UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

IMPLEMENTACIÓN DE UNA RED DE ÁREA AMPLIA BAJO EL PARADIGMA DE REDES VIRTUALES

TESIS

Que para obtener el título de

Ingeniero en Telecomunicaciones

P R E S E N T A

Irving Yohanan Peña Núñez

DIRECTOR DE TESIS

Dr. Luis Francisco García Jiménez

Ciudad Universitaria, Cd. Mx., 2024

Agradecimientos

A mis padres, Guadalupe y Jesús, por su amor incondicional, paciencia y sabiduría, los cuales han sido la fuerza impulsora detrás de mi para convertirme en la persona que soy. Gracias a su inquebrantable dedicación y sacrificio, he podido perseguir mis metas. Por ello, este logro también les pertenece a ustedes.

A mi hermano Felipe, que ha sido mi ejemplo a seguir en mi camino académico y profesional. Le agradezco por estar siempre al pendiente de mí, apoyarme en todo momento y por ser un hermano excepcional.

A mi hermano Axel, por ser mi compañero de aventuras y por brindarme una perspectiva diferente en cada momento de mi vida.

A todos mis compañeros y amigos, por los momentos felices que hemos compartido juntos.

A todos los profesores que me acompañaron durante este camino académico, por sus enseñanzas y consejos.

Al Dr. Francisco García, por su tiempo y dedicación en la elaboración de esta Tesis.

Igualmente, agradezco el valioso apoyo otorgado por el proyecto PAPIIT-IN109624.

Irving Yohanan Peña Núñez

Índice general

Índice de figuras

Índice de Consolas

Índice de tablas

Acrónimos

1. IP: *Internet Protocol*, protocolo de Internet.
2. VPN: *Virtual Private Network*, red privada virtual.
3. IoT: *Internet of Things*, Internet de las cosas.
4. VPS: *Virtual Private Server*, servidor privado virtual.
5. API: *Application Programming Interface*, interfaz de programación de aplicaciones.
6. CPU: *Central Processing Unit*, unidad central de procesamiento.
7. OVS: *Open Virtual Switch*, conmutador virtual abierto.
8. OS: *Operating System*, sistema operativo.
9. NaaS: *Network as a Service*, red como servicio.
10. IaaS: *Infrastructure as a service*, infraestructura como servicio.
11. PaaS: *Platform as a service*, plataforma como servicio.
12. SaaS: *Software as a service*, software como servicio.
13. VLAN: *Virtual Local Area Network*, red de área local virtual.
14. LVM: *Logical Volume Manager*, gestor de volúmen lógico.
15. MB: Megabyte.
16. VM: *Virtual Machine*, máquina virtual.
17. DHCP: *Dynamic Host Configuration Protocol*, protocolo de configuración dinámica de Host.
18. MAC: *Media Access Control*, control de acceso al medio.
19. IPv4: *Internet Protocol Version 4*, protocolo de Internet versión 4.
20. TCP: *Transport Control Protocol*, protocolo de control de transporte.
21. IPv6: *Internet Protocol Version 6*, protocolo de Internet versión 6.
22. UDP: *Universal Datagram Protocol*, protocolo de datagrama universal.
23. RAM: *Random Access Memory*, memoria de acceso aleatorio.
24. SDN: *Software Defined Network*, red definida por software.
25. SQL: *Structured query language*, lenguaje de consulta estructurada.
26. SSH: *Secure Shell*.
27. URL: *Uniform Resource Locatorl*.

Resumen

La forma en que nos comunicamos, relacionamos y vivimos se ha ido transformando a medida que aumenta el consumo de servicios en Internet. Por ejemplo, estos servicios han permitido que cualquier persona pueda comprar en línea, responder un correo electrónico y acceder a enormes cantidades de información desde cualquier lugar del mundo. En gran parte, estos beneficios se deben al uso del cómputo en la nube.

Dentro del cómputo en la nube existe un modelo que se denomina infraestructura como servicio (IaaS), el cual proporciona a los usuarios acceso a recursos de red, procesamiento y almacenamiento de datos desde cualquier parte del mundo. Estos recursos son creados bajo el paradigma de virtualización de funciones de red (NVF). En la actualidad, existen varios proveedores que ofrecen el modelo IaaS, por ejemplo AWS, VMWARE y OPENMETAL, sin embargo, estas organizaciones operan bajo licencias privadas y por tanto hay que pagar para acceder a sus recursos. No obstante, existe una alternativa llamada OpenStack, una plataforma de código abierto que gestiona recursos de red, procesamiento y almacenamiento a través de un panel de control bajo la licencia GNU.

Por otro lado, algunas organizaciones empresariales o instituciones de educación superior tienen sucursales físicamente distantes que utilizan la infraestructura MPLS para mantenerse conectados y unificarse entre sí. Esto implica que estas organizaciones deben contratar a operadores que les brinden este servicio, lo que se traduce en un monto económico extra. Además, conectar varios sitios lejanos usando MPLS no suele ser una tarea fácil. Actualmente, es posible enlazar diferentes sedes de una misma organización mediante el paradigma SD-WAN. Esta tecnología basa sus principios en las redes definidas por software (SDN). Con base en esta tecnología, se puede gestionar y optimizar el rendimiento de las redes virtuales de área extensa (V-WAN) mediante la implementación de un controlador que encamina y distribuye la información de manera inteligente.

SD-WAN ofrece una serie de ventajas sobre MPLS. Entre ellas se incluye una mejor disponibilidad gracias al encaminamiento dinámico e inteligente, una mayor visibilidad y control sobre la red, mayor resiliencia y redundancia al permitir el uso de múltiples rutas de conexión, así como una mayor flexibilidad en la configuración con los sistemas en la nube. Además, SD-WAN puede reducir los costos al utilizar conexiones de Internet, como *VPN*, en lugar de las líneas dedicadas que utiliza MPLS. Sin embargo, es importante tener en cuenta que hasta nuestro conocimiento, las soluciones de SD-WAN disponibles en el mercado son privadas, lo que implica que no hay alternativas de código abierto disponibles.

Este proyecto de tesis aborda la primera etapa (de dos fases) de la implementación de una SD-WAN. Esta primera etapa consta de la creación de una red virtual de área amplia (V-WAN). Para ello, se crean dos nubes privadas (IasS) orquestadas con OpenStack que se encuentran en dos ubicaciones distantes. También, se añade un tercer elemento (usuario invitado) que se conecta a la nube privada desde un punto externo. Los tres elementos se conectan mediante una *VPN* que utiliza el protocolo WireGuard. Finalmente, se realizan pruebas de conectividad entre los tres elementos para evaluar el funcionamiento de la V-WAN. Una vez concluida esta etapa, la segunda fase (en un trabajo futuro), se abordará el desarrollo de un controlador inteligente que tome decisiones en tiempo real, además de la implementación de medidas de seguridad.

Capítulo 1

Introducción

La comunicación, las relaciones interpersonales y nuestra forma de vida han experimentado una notable transformación a medida que el consumo de servicios en Internet han ido en aumento. Esta transformación ha sido posible gracias al rápido desarrollo tecnológico, el cual ha permitido que cualquier persona tenga la capacidad de realizar compras en línea, responder correos electrónicos desde la comodidad de su hogar y acceder a una vasta cantidad de información desde cualquier lugar del mundo. Estos beneficios, en gran medida, se atribuyen al uso del cómputo en la nube.

El cómputo en la nube o *cloud computing* ofrece tres modelos principales: IaaS (infraestructura como servicio), PaaS (plataforma como servicio) y SaaS (software como servicio). El primero de ellos proporciona a los usuarios acceso a una variedad de recursos, incluyendo sistemas de redes, capacidad de procesamiento y almacenamiento de datos. En lo que respecta a los recursos de red, el modelo IaaS opera mediante la virtualización de dispositivos como *routers*, *switches*, y balanceadores de carga.

IaaS permite a muchas empresas y a instituciones de educación superior (IES) utilizar aplicaciones, sistemas de procesamiento y almacenamiento basado en páginas web sin tener que comprar, administrar y brindar soporte a la infraestructura subyacente. Por ejemplo, Amazon Web Services (AWS), Microsoft Azure, VMWARE y OPENMETAL ofrecen el modelo IaaS. Sin embargo, el uso de sus recursos conlleva un costo elevado, especialmente si se requiere un alto uso de procesamiento. Esto puede convertirse en un problema para las IES que tienen recursos limitados y no pueden pagar este tipo de servicios. Como alternativa se encuentra OpenStack, una plataforma IaaS de código abierto que gestiona recursos de red, procesamiento y almacenamiento a través de un panel de control bajo la licencia GNU.

Las organizaciones empresariales, así como las IES que tienen sucursales físicamente distantes suelen utilizar la infraestructura de red MPLS para mantenerse conectados entre sí. Esto les permite usar sus recursos internos como si todas las sucursales estuvieran localizadas en un mismo lugar. La conmutación de etiquetas multi-protocolo, o MPLS, es una tecnología de red que encamina el tráfico utilizando la ruta más corta basada en etiquetas, en lugar de direcciones *IP* [46]. El uso de MPLS conlleva ciertos inconvenientes. Por ejemplo, requiere *hardware* y *software* especializado para funcionar, las conexiones dedicadas son más caras en comparación con las conexiones compartidas. MPLS no proporciona cifrado de forma predeterminada, el proveedor de servicios de Internet controla las divisiones de encaminamiento y el tráfico. Todo esto conlleva a que hoy en día utilizar MPLS, en algunas condiciones, no es la mejor opción. Por lo tanto, en lugar de depender de la infraestructura de red MPLS, es factible implementar una red de área amplia mediante el paradigma de redes definidas por software (SD-WAN). Esta tecnología no solo interconecta sucursales distantes, sino que también ofrece una serie de ventajas en comparación con MPLS. Por ejemplo, la reducción de costos, al utilizar conexiones de Internet (*VPN*), en lugar de las líneas dedicadas. Además, puede optimizar el rendimiento de la red al emplear múltiples conexiones de Internet. Asimismo, proporciona una mayor visibilidad y control sobre la red, lo que permite a los administradores monitorizar y gestionar el tráfico de manera eficiente.

SD-WAN es un tipo de tecnología de red que utiliza los principios de las redes definidas por

software (SDN). Sobre la base de este paradigma se puede gestionar y optimizar el rendimiento de las redes virtuales de área extensa (V-WAN) [2]. SD-WAN no tiene una única arquitectura, sin embargo, todas las implementaciones comparten ciertos elementos en común. Por jemplo, incluyen un dispositivo *edge*, que ejecuta funciones de red virtual (VNF) y puede implementarse en plataformas de nube pública o privada de una organización; un controlador, que permite a los operadores administrar la red a través de un único panel; y por último, un orquestador, que es el administrador virtualizado de la red encargado de supervisar el tráfico y la aplicación de políticas y protocolos establecidos por los operadores de red [3].

