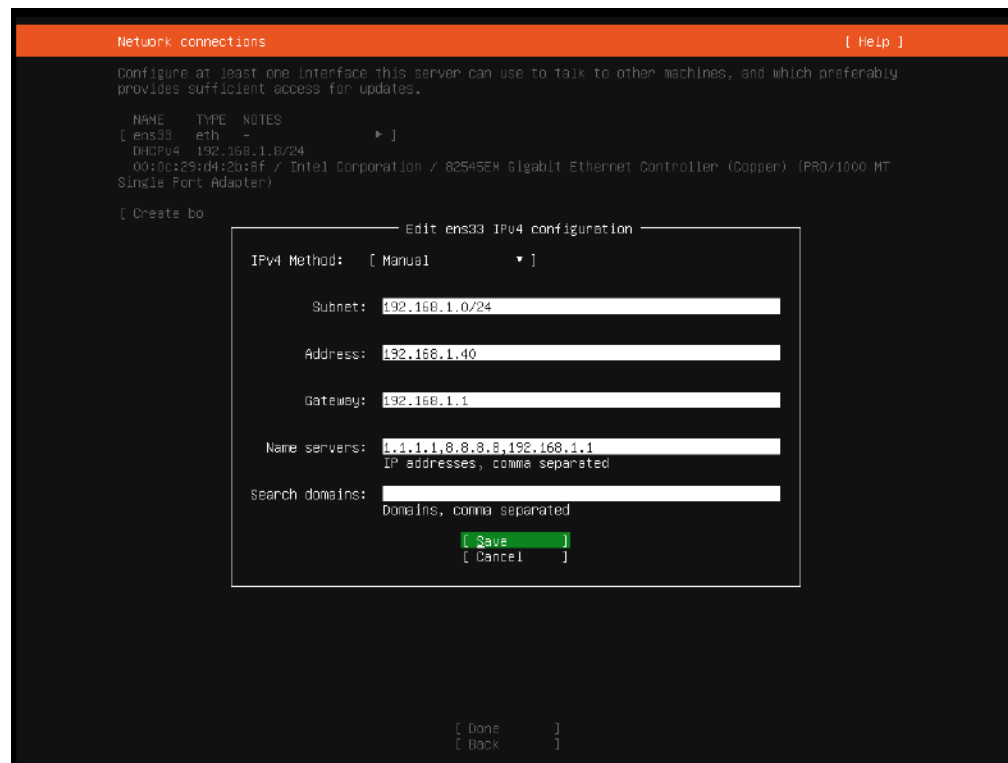


Figura 180 Configuración Manual Ip.

De esta manera aseguramos que la ip no cambie en el desarrollo del proyecto, permitiendo un funcionamiento de OpenStack.

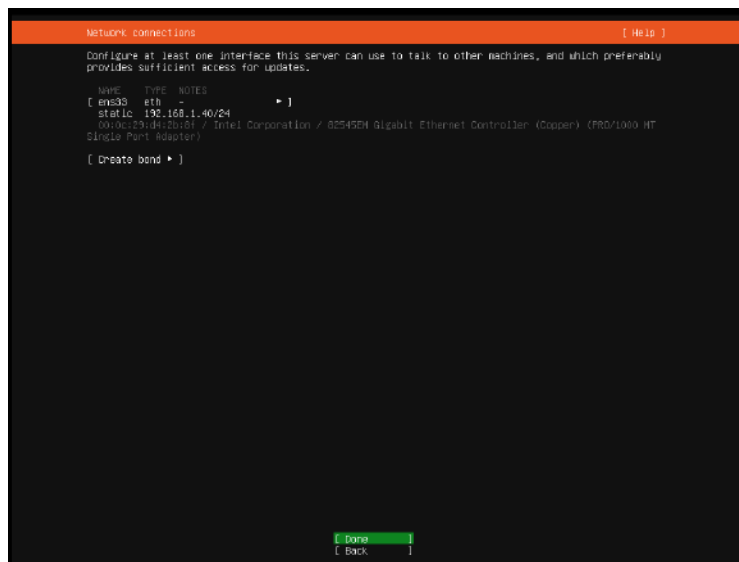


Figura 181 Aplicación Configuración.

Confirmamos la aplicación de las configuraciones de red manuales mostradas en la figura 169. Y continuamos con el proceso como se indica en la figura 170, al no existir configuraciones necesarias dejamos en blanco y continuamos con el procedimiento.

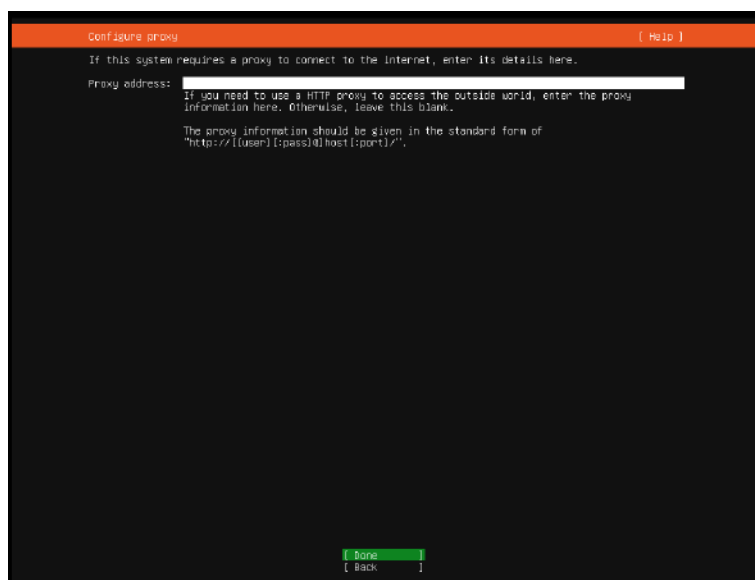


Figura 182 Configuración Proxi. Fuente: Autor

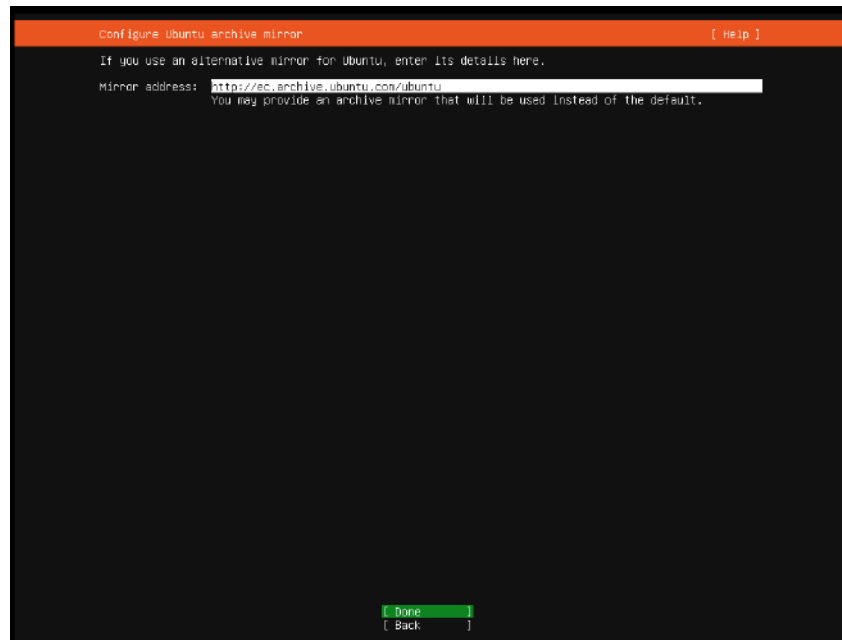


Figura 183 Repositorio Linux.