Numerosas empresas ofrecen soluciones SD-WAN, cada una con sus propias características y enfoques. Entre los principales proveedores se encuentran Fortinet, VMware, Versa Networks y Palo Alto Networks. Sin embargo, hasta nuestro conocimiento, no existe una alternativa de código abierto.

Ante la falta de alternativas SD-WAN de código abierto, en esta tesis, se plantea desarrollar una propuesta de SD-WAN en dos etapas. Específicamente, en esta tesis solo se abarca la primera etapa que consiste en implementar una red virtual de área amplia (*backbone* de la SD-WAN). Para ello, se pone en funcionamiento dos nubes privadas orquestadas con OpenStack en dos ubicaciones distantes, ambas bajo Ubuntu Server 22.04 LTS. En cada ubicación, se crea una infraestructura virtual de red con Open vSwitch y máquinas virtuales creadas con OpenStack, las cuales representan las dos sucursales de una organización. Además, se añade un usuario que se conecta a la nube privada desde un punto externo. Finalmente, se establece la conexión entre las dos nubes privadas y el usuario externo mediante una *VPN* que utiliza el protocolo WireGuard. Se utiliza WireGuard debido a que es un protocolo *VPN* OpenSource que utiliza cifrado ChaCha20, y se destaca por su rapidez en comparación con OpenVPN o IPsec. La segunda etapa de esta propuesta de SD-WAN será abordada en trabajos futuros, esta etapa abarcará el desarrollo de un controlador, un orquestador y la implementación de medidas de seguridad adicionales, como *firewalls*, *proxies* y *kerberos*.

1.1. Justificación

Durante la crisis de COVID-19, varias instituciones de educación superior (IES) tuvieron que suspender temporalmente servicios como pagos, trámites escolares y administrativos. Además, muchas de estas instituciones tienen sedes distantes cuyos trámites dependen de una única dependencia centralizada, a la cual no todas las sedes tienen acceso. En este sentido, una SD-WAN permitiría el acceso a recursos desde varias sedes o incluso desde cualquier parte del mundo cuando la V-WAN está basada en un modelo de nube. Esto a su vez ayuda a que clases en linea, la administración, las aplicaciones educativas, las inscripciones y muchas tareas más se puedan realizar a través de Internet, no importando si la institución está cerrada por un paro u otra situación de fuerza mayor. Por esta razón, es de suma importancia que las IES centren sus esfuerzos en adoptar nuevas tecnologías como SD-WAN y las nubes privadas, las cuales pueden ser accesibles desde cualquier lugar, con el fin de compartir sus recursos internos independientemente de la ubicación de sus sedes.

1.2. Objetivo

Implementar una red virtual de área amplia (V-WAN) a partir de la interconexión de dos servidores de nube privada orquestados por OpenStack y un VPS (*Virtual Private Server*) mediante el uso del protocolo *VPN* WireGuard.

1.3. Metodología

El desarrollo de este proyecto de tesis consta de las siguientes etapas:

- Configurar dos equipos con *Ubuntu server* como servidores de virtualización con QEMU/KVM para la creación de los nodos de OpenStack.

- Instalar, configurar y adaptar los componentes de OpenStack para su uso con Open vSwitch.
- Crear infraestructura de red y máquinas virtuales (VM) con OpenStack en cada nube privada.
- Implementar una red virtual de área amplia al interconectar dos IaaS mediante el protocolo *VPN* WireGuard.
- Conectar a un usuario externo a la nube privada mediante una *VPN* WireGuard.
- Realizar pruebas de conectividad y de rendimiento en la V-WAN.

1.4. Estructura de la tesis

Para alcanzar el objetivo establecido en esta tesis, el contenido se estructura de la siguiente manera:

El capítulo 2 presenta los conceptos básicos que permiten la implementación de una red de área amplia bajo el paradigma de redes virtuales. Se describen las herramientas de código abierto que se utilizan en este proyecto de tesis como Open vSwitch, Certbot, OpenStack y WireGuard.

El capítulo 3 presenta la topología de red y se define su alcance. Se abordan los recursos utilizados en este proyecto de tesis, así como la arquitectura de OpenStack y las máquinas virtuales necesarias para implementar la nube privada.

El capítulo 4 describe la implementación de una nube privada orquestada con OpenStack bajo *Ubuntu Server* 22.04 *LTS*. Además, se detalla el procedimiento para crear infraestructura virtual de red utilizando Open vSwitch y máquinas virtuales mediante OpenStack.

El capítulo 5 se da una explicación detallada de cómo implementar el servidor *VPN* y los clientes *VPN* (WireGuard).

El capítulo 6 realiza una evaluación de las funcionalidades de la red V-WAN planteadas en el capitulo 3.

El capítulo 7 describe las conclusiones generales y una descripción de los trabajos futuros.

Capítulo 2

Antecedentes

En este capítulo se presentan los conceptos básicos para entender cómo se implementa una red de área amplia bajo el paradigma de redes virtuales (V-WAN). También se describen las herramientas de código abierto que se utilizaron en este proyecto de tesis como Open vSwitch, Certbot, OpenStack y WireGuard.

2.1. LAN

Una red de área local (*LAN*) es un conjunto de dispositivos conectados entre sí en una ubicación física, como un edificio, una oficina o una casa. Una *LAN* puede contener a un usuario o a miles de usuarios y dispositivos en una oficina o escuela [45].

2.1.1. VXLAN

VXLAN, o *LAN* extensible virtual, es una tecnología de virtualización de red ampliamente utilizada en grandes redes de capa 2. *VXLAN* establece un túnel lógico entre los dispositivos de red origen y destino mediante encapsulación *MAC-in-UDP*. Específicamente, encapsula tramas *Ethernet* creadas por una VM en paquetes *UDP*. Posteriormente, encapsula los paquetes *UDP* con el encabezado *IP* y el encabezado Ethernet de la red física como encabezados externos, lo que permite que estos paquetes se encaminen a través de la red como paquetes *IP* comunes. Esto libera a las máquinas virtuales de las limitantes estructurales de las redes de capa 2 y 3. *VXLAN* utiliza el campo *VNI* de 24 bits para identificar hasta 16 millones de inquilinos, mucho más que el admitido por *VLAN* (alrededor de 4000 inquilinos) [42].

2.2. WAN

Una red de área amplia (WAN) es una red que conecta sedes de una institución a grandes distancias. Tanto las empresas como las *IES* suelen utilizar redes de área amplia para conectar sus oficinas; donde cada oficina suele tener su propia *LAN*. Estas *LAN* se conectan a través de una WAN. Estas conexiones se pueden crear de diferentes formas, las cuales incluyen líneas privadas alquiladas, *VPN*, o túneles *IP* [1].

2.3. SD-WAN

Una red de área amplia definida por software (SD-WAN) es un tipo de tecnología de red que utiliza principios de redes definidas por software (SDN). Permite a las organizaciones conectar de forma segura usuarios, aplicaciones y datos en múltiples ubicaciones, al mismo tiempo que proporciona rendimiento, confiabilidad y escalabilidad. SD-WAN también simplifica la gestión de las V-WAN al proporcionar control centralizado y visibilidad en toda la red [2].

La arquitectura SD-WAN, definida por el Foro MEF (un consorcio industrial sin fines de lucro), consta de un SD-WAN Edge, un controlador SD-WAN y un orquestador SD-WAN [3]. A continuación se explican estos elementos:

1. SD-WAN Edge es una función de red física o virtualizada (VNF) que se puede implementar en la sucursal/regional/oficina central, centro de datos y/o en plataformas de nube pública o privada de una organización.
2. El controlador SD-WAN centraliza la seguridad y la gestión. Permite a los operadores administrar la red a través de un único panel y establecer políticas para que las ejecute el orquestador.
3. SD-WAN Orchestrator es el administrador virtualizado de la red que supervisa el tráfico y la aplicación de políticas y protocolos establecidos por los operadores de red.

El orquestador y controlador de SD-WAN gestionan dinámicamente el flujo de datos y determinan el nivel de prioridad de las aplicaciones. Parte del tráfico está destinado a ubicaciones de Internet y normalmente no necesita permanecer en la red corporativa, por ejemplo, aplicaciones SaaS, tráfico web general y acceso a recursos de la nube pública. Sin embargo, el tráfico de aplicaciones críticas para el negocio y otras comunicaciones de alta prioridad, como llamadas sobre VoIP, puede utilizar conexiones *VPN* por motivos de seguridad, privacidad y rendimiento. Aunque también es posible optar por un enfoque híbrido, donde se continua utilizando enlaces MPLS junto con enlaces de Internet público.

Numerosas empresas ofrecen soluciones SD-WAN, cada una con sus propias características y enfoques. De acuerdo con el *Gartner Magic Quadrant* en el año 2023, los proveedores líderes son Fortinet, VMware, Versa Network y Palo Alto [4].

2.4. Certbot

Certbot es una herramienta de código abierto para solicitar certificados Let's Encrypt e implementar sitios web que requieran habilitar el protocolo *HTTPS* [5].

Certbot tiene soporte para certificados wildcard a partir de la versión 0.22.0. Para obtener un certificado wildcard es necesario utilizar el método de autenticación *DNS*, ya sea vía --manual o mediante un DNS plugin de Certbot apropiado para el proveedor de *DNS* [8].

2.4.1. Certificado Wildcard

Un certificado wildcard es un certificado que incluye uno o más nombres que comienzan con la expresión regular `*.`. Los navegadores aceptarán cualquier etiqueta en lugar de la expresión regular (`*.`). Por ejemplo, un certificado para `*.example.com` será válido para www.example.com, mail.example.com, hello.example.com y goodbye.example.com. Además, la expresión regular (`*`) sólo puede sustituirse por una única etiqueta y no por varias etiquetas consecutivas, por ejemplo, `*.*.example.com` no es válido [7].

2.4.2. Let's Encrypt

Let's Encrypt es una autoridad certificadora (CA) automatizada y abierta. Este servicio fue creado sin fines de lucro por *Internet Security Research Group* (ISRG). Permite la creación de certificados digitales para habilitar *HTTPS* sobre *SSL/TLS* en sitios web [6].