El instalador seleccionará el repositorio más cercano para su configuración como se muestra en la figura 171. A continuación se realiza la configuración de disco, donde utilizaremos la opción de todo el disco debido a que no requerimos particionamiento.

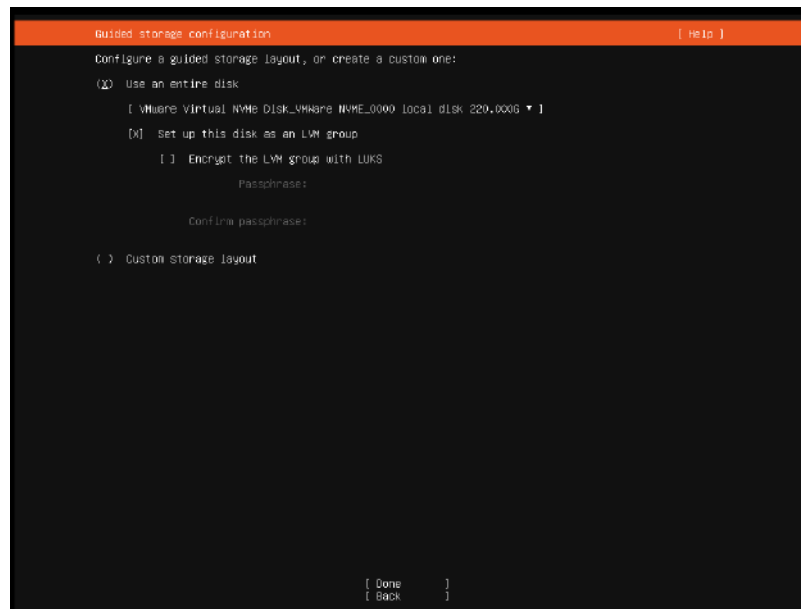


Figura 184 Configuración Disco.

De esta manera el sistema utilizará y configurará el disco completo indicado en la figura 172.

Seleccionado esto nos indicará un resumen mostrado en la Figura 173 de los cambios a realizar en el disco como son las particiones y las funciones de cada una de ellas.

```

Storage configuration [ Help ]

FILE SYSTEM SUMMARY
MOUNT POINT  SIZE  TYPE  DEVICE TYPE
[ /           109,248G  new ext4  new LVM logical volume ▶ ]
[ /boot       1,000G  new ext4  new partition of local disk ▶ ]
[ /boot/efi   512,000M  new fat32  new partition of local disk ▶ ]

AVAILABLE DEVICES
DEVICE              TYPE              SIZE
[ ubuntu-vg (new)   LVM volume group  218,496G ▶ ]
Free space          109,248G

[ Create software RAID (md) ▶ ]
[ Create volume group (VG) ▶ ]

USED DEVICES
DEVICE              TYPE              SIZE
[ ubuntu-vg (new)   LVM volume group  218,496G ▶ ]
ubuntu-lv           new, to be formatted as ext4, mounted at /  109,248G ▶ ]

[ VMware VIRTUAL NINE Disk_VMWare NINE_0000  local disk  200,000G ▶ ]
partition 1 new, primary ESP, to be formatted as fat32, mounted at /boot/efi  512,000M ▶ ]
partition 2 new, to be formatted as ext4, mounted at /boot  1,000G ▶ ]
partition 3 new, PV of LVM volume group ubuntu-vg  218,498G ▶ ]

[ Done ]
[ Reset ]
[ Back ]

```

Figura 185 Particiones Disco.

```

Storage configuration [ Help ]

FILE SYSTEM SUMMARY
MOUNT POINT  SIZE  TYPE  DEVICE TYPE
[ /           109,248G  new ext4  new LVM logical volume ▶ ]
[ /boot       1,000G  new ext4  new partition of local disk ▶ ]
[ /boot/efi   512,000M  new fat32  new partition of local disk ▶ ]

AVAILABLE DEVICES
DEVICE              TYPE              SIZE
[ ubuntu-vg (new)   LVM volume group  218,496G ▶ ]
Free space          109,248G

[ Create sp ]
[ Create vg ]

USED DEVICES
DEVICE              TYPE              SIZE
[ ubuntu-vg (new)   LVM volume group  218,496G ▶ ]
ubuntu-lv           new, to be formatted as ext4, mounted at /  109,248G ▶ ]

[ VMware V1 partition  200,000G ▶ ]
partition           512,000M ▶ ]
partition           1,000G ▶ ]
partition           218,498G ▶ ]

Confirm destructive action
Selecting Continue below will begin the installation process and
result in the loss of data on the disks selected to be formatted.

You will not be able to return to this or a previous screen once the
installation has started.

Are you sure you want to continue?

[ No ]
[ Continue ]

[ Done ]
[ Reset ]
[ Back ]

```

Figura 186 Confirmación Cambios Disco,

Como se muestra en la figura 174, la cual solicita la confirmación para que se realicen cambios en el disco ya que una vez confirmados no se puede recuperar la información de partición ni datos del disco.

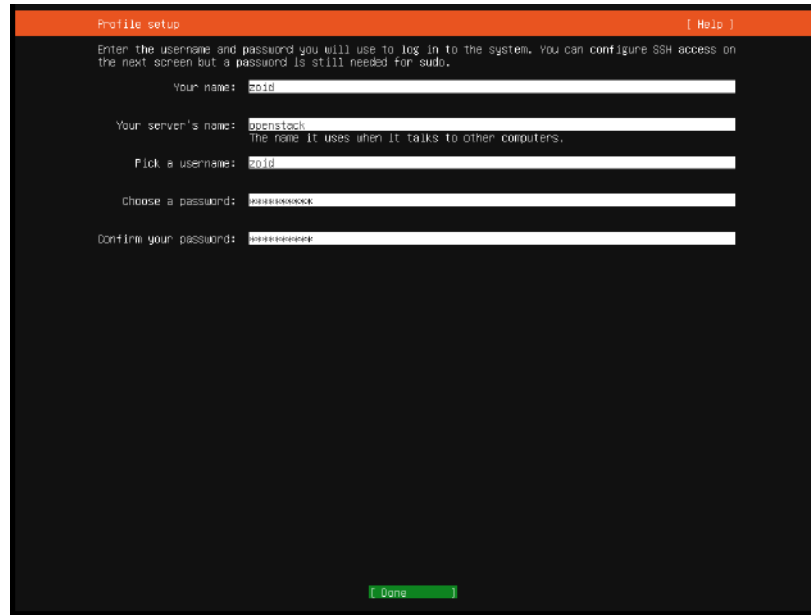


Figura 187 Configuración Usuario.