2.5. Virtualización

La virtualización es una tecnología que permite emular los recursos de cómputo como servidores, *routers* y *switches*, es decir, mediante *software* la virtualización emula las funciones del *hardware* para que sean procesados en una máquina física. Este concepto es utilizado para reducir los recursos de *hardware* y por ende decrementar los costos en la compra de recursos físicos [13].

2.5.1. Hipervisor

Un hipervisor es una capa de *software* que coordina las máquinas virtuales (*VM*). Sirve como una interfaz entre la máquina virtual y el hardware físico subyacente, lo que garantiza que cada *VM* tenga acceso a los recursos físicos. También garantiza que las *VM* no interfieran entre sí. Esto lo logra al reservar el espacio de memoria y los hilos de procesamiento [15].

Los hipervisores se pueden dividir en dos grupos [16].

- hipervisor tipo 1:

  Un hipervisor tipo 1 se ejecuta de manera directa en el *hardware* físico del equipo subyacente, interactuando directamente con su *CPU*, memoria y almacenamiento físico. Por este motivo, los hipervisores de tipo 1 también se denominan hipervisores *bare metal*. Un hipervisor de tipo 1 toma el lugar del sistema operativo principal.

  Los hipervisores tipo 1 son altamente eficientes porque tienen acceso directo al *hardware* físico. Esto también aumenta su seguridad, porque no hay nada entre ellos y la *CPU*. Ejemplos de hipervisores tipo 1 son: `VMware ESXi`, `VMware VSphere`, `Hyper-V`, `XenServer` y `KVM`.

- hipervisor tipo 2:

  Un hipervisor tipo 2 no se ejecuta directamente en el *hardware* subyacente. Se ejecuta como una aplicación del sistema operativo. Los hipervisores tipo 2 no suelen ser usados en entornos de producción. En cambio, son adecuados para entornos individuales, donde un usuario requiere ejecutar múltiples sistemas operativos. Ejemplos de hipervisores tipo 2 son: `VirtualBox`, `VMware Workstation` y `VMware Fusion`.

2.5.2. KVM

Kernel-based Virtual Machine (`KVM`) es una tecnología de virtualización *open source* integrada a Linux. Esta tecnología permite que un servidor Linux se pueda transformar en un hipervisor tipo 1 (*bare-metal*) que permite que la máquina *host* (sistema principal) ejecute varios entornos virtuales llamados máquinas virtuales o *guests* [14]. `KVM` se convirtió en parte principal del *kernel* de Linux en 2007 y complementa a `QEMU`, que es un hipervisor que emula el procesador de la máquina física completamente en software [16].

`KVM` es compatible con los procesadores Intel y *AMD* que incorporaron virtualización. Estas son llamadas *Intel VT* y *AMD-V*, respectivamente, y permiten que el procesador ayude al hipervisor a gestionar varias máquinas virtuales. Solo cuando estas extensiones están disponibles, el *kernel* de Linux puede utilizar `KVM`. De lo contrario se debe utilizar `QEMU`[16].

2.6. Open vSwitch

`Open vSwitch` es un módulo del *kernel* de Linux que permite emular un *switch* virtual multicapa con licencia de código abierto. `Open vSwitch` admite múltiples tecnologías de virtualización basadas en Linux, incluidas `KVM`, `XenServer` y `VirtualBox`. `Open vSwitch` admite las siguientes funciones [17]:

- *OpenFlow* 1.0 y sus extensiones.
- *Geneve, GRE, VXLAN, STT, ERSPAN, GTP-U, SRv6, Bareudp y tunelización LISP.*
- Configuración de *QoS* (calidad de servicio), además de vigilancia.
- Gestión de fallos de conectividad *802.1ag*.
- Reenvío de alto rendimiento mediante un módulo del kernel de Linux.
- *NetFlow*, *sFlow(R)* y *mirroring* para una mayor visibilidad.

Algunos de los componentes (*daemon* Linux) y herramientas de `Open vSwitch` son [17]:

- `ovs-vswitchd`: un *daemon* que implementa el *switch* junto con un módulo complementario del *kernel* de Linux para conmutación basada en flujo.

- `ovs-vsctl`: una herramienta para consultar y actualizar la configuración de *ovs-vswitchd*.
- `ovs-ofctl`: una herramienta para consultar y controlar los *switches* y controladores de *OpenFlow*.

2.7. IaaS

La infraestructura como servicio (`IaaS`) es un modelo de servicio en la nube que ofrece recursos de infraestructura bajo demanda, como servicios de procesamiento, almacenamiento, redes y virtualización, a empresas y particulares a través de la nube. La `IaaS` ayudan a eliminar gran parte de la complejidad y los costos asociados a la construcción y al mantenimiento de la infraestructura física en un centro de datos [11].

2.8. OpenStack

`OpenStack` fue lanzado originalmente en julio de 2010 por Rackspace y la NASA como una iniciativa de código abierto que combinaba la plataforma *Nebula* de la NASA y la plataforma *Cloud Files de Rackspace*. `OpenStack` es una plataforma de código abierto en la nube que administra recursos informáticos y de almacenamiento distribuidos. Permite la creación de entornos virtualizados bajo demanda mediante un panel de control basado en una página web [10]. En resumen, `OpenStack` es una solución de infraestructura como servicio (`IasS`) que se compone de un conjunto de módulos interrelacionados.

Dentro de `OpenStack`, hay varios servicios disponibles. La figura 2.1 muestra los seis módulos más importantes de `OpenStack`.

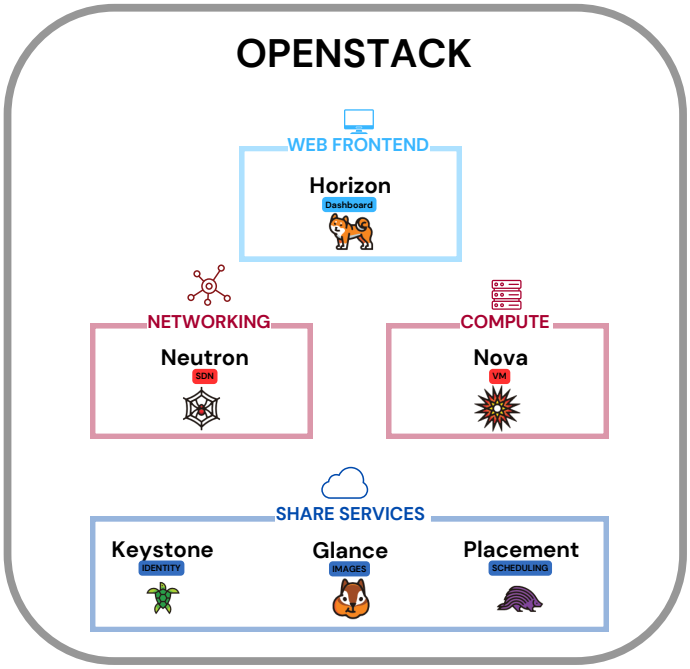

Figura 2.1: Módulos de OpenStack.

A continuación se describe el funcionamiento de cada uno de los módulos mostrados en la figura 2.1:

1. Keystone: identity service

   `Keystone` es el servicio responsable de gestionar la autenticación y de autorizar el acceso de los usuarios a los diversos servicios de `OpenStack`. También tiene a su cargo la gestión

de bases de datos, así como los catálogos de servicios de OpenStack y sus API *endpoints* [20].

Para comprender la implementación de Keystone, es necesario definir los elementos más importantes propios de este servicio [20]:

- *Users*: los usuarios son representaciones de clientes que utiliza OpenStack.
- *Roles*: los roles determinan las acciones que puede llevar a cabo los usuarios. Por defecto, existen dos roles predefinidos: administrativo y no administrativo.
  Existen varios niveles de permisos y es posible crear múltiples roles según las necesidades del administrador.
- *Projects*: los proyectos son la unidad fundamental de OpenStack. Todos los recursos de OpenStack deben estar asociados a un proyecto específico para su agrupación o aislamiento.
- *Domains*: los dominios representan la agrupación de proyectos, usuarios y grupos en OpenStack.
- *Token Fernet*: los tokens Fernet son fragmentos de texto que contienen una cantidad limitada de información, como la identidad del usuario y el proyecto. Estos tokens se utilizan para acceder a recursos y tienen una validez por un corto período de tiempo. Los datos dentro de un token de *Fernet* están protegidos mediante claves de cifrado simétrico.

2. Glance: image service

   Glance es el servicio encargado de almacenar, administrar y recuperar las imágenes en la nube para las instancias (Sistemas Operativos) que se ejecutan en OpenStack [10]

   El servicio de imágenes de OpenStack incluye los siguientes componentes [18]:

   - Glance-apiGlance-api: acepta peticiones para el descubrimiento, la recuperación y el almacenamiento de imágenes.
   - Glance-registry: procesa y recupera metadatos sobre las imágenes.
   - Repositorio de almacenamiento.

3. Placement:

   Placement es un servicio de OpenStack que ofrece una API HTTP para rastrear el inventario y el uso de recursos en la nube (memoria, espacio en disco duro, número de procesadores). Esto ayuda a otros servicios a administrar y asignar sus recursos de manera más efectiva [49].

4. Nova: compute service

   Nova es un servicio de OpenStack que ofrece a los usuarios instancias virtuales (máquinas virtuales) para ejecutar aplicaciones en ellas [23]. Nova admite una amplia gama de hipervisores, incluidos QEMU/KVM, Hyper-V y VMware ESXi [10].

5. Neutron: networking service

   Neutron es un servicio de OpenStack que proporcionar redes como servicios (NaaS)". Neutron proporciona una API que permite a los usuarios crear topologías de red, configurar y definir conectividad de red, configurar políticas de red y direccionamiento en la nube [24].

   Neutron maneja la creación y administración de la infraestructura de red virtual, incluidas *switches*, subredes y *routers* para conectar los dispositivos administrados por Nova.

   Neutron proporciona conectividad de red entre instancias de OpenStack. Para ello, Neutron utiliza varias tecnologías de redes definidas por software (*SDN*), incluidas *Open Virtual Network* (*OVN*), Open vSwitch (OVS), *Juniper Contrail*, etc [10].

   Dentro de OpenStack, existen tres modelos de redes [27]:

- `Management network`: se utiliza para la comunicación interna entre los componentes de `OpenStack`.
- `Provider networks`: el administrador de `OpenStack` crea las redes de proveedores y las asigna directamente a una red física existente.
- `Self-service network`: las *self-service network* son creadas por los usuarios para la conectividad dentro de los proyectos. Por defecto, están completamente aisladas y no se comparten con otros proyectos.