Establecemos Usuario y Contraseña de usuario para utilizar en el sistema, así como el nombre del host, como se muestra en la figura 175

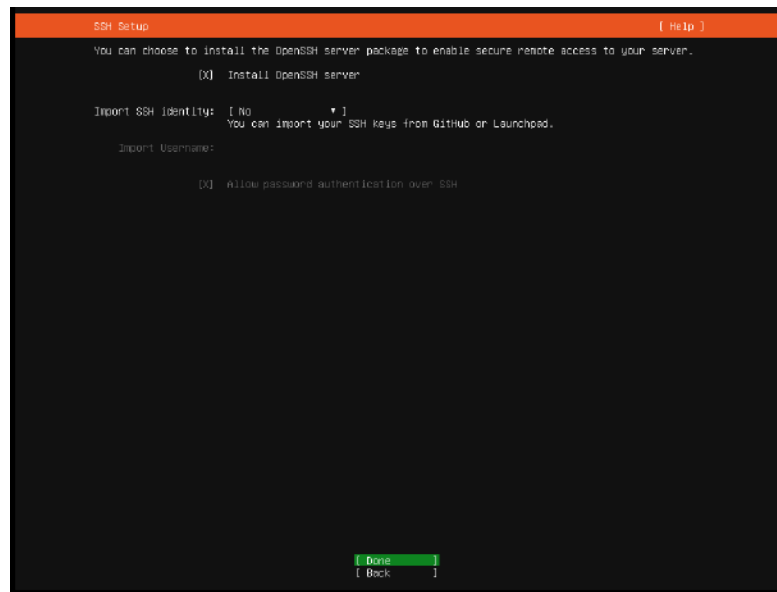


Figura 188 Servicio SSH.

Aceptamos la instalación del Servidor de SSH en el sistema para realizar tareas de administración remota en el sistema. Indicado en la figura 176

```

Featured Server Snaps [ Help ]
These are popular snaps in server environments. Select or deselect with SPACE, press ENTER to see
more details of the package, publisher and versions available.
[ ] microk8s canonical Lightweight Kubernetes for workstations and appliances
[ ] nextcloud nextcloud Nextcloud Server - A safe home for all your data
[ ] wekan xet7 Open-Source Kanban
[ ] kata-containers katacontainers Lightweight virtual machines that seamlessly plug into
[ ] docker canonical Docker container runtime
[ ] canonical-livepatch canonical Canonical Livepatch Client
[ ] rocketchat-server rocketchat Group chat server for 100s, installed in seconds.
[ ] mosquito mosquito Eclipse Mosquitto MQTT broker
[ ] etcd canonical Resilient key-value store by CoreOS
[ ] powershell microsoft-powershell PowerShell for every system
[ ] stress-ng cking-kernel-tools A tool to load, stress test and benchmark a computer
[ ] sdn2db sdn2db get things from one computer to another, safely
[ ] wemais snaccrafters Universal Command Line Interface for Amazon Web Serv
[ ] aws-cli aws Command-line interface for Google Cloud Platform prod
[ ] google-cloud-sdk google-cloud-sdk Python based SoftLayer API Tool
[ ] scli softlayer The official DigitalOcean command line interface
[ ] doctl digitalocean Package runtime for conjure-up spells
[ ] conjure-up canonical server software with the aim of being fully compliant
[ ] minidino-escond cond Postgres is a powerful, open source object-relatio
[ ] postgresql cond Client for Heroku
[ ] heroku heroku High availability VRRP/BFD and load-balancing for Lin
[ ] keepalived keepalived-project The Prometheus monitoring system and time series data
[ ] prometheus canonical
[ ] juju canonical A model-driven operator lifecycle manager
[ Done ]
[ Back ]

```

Figura 189 Características Extra.

En el menú mostrado en la figura 177 se presentan las preconfiguraciones de características que podemos instalar en el servidor, al ninguna ser de utilidad, no seleccionamos ninguna y proseguimos con la instalación.

```

Install complete! [ Help ]

configuring mount: mount-2
configuring mount: mount-1
configuring mount: mount-0
writing install sources to disk
  running 'curtin extract'
    curtin command extract
      acquiring and extracting image from cp:///media/filesystem
configuring installed system
  running '/snap/bin/subiquity.subiquity-configure-apt /snap/subiquity/2501/usr/bin/python3
true'
  curtin command apt-config
  curtin command in-target
  running 'curtin curthooks'
  curtin command curthooks
    configuring apt configuring apt
    installing missing packages
    Installing packages on target system: ['efibootmgr', 'grub-efi-amd64',
'grub-efi-amd64-signed', 'shim-signed']
    configuring iscsi service
    configuring raid (mdadm) service
    installing kernel
    setting up swap
    apply networking config
    writing etc/fstab
    configuring multipath
    updating packages on target system
    configuring pollinate user-agent on target
    updating initramfs configuration
    configuring target system bootloader
    installing grub to target devices
  finalizing installation
  running 'curtin hook'
  curtin command hook
  executing late commands
final system configuration
  configuring cloud-init
  installing openssh-server
  restoring apt configuration
  downloading and installing security updates |

[ View full log ]
[ Cancel update and reboot ]

```

Figura 190 Instalación Servidor.

Una vez terminado ese proceso automáticamente inicia la instalación y configuración del servidor, aguardamos mientras completa la instalación ya que puede tomar varios minutos. Como se muestra en la figura 178

Y una vez que ha terminado es posible reiniciar el equipo. Mostrado en la figura 179

```

Install complete! [ Help ]

Configuring mount: mount-1
Configuring mount: mount-0
writing install sources to disk
running 'curtin extract'
curtin command extract
acquiring and extracting image from cpi:///media/filesystem
configuring installed system
running '/snap/bin/subiquity.subiquity-configure-apt /snap/subiquity/2501/usr/bin/python3
true'
curtin command apt-config
curtin command in-target
running 'curtin curthooks'
curtin command curthooks
configuring apt configuring apt
installing missing packages
Installing packages on target system: ['efibootmgr', 'grub-efi-amd64',
'grub-efi-amd64-signed', 'shim-signed']
configuring lscc service
configuring raid (mdadm) service
installing kernel
setting up sudo
apply networking config
writing etc/fstab
configuring multipath
updating packages on target system
configuring collinate user-agent on target
updating intramfs configuration
configuring target system bootloader
installing grub to target devices
finalizing installation
running 'curtin hook'
curtin command hook
executing late commands
final system configuration
configuring cloud-init
installing openssh-server
restoring apt configuration
downloading and installing security updates
subiquity/late/run

[ View full log ]
[ Reboot Now ]

```

Figura 191 Sistema Instalado.

En la figura 180 se muestra la interfaz cli correspondiente al sistema instalado en el disco.

```

Ubuntu 21.04 openstack tty1

openstack login: zoid
Password:
Welcome to Ubuntu 21.04 (GNU/Linux 5.11.0-22-generic x86_64)

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

System information as of Wed Jun 23 15:55:51 UTC 2021

System load: 0.19           Memory usage: 2%          Processes:   662
Usage of /:  5.9% of 107.04GB  Swap usage:  0%         Users logged in: 0

=> There were exceptions while processing one or more plugins. See
/var/log/landscape/sysinfo.log for more information.

35 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

```

Figura 192 Boot Terminal.

Una vez reiniciado se obtendrá la terminal del servidor ya funcionando correctamente. Al cual realizamos una actualización de todos sus componentes mediante la utilización del comando:

Sudo apt update && sudo apt upgrade -y

El cual permite actualizar la lista de repositorio y actualización general del sistema. Seguido a esto realizamos la instalación de la interfaz gráfica KDE, para mayor facilidad de administración del sistema con el siguiente comando

`sudo apt-get install kubuntu-desktop --no-install-recommends`

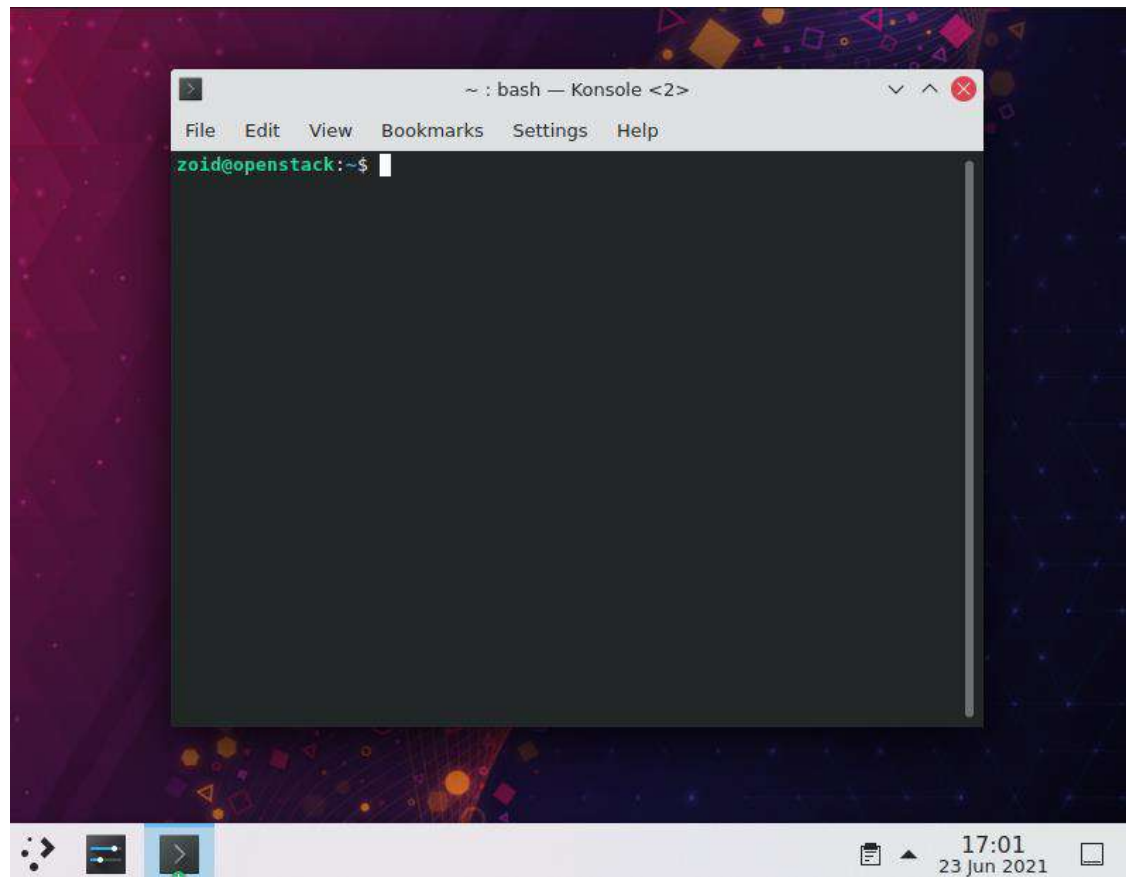


Figura 193 Interfaz Gráfica.

Una vez terminada la instalación es posible acceder a la interfaz gráfica. Tal y como se muestra en la figura 181

Instalación Sistema Imagen Host

Una vez generado el perfil y encendiendo la máquina KVM, se obtiene una ventana como la indicada en la figura 182

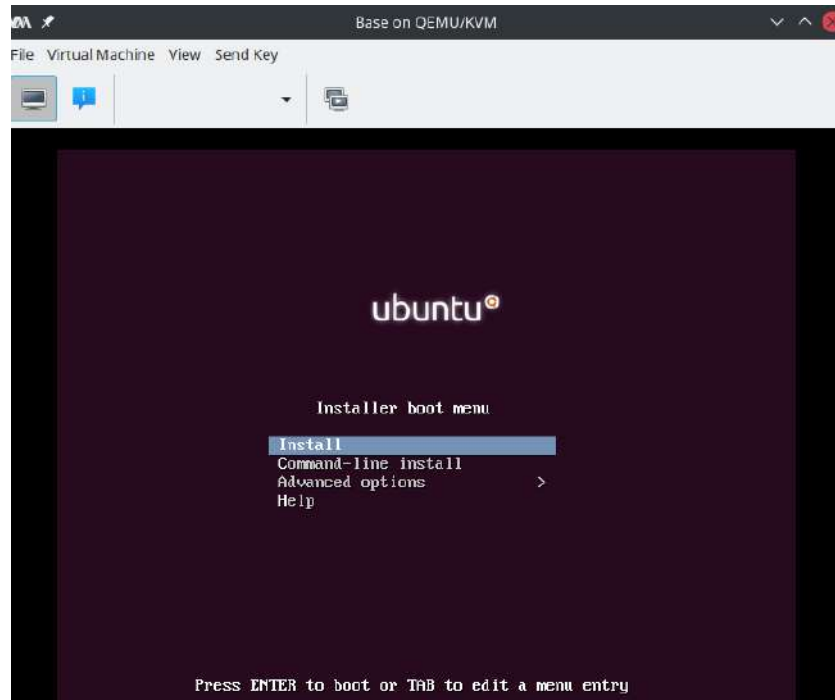


Figura 194 KVM Fuente: Autor

La continuación de la instalación se encuentra detallada en el anexo 2

Al obtener esta imagen realizamos la instalación básica del sistema Ubuntu.

Con la selección del idioma del sistema. Figura 183

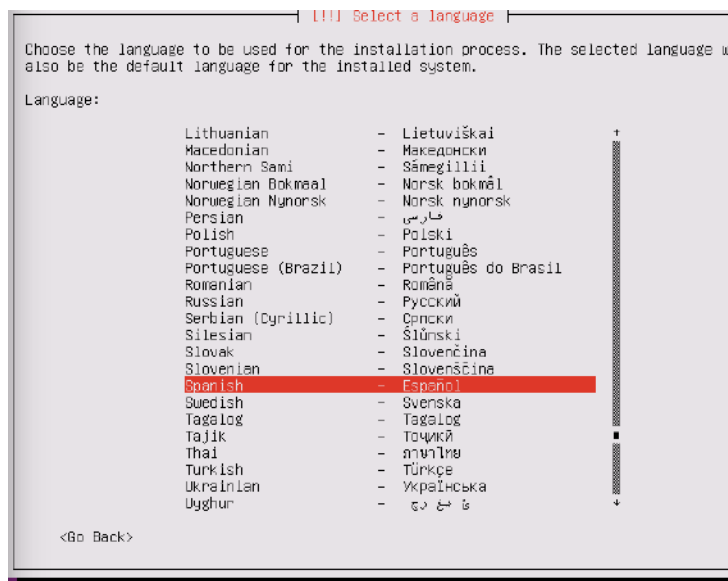


Figura 195 Selección Idioma SO Base

De la misma manera seleccionamos la ubicación para establecer el sistema horario del sistema como se indica en la figura 184.