A continuación, se presenta los componentes esenciales de `Neutron`[26]:

- Neutron-server: es el *daemon* de Linux escrito en Python que dirige las solicitudes de las VM.
- Plugin agent: se ejecuta en cada nodo de cómputo para administrar la configuración del *swtich* virtual (vswitch).
- DHCP agent: proporciona servicios DHCP a las redes internas dentro de `OpenStack`.
- L3 agent: proporciona servicios de capa 3 para que las máquinas virtuales tengan acceso a la red externa.
- Message queue: servicio de mensajes para coordinar operaciones e informar del estado entre servicios [32].

6. Horizon:

   `Horizon` es la interfaz de usuario web de `OpenStack` basada en Django, funciona como un panel de control que facilita el acceso y la gestión de servicios (`Keystone`, `Glance`, `Placement`, `Nova` y `Neutron`) [28].

2.9. WireGuard

`WireGuard` es un protocolo de red capaz de crear túneles *VPN*. Está sujeto a la licencia *GPLv2* de software libre y es multiplataforma. `WireGuard` está escrito en los lenguajes *C* y *Go* [35]. Este protocolo utiliza las siguientes funciones de cifrado: *ChaCha*20 para cifrado simétrico, autentica con *Poly*1305, usa *Curve*25519 para intercambio de claves mediante curvas elípticas. Utiliza *BLAKE*2 para funciones *hash* y *hash* con clave. También usa *SipHash*24 para claves de tabla *hash*. Finalmente, usa *HKDF* para derivación de claves [37].

Una de las características principales del protocolo `WireGuard` es el llamado `Cryptokey Routing` que funciona asociando claves públicas con una lista de direcciones *IP* permitidas dentro del túnel. Cada interfaz de red tiene una clave privada y una lista de pares. Cada par tiene una clave pública. Las claves públicas son utilizadas por los pares para autenticarse entre sí [38].

`WireGuard` encapsula de forma segura paquetes `IP` a través de *UDP*, sin embargo, no admite explícitamente túneles sobre *TCP*, dado que evita la pérdida de rendimiento sobre *TCP*. No obstante, si se desea usar *TCP*, se puede utilizar *udptunnel* y *udp2raw* [39].

2.9.1. Interfaces de red WireGuard

`WireGuard` se puede pensar como una interfaz de red. Esta tiene atributos como dirección *IP* y encaminamiento entre dominios *CIDR*. Sin embargo, tiene atributos específicos para el manejo de la *VPN*. Estos atributos se muestran a continuación [36]:

- `Private key`: es la clave privada de la interfaz codificada en hexadecimal.
- `Listen port`: es un entero correspondiente al puerto de escucha de la interfaz.
- `[Peers]`:
  - *a*) `Public key`: es la clave pública codificada en hexadecimal.
  - *b*) `Endpoint`: es una tupla (*IP*, puerto) para *IPv4* que indica dónde enviar el tráfico cifrado.
  - *c*) `Allowed_IP`: es una lista de redes o direcciones destino del túnel para enviar o recibir tráfico.

2.9.2. Descripción del funcionamiento de WireGuard

`WireGuard` es un protocolo *VPN* descentralizado *peer-to-peer* (de pares) que es capaz de abrir un túnel entre dos o varios enlaces (*peers*) [40]. Las conexiones en `WireGuard` funciona de forma similar al servicio *Secure Shell* (SSH), donde los usuarios generan claves públicas y las intercambian entre ellos. Mediante este intercambio se pueden identificar y cifrar los mensajes [38].

De forma general el comportamiento de `WireGuard` cuando se envía un paquete a un peer es la siguiente [41]:

1. `WireGuard` lee la *IP* de destino del paquete y la compara con la lista de direcciones *IP* permitidas en la configuración local. Si no se encuentra el par, `WireGuard` descarta el paquete.
2. Si el par es válido, entonces cifra el paquete utilizando la clave pública del par. El emisor busca la dirección *IP* y envía el paquete cifrado.

Cuando `WireGuard` recibe un paquete [41]:

1. `WireGuard` descifra el paquete utilizando la clave privada.
2. Lee la dirección origen del paquete y busca si la *IP* está configurada en la lista de direcciones *IP* permitidas. Si la *IP* origen está en la lista , `WireGuard` acepta el paquete, caso contrario, lo descarta.

Capítulo 3

Construcción del entorno V-WAN

En este capítulo se presenta la topología de red y se define su alcance. Se abordan los recursos utilizados en este proyecto de tesis, así como la arquitectura de `OpenStack` y la creación de máquinas virtuales para implementar la nube privada.

3.1. Topología de la red V-WAN

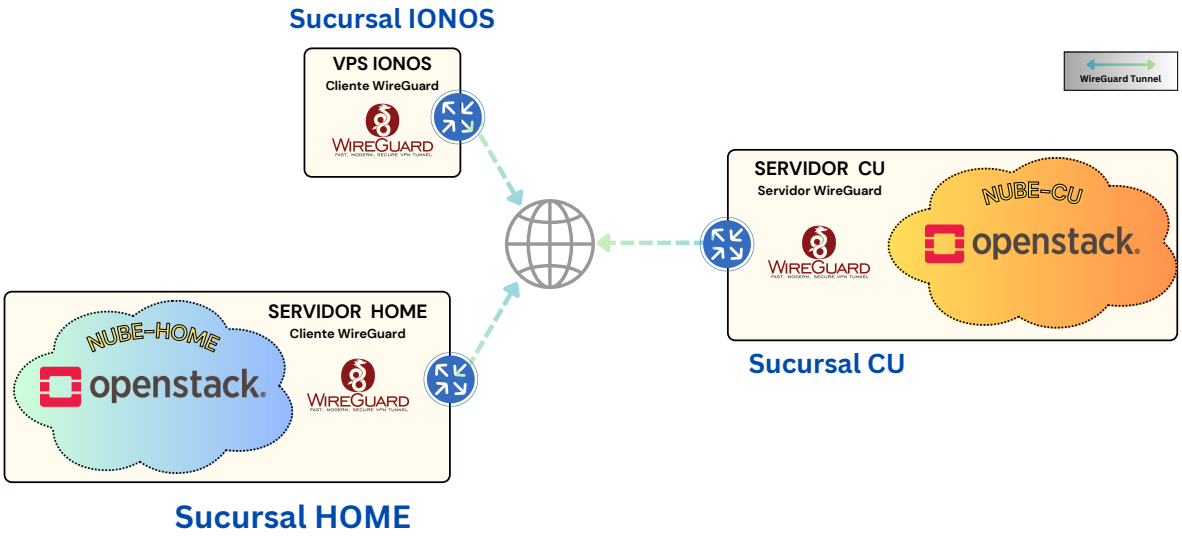

Figura 3.1: Topología de la red V-WAN.

La figura 3.1 muestra la topología de red utilizada en esta tesis (`V-WAN`). Esta topología está compuesta de 2 sucursales con una nube privada cada una y un cliente externo (sucursal IONOS). Los tres elementos se conectan mediante túneles `WireGuard`. La sucursal CU tiene el servidor `WireGuard`, mientras la sucursal HOME y el cliente externo (sucursal IONOS) son clientes de la *VPN*. La sucursal HOME como la sucursal CU cuentan con una nube privada orquestada por `OpenStack`. La sucursal IONOS (cliente externo) no cuenta con las características necesarias para implementar una nube (*hardware* limitado), por lo que este elemento no dispone de una nube `OpenStack`.

La topología de red de la NUBE-HOME se muestra en la figura 3.2.

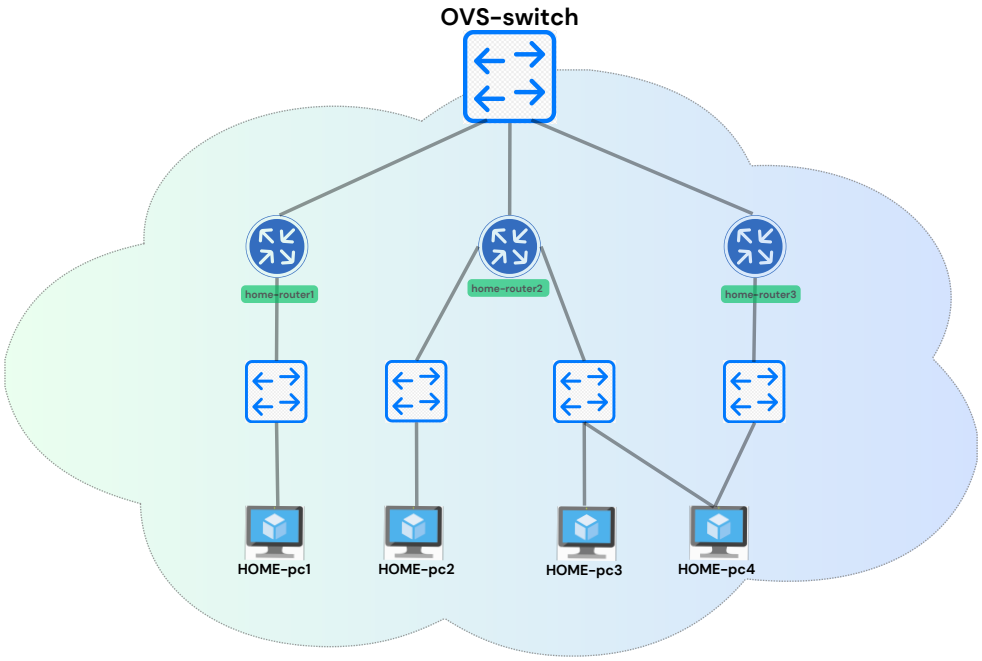

Figura 3.2: Topología de red de la NUBE-HOME.

La NUBE-HOME cuenta con un *switch* central virtualizado mediante `Open vSwitch`. Este *switch* actúa como punto principal para la conexión de tres *routers* creados dentro de `OpenStack`. Además, cuenta con cuatro *switches* adicionales, en los cuales están conectados a 4 máquinas virtuales de `OpenStack`.

La topología de red de la NUBE-CU se muestra en la figura 3.3.