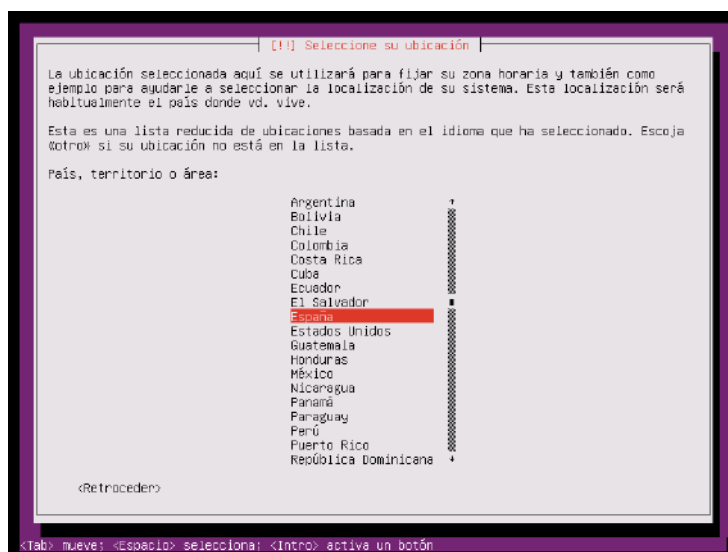


Figura 196 Ubicación

Una vez seleccionada esto establecerá la conexión de red automáticamente, no es necesario utilizar ip fija ya que se requiere que toma ip automáticamente.

Establecemos el Hostname de la máquina como se muestra la figura 185

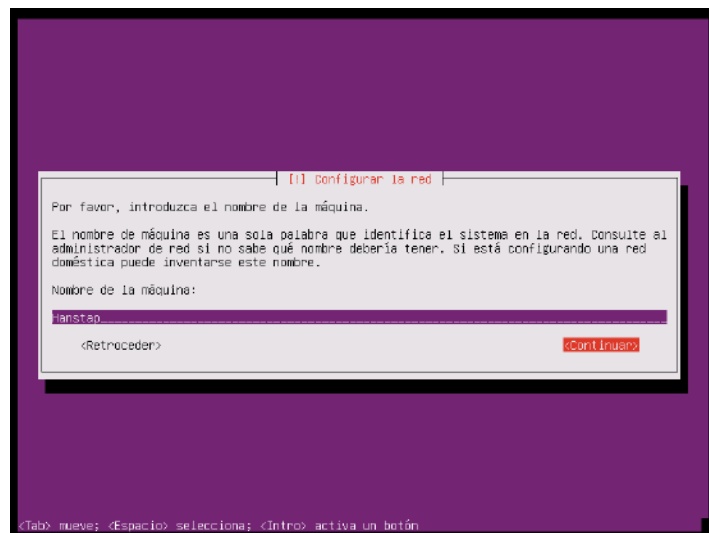


Figura 197 Hostname.

Una vez establecidas estas configuraciones el instalador descargará los ficheros de instalación, así como un asistente de instalación, con el que será posible establecer las configuraciones del sistema operativo.

Designación de Usuarios

En la figura 186 se observa el dialogo donde se requiere la configuración del nombre de usuario de sistema.

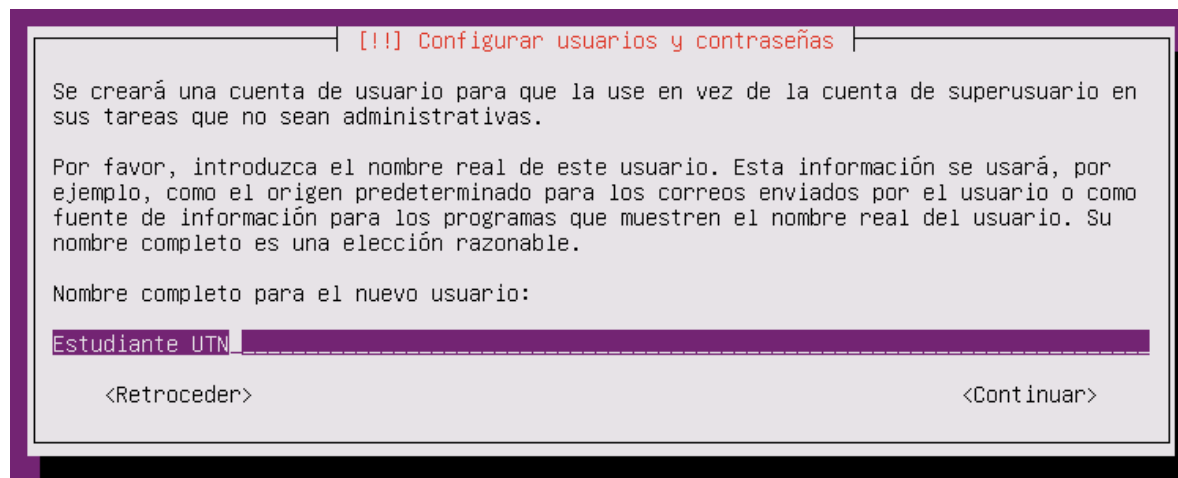
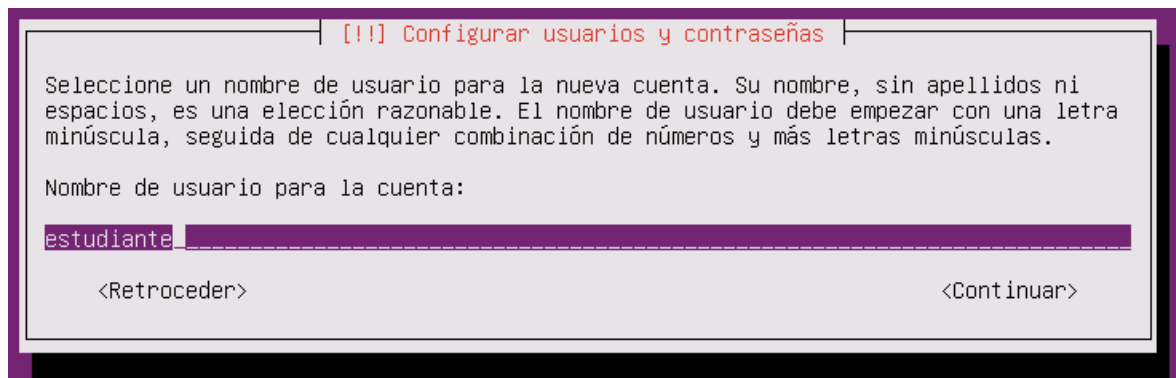


Figura 198 Nombre Cuenta. Fuente: Autor,

De esta manera tomara como usuario estudiante como se aprecia en la figura 187



[!!] Configurar usuarios y contraseñas

Seleccione un nombre de usuario para la nueva cuenta. Su nombre, sin apellidos ni espacios, es una elección razonable. El nombre de usuario debe empezar con una letra minúscula, seguida de cualquier combinación de números y más letras minúsculas.