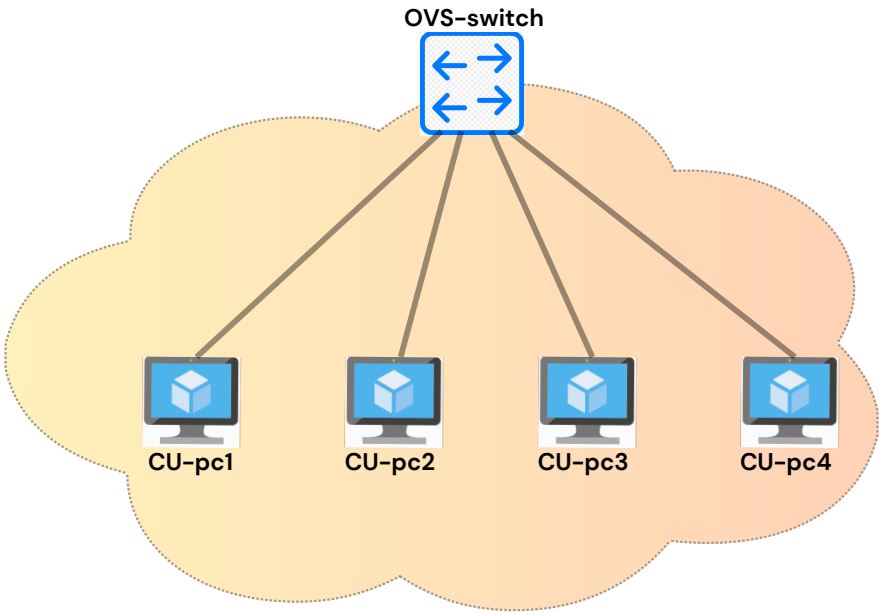

Figura 3.3: Topología de red de la NUBE-CU.

La NUBE-CU está compuesta por un *switch* central creado con `Open vSwitch`, el cual sirve como punto de conexión principal para 4 las máquinas virtuales que operan en el entorno de

OpenStack.

La unión entre la sucursal HOME, la sucursal CU y la sucursal IONOS conforman la red V-WAN, la cual cumple con las funciones que se describen a continuación (requisitos):

- **Función A**: En la Figura 3.4 se representa la función A de la red V-WAN, donde todas las máquinas virtuales de NUBE-HOME (consultar figura 3.2) se conectan a las máquinas virtuales de NUBE-CU (consultar figura 3.3) mediante el canal rojo (túnel *VPN*). De forma similar, el VPS IONOS establece conexión con las máquinas virtuales de la NUBE-CU (ver figura 3.3) mediante el canal morado, correspondiente a un túnel WireGuard.

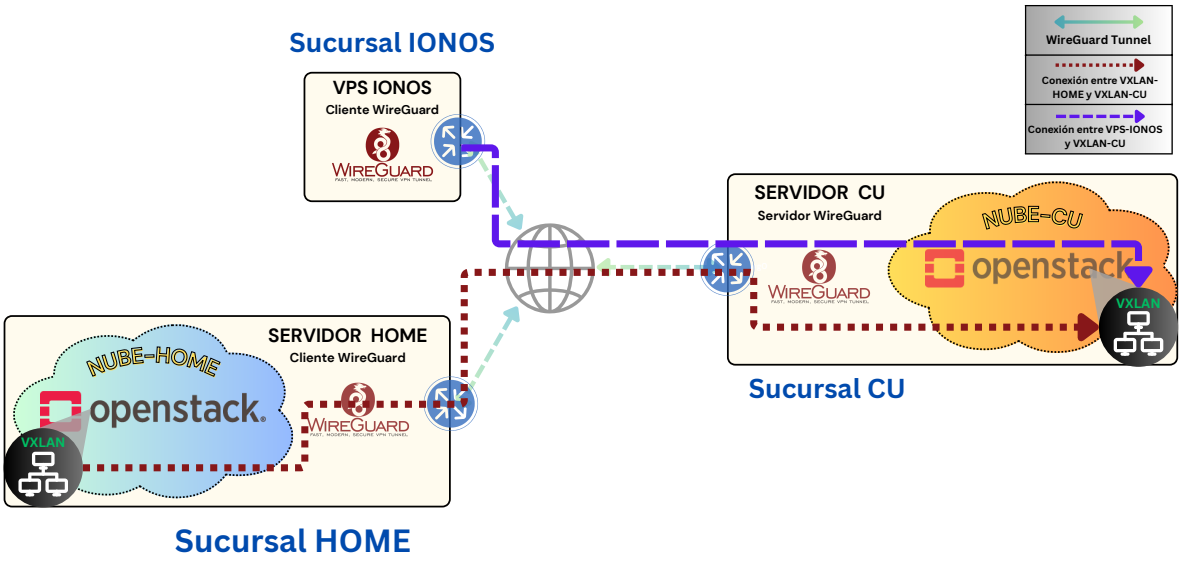

Figura 3.4: Función A de la red V-WAN.

- **Función B**: En la figura 3.5 se representa la función B de la red V-WAN, donde las máquinas virtuales de la NUBE-HOME (consultar figura 3.2) tienen salida a Internet (canal rojo) a través del servidor CU (servidor WireGuard). Igualmente, el VPS IONOS cuenta con acceso a Internet (representado por el canal morado) a través del servidor CU (servidor WireGuard). Es fundamental destacar que los paquetes de datos provenientes de las máquinas virtuales y del VPS IONOS van cifrados y tendrán la *IP* de la sucursal CU en lugar de su *IP* pública.

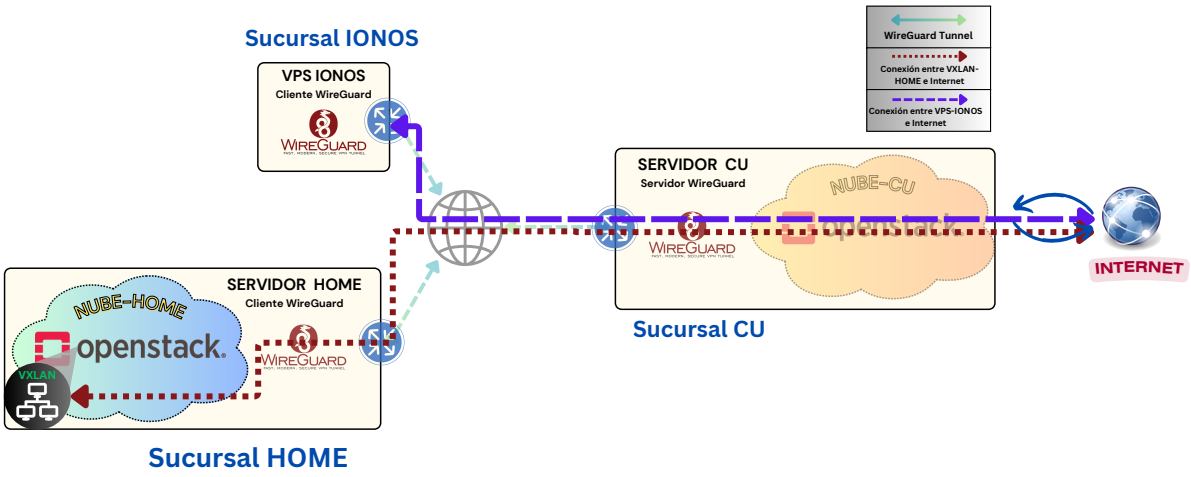

Figura 3.5: Función B de la red V-WAN.

- **Función C**: En la figura 3.6 se muestra la función C de la red `V-WAN`. El canal azul representa la comunicación bidireccional entre el servidor HOME y el VPS IONOS mediante un túnel *VPN*. El canal rojo representa la comunicación bidireccional entre el servidor HOME y el servidor CU a través de un túnel *VPN*. Finalmente, el canal morado indica la comunicación bidireccional entre el VPS IONOS y el servidor CU mediante un túnel *VPN*.

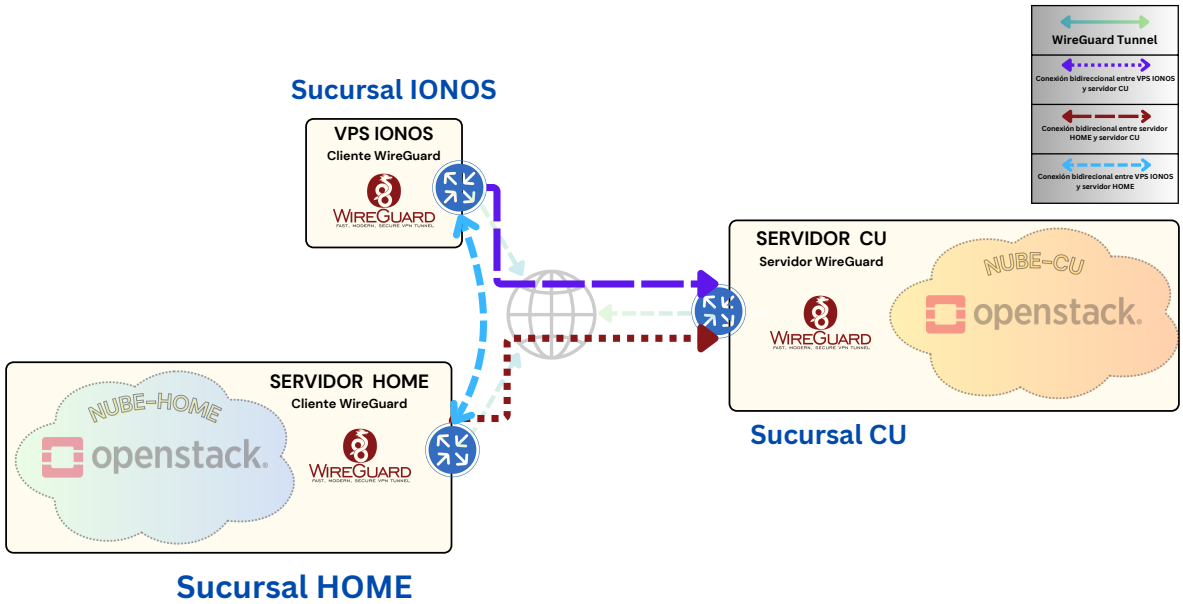

Figura 3.6: Función C de la red V-WAN.

Los recursos disponibles y particularidades de cada sucursal se describen a continuación.

3.1.1. Sucursal CU

La sucursal CU dispone de la *IP* pública 132.248.59.195, y cuenta con un servidor fisico sobre el que se monta `OpenStack` y el servidor `WireGuard`. Para esta tesis, esta máquina se le denomina servidor CU. El sistema operativo elegido para el servidor CU es Debian 11 "*bullseye*". Este sistema operativo funge como el hipervisor o máquina *host*. En la tabla 3.1, se detallan las características de *hardware* del servidor CU, obtenidas mediante los comandos `lscpu`, `dmidecode` y `lshw`.

ESPECIFICACIONES	SERVIDOR CU
Sistema operativo	Debian GNU/Linux 11 (bullseye)
CPU	Intel(R) Xeon(R) Silver 4214R 2.40 GHz
IP	IP Privada = 192.168.10.2
RAM	256 GB
Almacenamieto	SSD de 960 GB y HDD de 2 TB

Tabla 3.1: Especificaciones del servidor CU.

3.1.2. Sucursal HOME

La sucursal HOME dispone de una *IP* dinámica que otorga el *ISP* de `TELMEX`. Esta máquina es un servidor fisico en la cual se monta `OpenStack` y un cliente de `WireGuard`. Esta máquina es denominada servidor HOME. Del mismo modo que el servidor CU, el sistema operativo elegido para el hipervisor es Debian 11 "bullseye". En la tabla 3.2 se detallan las características de *hardware* del servidor HOME.

ESPECIFICACIONES	SERVIDOR HOME
Sistema operativo	Debian GNU/Linux 11 (bullseye)
CPU	Intel(R) Core(TM) i5-3337U 1.80 GHz
IP	IP Privada = 192.168.0.65
RAM	12 GB
Almacenamiento	SSD de 480 GB

Tabla 3.2: Especificaciones del servidor HOME.

3.1.3. Sucursal IONOS

La sucursal IONOS es un servidor virtual privado (VPS) rentado a la empresa IONOS (empresa dedicada al *hosting* y *cloud* para las pymes). De aquí en adelante al cliente IONOS se le denominará VPS IONOS.

El objetivo de utilizar el VPS IONOS es crear un usuario externo que se conecta a la `WAN`. Debido a que este cliente consta de capacidades limitadas en *hardware*, solo funciona como un cliente de `WireGuard` y no cuenta con `OpenStack`. La tabla 3.3 muestra las características del VPS IONOS.

ESPECIFICACIONES	VPS IONOS
Sistema operativo	Debian GNU/Linux 11 (bullseye)
CPU	AMD EPYC-Milan @ 1.996 GHz; vCPUs = 1
Virtualización	KVM/QEMU
IP	82.165.209.114
RAM	1 GB
Almacenamiento	NVMe de 10 GB

Tabla 3.3: Especificaciones del VPS IONOS.

3.2. Arquitectura planteada para OpenStack

`OpenStack` proporciona la flexibilidad necesaria para diseñar la arquitectura de nube según las necesidades específicas que se deseen utilizar. En este proyecto de tesis se crean dos sucursales de una organización, donde las nubes tienen las siguientes características:

1. Capacidad para crear máquinas virtuales tanto *GNU/Linux* como *Windows*.
2. Tener posibilidad para escalar y potenciar la migración.
3. Capacidad de crear redes virtuales internas con servicios de capa 3 utilizando métodos de segmentación superpuesta como *VXLAN*. Así como la capacidad de encaminar redes virtuales a redes físicas mediante *NAT*.
4. Contar con servicio de *DHCP*.
5. Compatibilidad con *bridges* de `Open vSwitch`.

Para cumplir con las características planteadas, se implementa la versión `OpenStack` 2023.1 (Antelope) con los siguientes módulos:

- `Keystone`: identity service.
- `Glance`: image service.
- `Placement`: placement service.
- `Nova`: compute service.
- `Neutron`: networking service.
- `Horizon`: OpenStack dashboard.

En la figura 3.7 se presenta el resumen de la arquitectura de `OpenStack` propuesta para esta tesis.

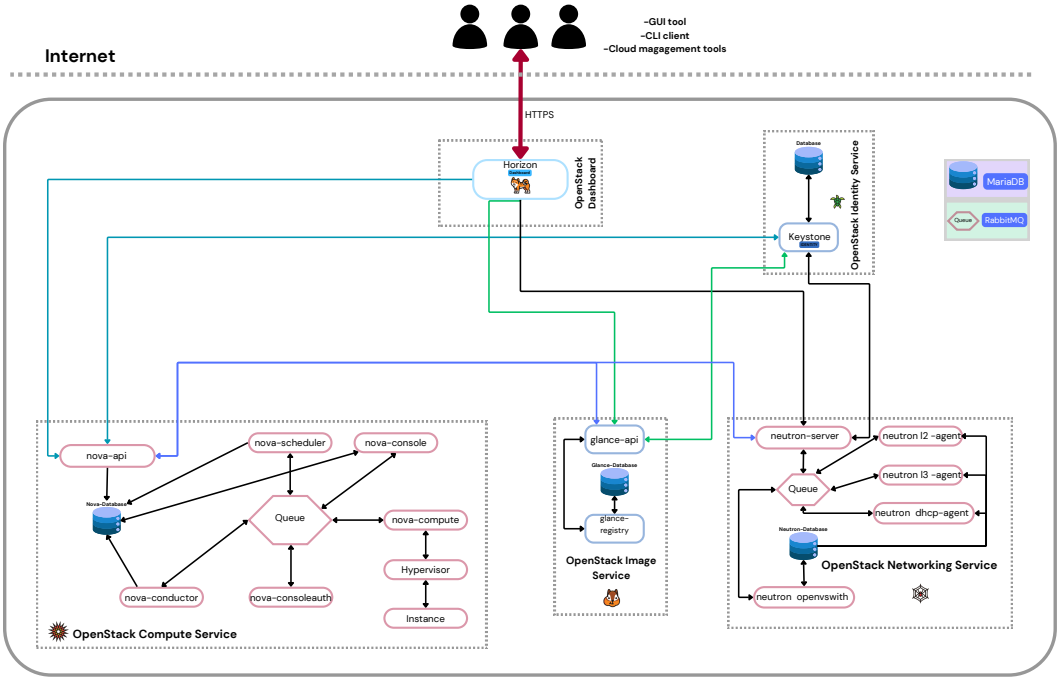

Figura 3.7: Resumen de la arquitectura para la NUBE-CU y NUBE-HOME.

Debido a la gran cantidad de procesos y servicios que ocurren internamente dentro de `OpenStack`, la figura 3.7 solo muestra los más importantes. En la arquitectura mostrada en la figura 3.7, se pueden observar los siguientes aspectos:

- Los usuarios pueden acceder a `OpenStack` mediante la interfaz de usuario basada en web implementada por el panel de control `Horizon` o través de la línea de comandos de clientes.
- En general, los servicios se comunican mediante una API, que escucha las solicitudes, las preprocesa y las envía a otros servicios.
- Todos los servicios se autentican a través del servicio de identidad (`Keystone`).
- Para la comunicación entre los procesos, se utiliza `RabbitMQ`.
- Se utiliza MariaDB como sistemas gestor de base de datos.

Una nube privada en `OpenStack` requieren de tres nodos independientes. Estos nodos se crean mediante la virtualización `KVM`. Es importante mencionar que si bien las *VM* facilitan una posible migración o cambio de sitio entre nubes, también reducen el rendimiento de los VPS. Sin embargo, para los objetivos de esta tesis dicha desventaja no anula la posibilidad de crear las sucursales.

Para distinguir entre los 3 nodos según sus funciones y servicios en `OpenStack`, se les nombra de la siguiente manera:

1. Nodo de control : como se puede observar en la figura 3.8, este nodo tiene la función de ser el servidor de la base de datos *SQL*, *NTP*, *RabbitMQ*, *Memcached* y *ETCD*. También maneja el servicio de identidad (`Keystone`), el servicio de imágenes (`Glance`), el servicio `Placement`, la administración de `Nova` y la administración de redes con `Neutron`.

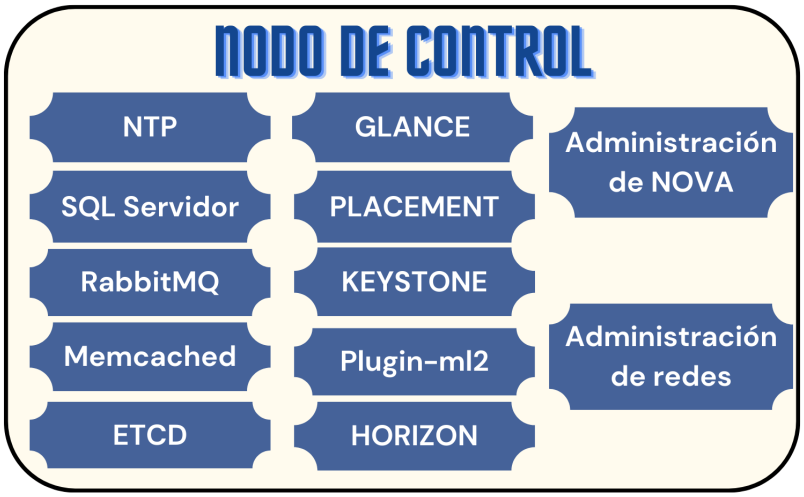

Figura 3.8: Servicios dentro del nodo de control.

2. Nodo de cómputo: como se puede observar en la figura 3.9, este nodo es el encargado de crear las máquinas virtuales en la nube de `OpenStack`. Ejecuta `Open vSwith` para que el servicio de red pueda conectar los VPS a la redes virtuales y proporciona servicios de *firewall*.

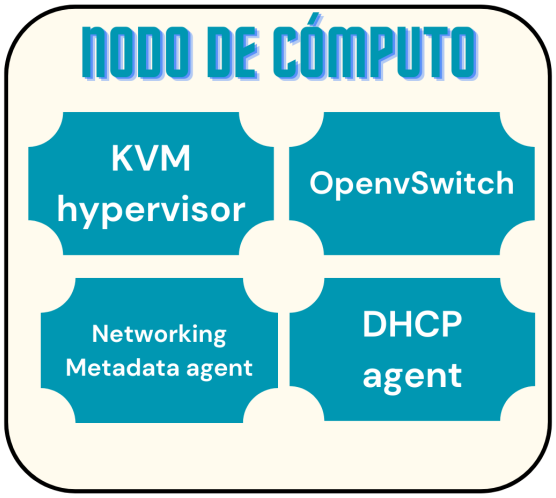

Figura 3.9: Servicios dentro del nodo de cómputo.

3. Nodo de red: como se puede observar en la figura 3.10, este nodo ejecuta `Open vSwith` para colaborar con el nodo de cómputo y así crear infraestructura de red capa 3, como *routers* virtuales e *IP* flotantes. Proporciona reenvío de capa 3 para permitir que las máquinas virtuales tengan acceso a la red externa.