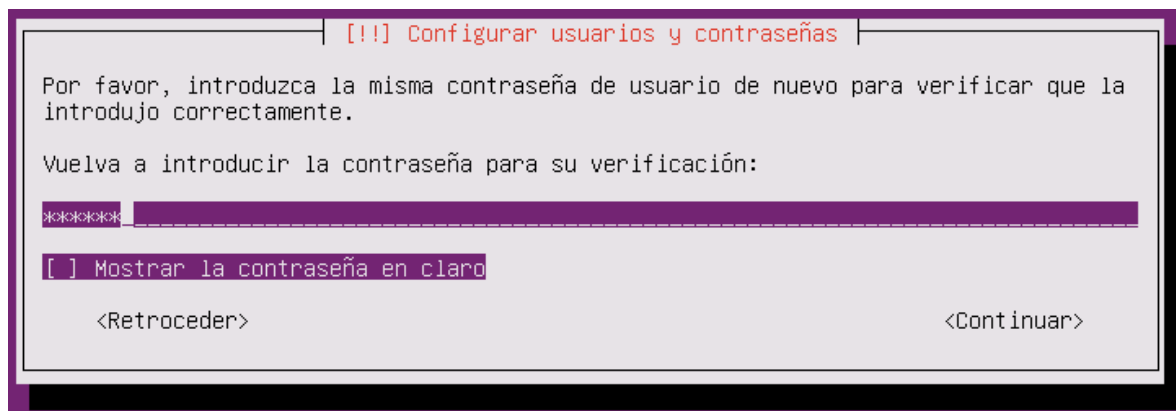
Nombre de usuario para la cuenta:

estudiante

<Retroceder> <Continuar>

Figura 199 Nombre usuario.

Siendo el usuario con el que se debe ingresar hacia el sistema, acto seguido establecemos la contraseña de este usuario para su administración, como indicamos en la figura 188



[!!] Configurar usuarios y contraseñas

Por favor, introduzca la misma contraseña de usuario de nuevo para verificar que la introdujo correctamente.

Vuelva a introducir la contraseña para su verificación:

Mostrar la contraseña en claro

<Retroceder> <Continuar>

Figura 200 Contraseña usuario.

Esta contraseña servirá para acciones administrativas del sistema, puesto que el usuario tiene derechos administrativos en el sistema necesarios para realizar configuraciones en las implementaciones SDN y NFV.

Configuración de disco.

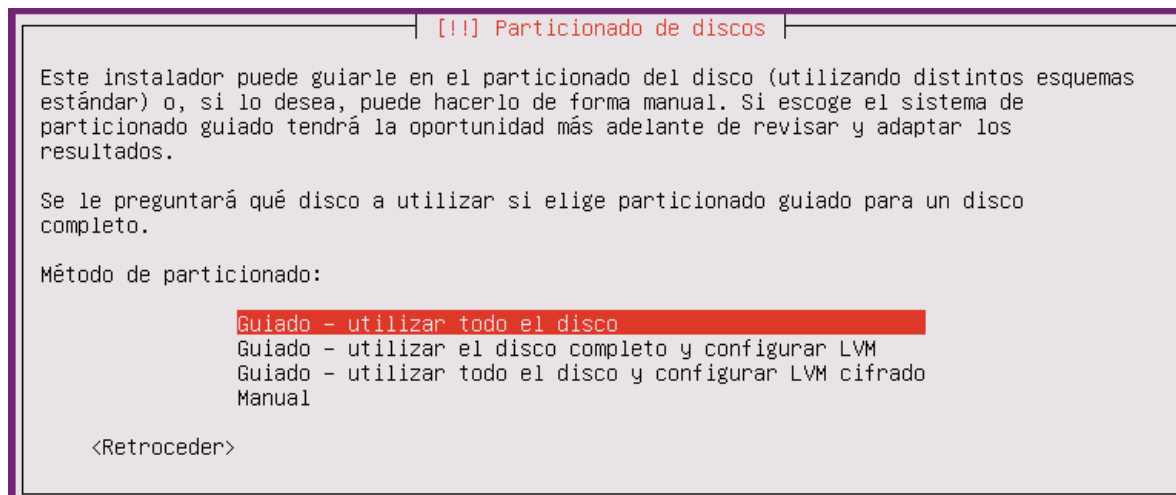


Figura 201 Configuración Disco.

Como se muestra en la figura 189 estableceremos la utilización de todo el disco ya que no realizaremos configuraciones de particiones específicas o dedicadas.

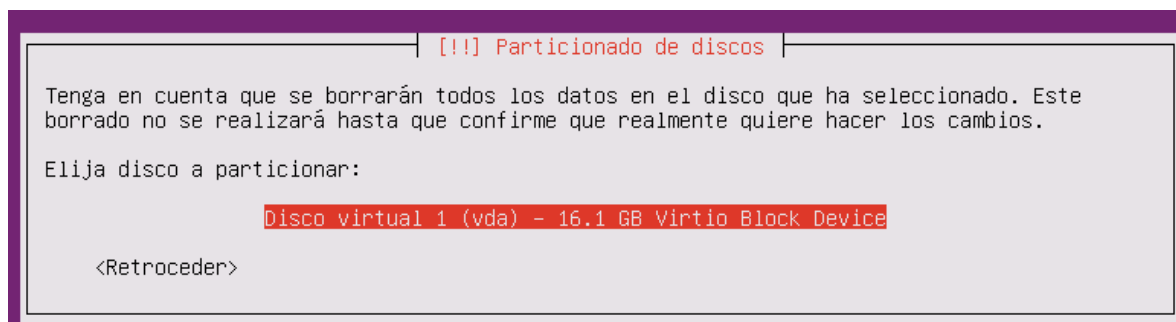


Figura 202 Disco Virtual.

Seleccionamos el disco a utilizar en este caso es el único disponible mostrado en la figura 190. Para la instalación donde realizará la asignación automática de las particiones requeridas por el sistema. Tal y como se muestra en la figura 191

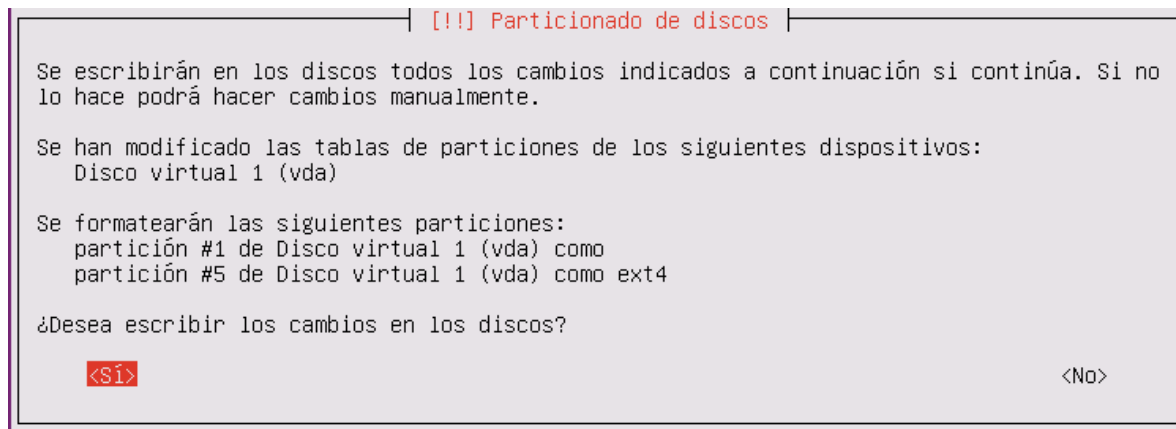


Figura 203 Particiones.