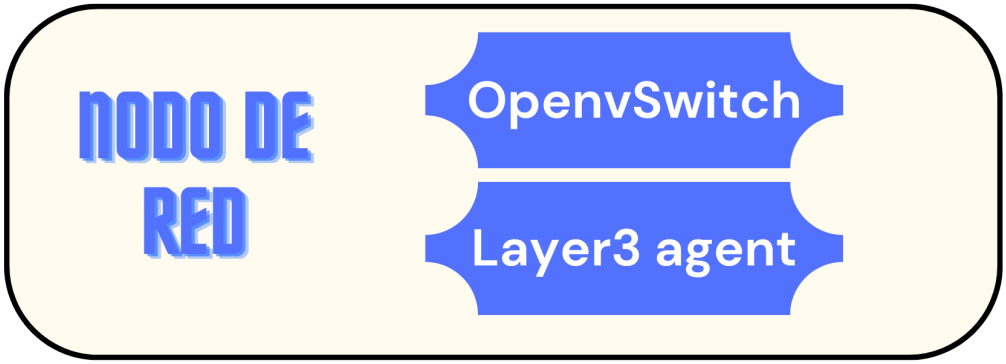

Figura 3.10: Servicios dentro del nodo de red.

En esta tesis, dado que cada nube necesita 3 nodos independientes, estos se nombran de la siguiente forma (ver tabla 3.4):

Sucursal CU	Sucursal HOME
Nodo de control-CU	Nodo de control-HOME
Nodo de cómputo-CU	Nodo de cómputo-HOME
Nodo de red-CU	Nodo de red-HOME

Tabla 3.4: Nombre de los nodos en cada sucursal.

La figura 3.11 presenta de manera conjunta los seis nodos necesarios para el funcionamiento de cada una de las nubes, así como sus servicios correspondientes, los cuales ya han sido mencionados previamente (consultar figura 3.8,3.9,3.10).

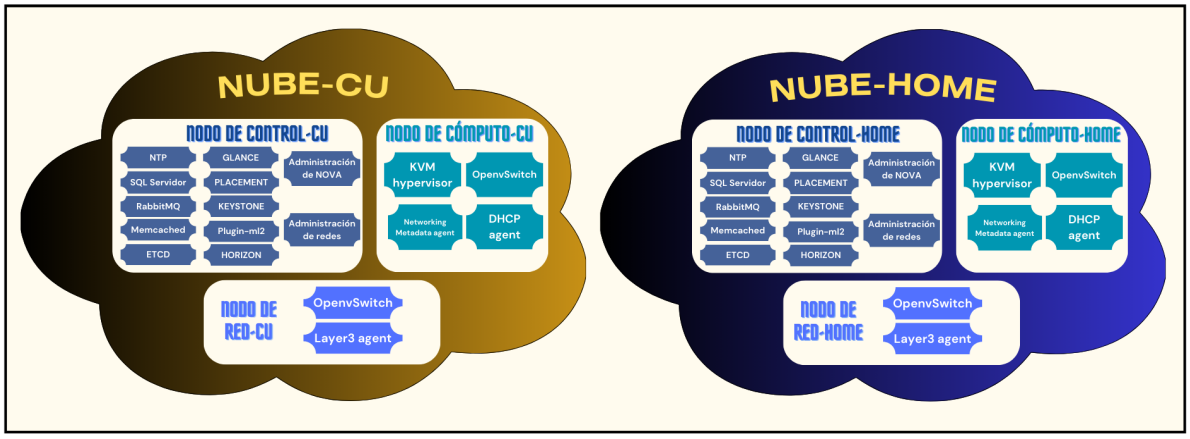

Figura 3.11: Nodos para NUBE-CU y NUBE-HOME.

3.3. Configuración del hipervisor KVM

Antes de comenzar a instalar `OpenStack`, se debe poner a punto el hipervisor. Para esta tesis se utiliza el hipervisor `QEMU/KVM` sobre el sistema Debian 11. Para ello, se puede seguir el proceso descrito del sitio [9]. De manera similar, para habilitar la virtualización anidada se puede seguir el método descrito en la url [29].

3.4. Nodos de OpenStack

En esta sección se presentan los tres nodos necesarios para implementar la NUBE-HOME, así como los tres nodos necesario para la NUBE-CU, utilizando el hipervisor `QEMU/KVM`. También se presenta un diagrama de red de cada sucursal con la presencia de los nodos de `OpenStack`, junto con un resumen detallado de las características de cada nodo.

3.4.1. Diagrama de red de la sucursal CU.

En la figura 3.12 se muestra el diagrama de red de la sucursal CU. En esta sucursal, los nodos de `OpenStack` tienen acceso a Internet a través de un *switch* virtual en modo puente (*bridged* `br0`), el cual está conectado a la interfaz física del servidor (`eno1`).

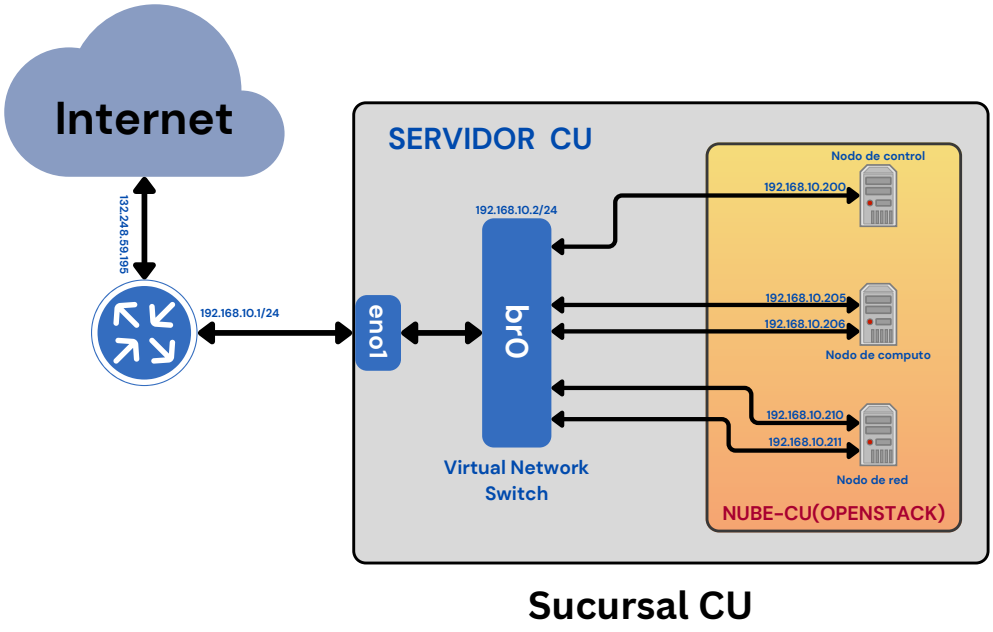

Figura 3.12: Diagrama de red de la sucursal CU.

3.4.2. Nodo de control-CU

El archivo de configuración `XML` del Nodo de control-CU se puede revisar en el apéndice A.1 y el resumen de sus características en la tabla 3.5.

ESPECIFICACIONES	NODO DE CONTROL-CU
Sistema operativo	Ubuntu Server 22.04.2 LTS
CPU	▪ Modo = host-passthrough; Modelo = Intel Xeon Silver 4214 ▪ vCPUs = 24
Interfaz de red 1	▪ Tipo = bridge; Modelo = virtio ▪ IP estática = 192.168.10.200
RAM	256 000 MB
Almacenamiento	▪ Tipo de bus = VirtIO ▪ Capacidad = 200 GB

Tabla 3.5: Especificaciones del nodo control-CU.

3.4.3. Nodo de cómputo-CU

El archivo de configuración XML del nodo cómputo-CU se puede revisar en el apéndice A.2 y el resumen de sus características en la tabla 3.6.

ESPECIFICACIONES	NODO DE CÓMPUTO-CU
Sistema operativo	Ubuntu Server 22.04.2 LTS
CPU	▪ Modo = host-passthrough; Modelo = Intel Xeon Silver 4214 ▪ vCPUs = 24
Interfaz de red 1	▪ Tipo = bridge; Modelo = virtio ▪ IP estática = 192.168.10.205
Interfaz de red 2	▪ Tipo = bridge; Modelo = virtio ▪ IP estática = 192.168.10.206
RAM	256 000 MB
Almacenamiento	▪ Tipo de bus = VirtIO ▪ Capacidad = 200 GB

Tabla 3.6: Especificaciones del nodo de cómputo-CU.

3.4.4. Nodo de red-CU

El archivo de configuración XML del nodo red-CU se puede revisar en el apéndice A.3 y el resumen de sus características en la tabla 3.7.

ESPECIFICACIONES	NODO DE RED-CU
Sistema operativo	Ubuntu Server 22.04.2 LTS
CPU	▪ Modo = host-passthrough; Modelo = Intel Xeon Silver 4214 ▪ vCPUs = 12
Interfaz de red 1	▪ Tipo = bridge; Modelo = virtio ▪ IP estática = 192.168.10.210
Interfaz de red 2	▪ Tipo = bridge; Modelo = virtio ▪ IP estática = 192.168.10.211
RAM	163 840 MB
Almacenamiento	▪ Tipo de bus = VirtIO ▪ Capacidad = 50 GB

Tabla 3.7: Especificaciones del nodo red-CU.

3.4.5. Diagrama de red de la sucursal HOME.

La figura 3.13 muestra el diagrama de red de la sucursal HOME. En esta configuración, se utiliza una NAT (`libvirt` como una red interna de la red 192.168.1.0/24, que será utilizada como red del proveedor para `OpenStack`). De igual manera, todos los nodos tienen acceso a la red 192.168.0.0/24 a través de un *switch* virtual en modo puente (*bridged* `br0`).