De esta manera se establecen los cambios al disco y aceptamos los cambios y configuraciones a realizar. A continuación, como se muestra en la figura 192 indicamos el manejo de actualizaciones de sistema.

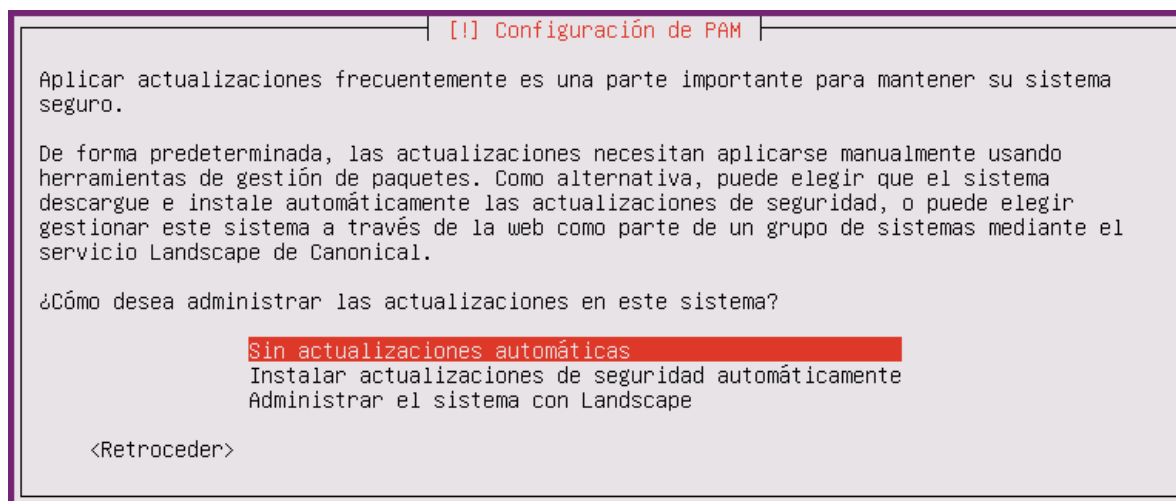


Figura 204 Actualizaciones.

En esta configuración utilizaremos actualizaciones automáticas.

Gestión de características.

En esta sección establecerá las características deseadas sistema, de manera que no contenga opciones o software no requerido para el propósito del proyecto. Tal y como se muestran en la figura 193



Figura 205 Características.

Estableciendo de esta manera la imagen que se está generando, como una instancia compatible con OpenStack. Y continuamos con la instalación como se indica en la figura 194. La cual dependiendo del número de características seleccionadas tomará más o menos tiempo.

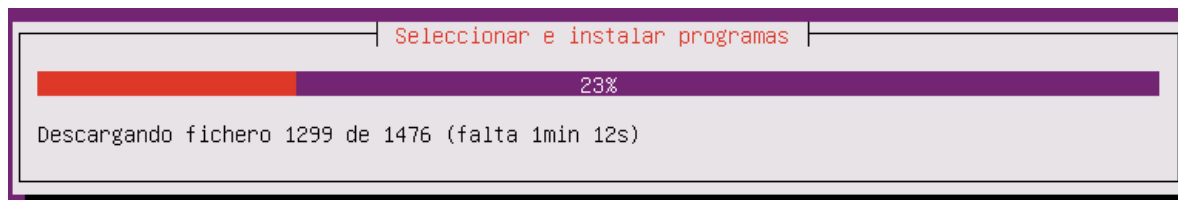


Figura 206 Progreso instalación.

Al punto final de la instalación del paquete grub le damos la opción yes, para poder obtener un cargador de sistema en el disco. Tal y como se muestra en la figura 195

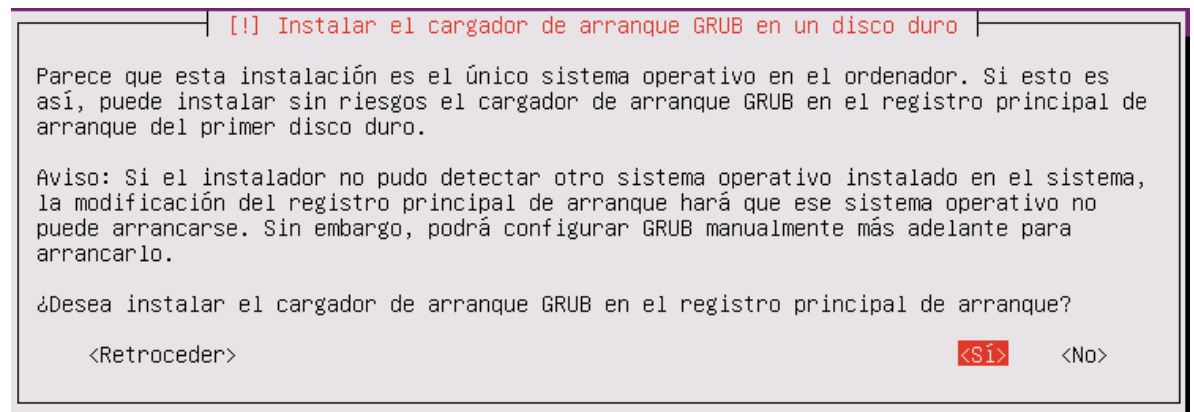


Figura 207 Instalación Grub.

Una vez completada la instalación nos aparecerá la pantalla indicando que la instalación ha sido completada como se indica en la Figura 196.

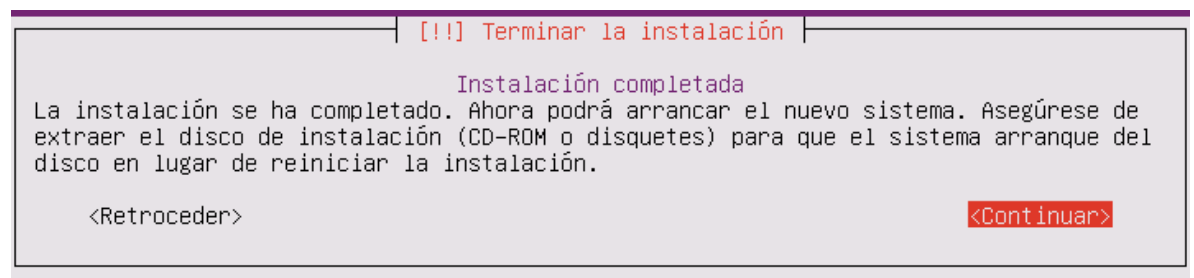


Figura 208 Instalación Completa.

Boot inicial.

Al reiniciar el equipo se obtiene la pantalla de Login para el sistema. Mostrada en la figura 197



Figura 209 Pantalla Login.

Una vez ingresados los parámetros se ingresa al escritorio del sistema mostrado en la figura 198, donde se configurará los elementos para poder realizar las configuraciones de SDN y NFV así como los paquetes requeridos.

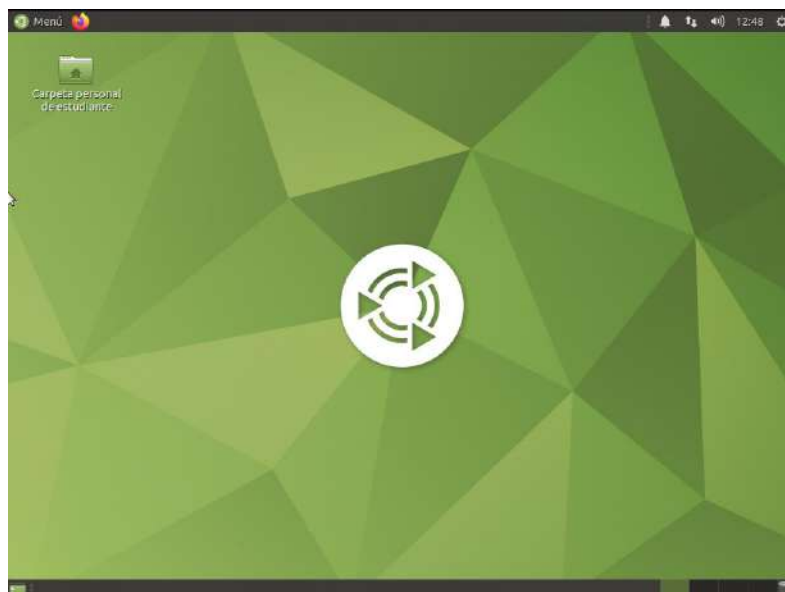


Figura 210 Escritorio Sistema.

Bibliografía

- Clavel, T. (8 de enero de 2018). *GIGAMON*. Obtenido de gigamon:
<https://blog.gigamon.com/2018/01/08/what-is-netflow/>
- Containernet. (22 de 07 de 2021). <https://github.com/containernet/containernet>.
 Obtenido de <https://github.com/containernet/containernet>:
<https://github.com/containernet/containernet>
- cpubenchmark. (10 de 06 de 2021). *cpubenchmark*. Obtenido de cpubenchmark:
<https://www.cpubenchmark.net>
- Doherty, J. (2016). *SDN and NFV Simplified*. Kendallville: PEARSON.
- Etsi. (2018). Network Functions Virtualisation (NFV). *Network Functions Virtualisation – Update White Paper*, 16.
- García, A. S. (2016). Next Generation Networks - The technologies and Enablers. Berlin: DAI-Labor.
- Göransson, P. (2017). *Software Defined Networks*. Cambridge: Morgan Kaufmann.
- IEEE. (20 de Oct de 1998). *Recommended Practice for Software Requirements Specifications*. Obtenido de Recommended Practice for Software Requirements Specifications: <https://ieeexplore.ieee.org/document/720574>
- IETF. (2020). *Benchmarking Methodology for Network Interconnect Devices*. Cambridge : IEEE.
- Intel. (2020). *Open vSwitch* Enables SDN and NFV Transformation*. <https://networkbuilders.intel.com/docs/open-vswitch-enables-sdn-and-nfv-transformation-paper.pdf>: Intel.
- IONOS. (24 de Junio de 2019). *ionos.es*. Obtenido de ionos.es:
<https://www.ionos.es/digitalguide/servidores/know-how/que-es-iaas/>
- Michael S. Bonfim. (2018). *Integrated NFV/SDN Architectures: A Systematic Literature Review* (pág. 52). Pernambuco: Universidad Federal de Pernambuco.

Microsoft. (17 de Enero de 2017). *Virtual vs Physical Memory?* Obtenido de Virtual vs Physical Memory?: <https://answers.microsoft.com/en-us/windows/forum/all/virtual-vs-physical-memory/23059844-5368-48e8-9683-7365831cb44e>

Microsoft. (22 de 07 de 2021). <https://azure.microsoft.com/en-us/overview/what-is-iaas/>. Obtenido de <https://azure.microsoft.com/en-us/overview/what-is-iaas/>: <https://azure.microsoft.com/en-us/overview/what-is-iaas/>

Mininet. (22 de 07 de 2021). <http://mininet.org/>. Obtenido de <http://mininet.org/>: <http://mininet.org/>

NIST. (22 de 07 de 2021). <https://csrc.nist.gov/>. Obtenido de <https://csrc.nist.gov/>: <https://csrc.nist.gov/publications/detail/sp/800-145/final>

opennetworking.org. (20 de 07 de 2021). <https://opennetworking.org/onos/>. Obtenido de Open Network Operating System: <https://opennetworking.org/onos/>

Openstack. (22 de 07 de 2021). <https://docs.openstack.org/releasenotes/openstack-manuals/index.html>. Obtenido de <https://docs.openstack.org/releasenotes/openstack-manuals/index.html>: <https://docs.openstack.org/releasenotes/openstack-manuals/index.html>

Openvswitch.org. (20 de 07 de 2021). *Open vSwitch*. Obtenido de Open vSwitch: <https://www.openvswitch.org/>

Pepelnjak, I. (18 de Noviembre de 2020). *ipspace.com*. Obtenido de [ipspace.com](https://blog.ipspace.net/2013/08/management-control-and-data-planes-in.html): <https://blog.ipspace.net/2013/08/management-control-and-data-planes-in.html>

Portnoy, M. (2016). *Virtualization ESSENTIALS*. Indianapolis: John Wiley & Sons.

Rogier, B. (13 de Julio de 2016). *Measuring network performance: links between latency, throughput and packet loss*. Obtenido de Measuring network performance: links between latency, throughput and packet loss: <https://accedian.com/blog/measuring-network-performance-latency-throughput-packet-loss/>

Schoenfelder, N. (28 de 10 de 2021). *pingplotter*. Obtenido de pingplotter:
<https://www.pingplotter.com/wisdom/article/is-my-connection-good>

Stallings, W. (2016). *Foundations of Modern Networking*. Indianapolis:
PEARSON.

Ubuntu.com. (2021). *Requerimientos Hardware*.

VMware. (2017). Architecting a vCloud NFV. *Architecting a vCloud NFV*, 49.

VMware. (1 de Marzo de 2022). *Performance Best Practices for VMware vSphere*. Obtenido de Performance Best Practices for VMware vSphere:
<https://download3.vmware.com/vcat/vmw-vcloud-architecture-toolkit-spv1-webworks/index.html#page/Core%20Platform/Architecting%20a%20vSphere%20Compute%20Platform/Architecting%20a%20vSphere%20Compute%20Platform.1.019.html>

Zhang, Y. (2018). *Network Functions Virtualization*. Chennai: Wiley, IEEE
PRESS.