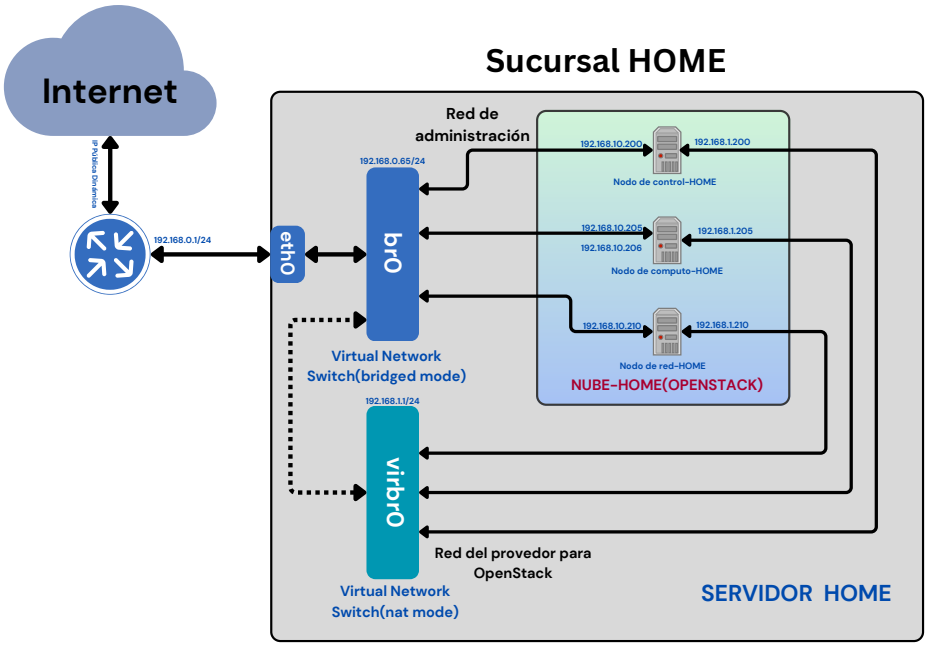

Figura 3.13: Diagrama de red de la sucursal HOME.

3.4.6. Nodo de control-HOME

El archivo de configuración `XML` del nodo control-HOME se puede revisar en el apéndice A.4 y el resumen de sus características en la tabla 3.8.

ESPECIFICACIONES	NODO DE CONTROL-HOME
Sistema operativo	Ubuntu Server 22.04.2 LTS
CPU	▪ Modo = host-passthrough ▪ Modelo = Intel i5-3337U @ 1.795 GHz; vCPUs = 4
Interfaz de red 1	▪ Tipo = bridge; Modelo = virtio ▪ IP estática = 192.168.0.200
Interfaz de red 2	▪ Tipo = NAT; Modelo = virtio ▪ IP estática = 192.168.1.200
RAM	11 846 MB
Almacenamiento	▪ Tipo de bus = VirtIO ▪ Capacidad = 100 GB

Tabla 3.8: Especificaciones del nodo control-HOME.

3.4.7. Nodo de cómputo-HOME

El archivo de configuración XML del nodo cómputo-HOME se puede revisar en el apéndice A.5 y el resumen de sus características en la tabla 3.9.

ESPECIFICACIONES	NODO DE CÓMPUTO-HOME
Sistema operativo	Ubuntu Server 22.04.2 LTS
CPU	▪ Modo = host-passthrough ▪ Modelo = Intel i5-3337U @ 1.795 GHz; vCPUs = 4
Interfaz de red 1	▪ Tipo = bridge; Modelo = virtio ▪ IP estática = 192.168.0.205
Interfaz de red 2	▪ Tipo = NAT; Modelo = virtio ▪ IP estática = 192.168.1.205
RAM	11 846 MB
Almacenamiento	▪ Tipo de bus = VirtIO ▪ Capacidad = 150 GB

Tabla 3.9: Especificaciones del nodo de cómputo-HOME.

3.4.8. Nodo de red-HOME

El archivo de configuración XML del nodo red-HOME se puede revisar en el apéndice A.6 y el resumen de sus características en la tabla 3.10.

ESPECIFICACIONES	NODO DE RED-HOME
Sistema operativo	Ubuntu Server 22.04.2 LTS
CPU	▪ Modo = host-passthrough ▪ Modelo = Intel i5-3337U @ 1.795 GHz; vCPUs = 4
Interfaz de red 1	▪ Tipo = bridge; Modelo = virtio ▪ IP estática = 192.168.0.210
Interfaz de red 2	▪ Tipo = NAT; Modelo = virtio ▪ IP estática = 192.168.1.210
RAM	11 846 MB
Almacenamiento	▪ Tipo de bus = VirtIO ▪ Capacidad = 50 GB

Tabla 3.10: Especificaciones del nodo red-HOME.

3.5. Servidor DNS

Para el servidor DNS, se adquirió el dominio `yohanpena.com` por medio de la empresa `Namecheap`. Esta empresa está acreditada por la Corporación de Internet para la Asignación de Nombres y Números `ICANN`. La razón principal por la cual se optó por comprar el dominio es debido a que el módulo `Horizon` maneja la interfaz de usuario basada en web [30]. Por lo que resulta más cómodo para los usuarios de la nube acceder a esta página por medio de un dominio y no de una *IP*.

3.5.1. Registro de IP dinámica

En la sección 3.1 se indicó que la sucursal HOME cuenta con una *IP* dinámica que otorga el ISP de TELMEX, por lo tanto, un `registro A` de DNS no sería adecuado porque al momento de cambiar la *IP*, el dominio dejaría de funcionar. Para solucionar este problema se utiliza el servicio gratuito `Dynamic DNS` que ofrece el proveedor `No-IP`. Los pasos que se siguieron para registrar la *IP* dinámica se describen a continuación:

1. Se crea una cuenta en `No-IP` y dentro de su plataforma, en el apartado `Dynamic DNS`, se genera un *Hostname* de nombre `unamcloud` como se ve la figura 3.14.

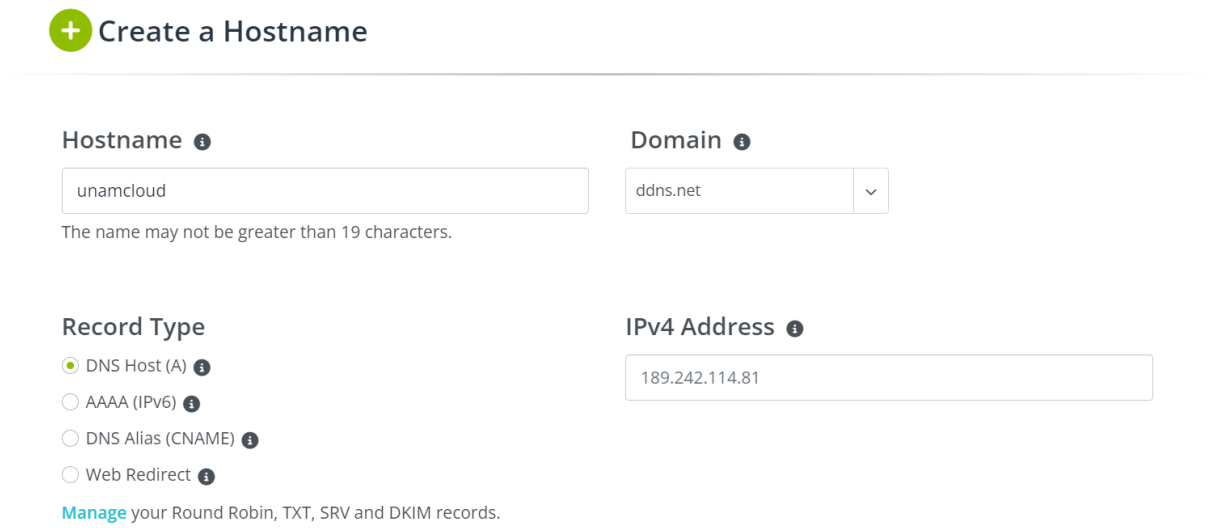

Figura 3.14: Panel de control de NO-IP.

2. Como se muestra en la figura 3.15, en las configuraciones del *router* de TELMEX se habilita la opción `Dynamic DNS` y se ingresan los datos necesarios de NO-IP.

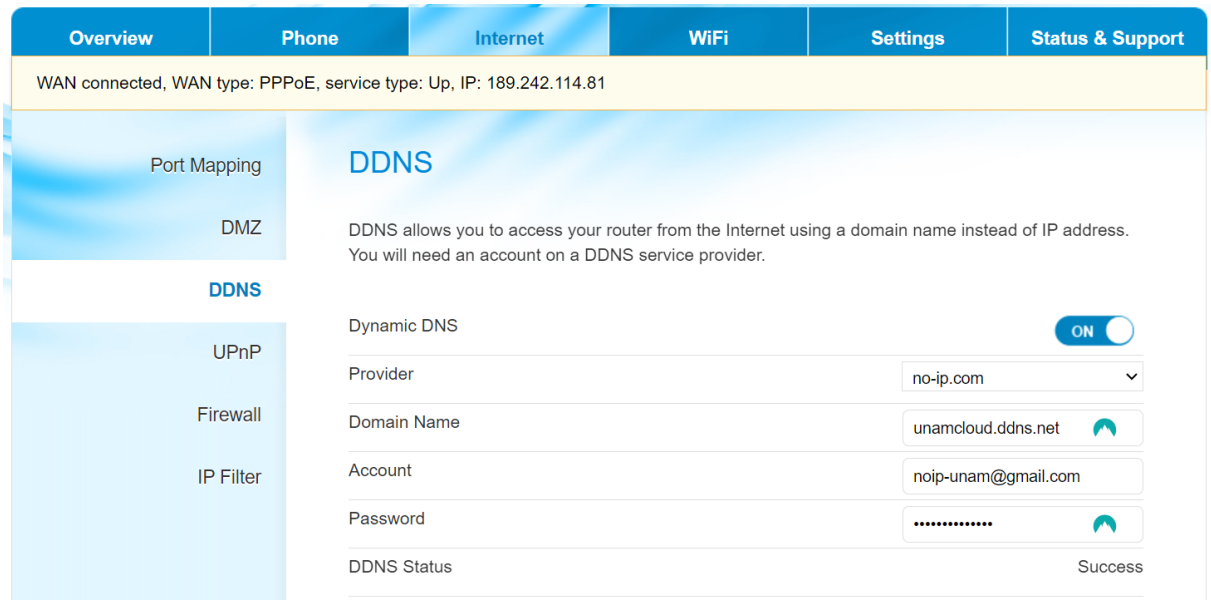

Figura 3.15: Configuración del *router* TELMEX

Una vez aceptada la nueva configuración del *router*, estará en funcionamiento el servicio de `Dynamic DNS`.

3.5.2. Registros de host en el panel de Namecheap

`Namecheap` por medio de su página web permite administrar los registros de domino. En la figura 3.16 se muestra este proceso, donde:

- yohanpena.com

  Es el dominio principal, se utiliza un registro tipo *A* para ligarlo con la *IP* pública de la sucursal CU.

- ionos.yohanpena.com

  Es el subdominio ionos que se vincula con la *IP* pública de la sucursal IONOS mediante un registro tipo *A*.

- home.yohanpena.com

  Utiliza un registro CNAME, se liga al subdominio `home` con el registro tipo *A* de NO-IP, hecho en la sección 3.5.1 que apunta a la *IP* dinámica de la sucursal HOME.

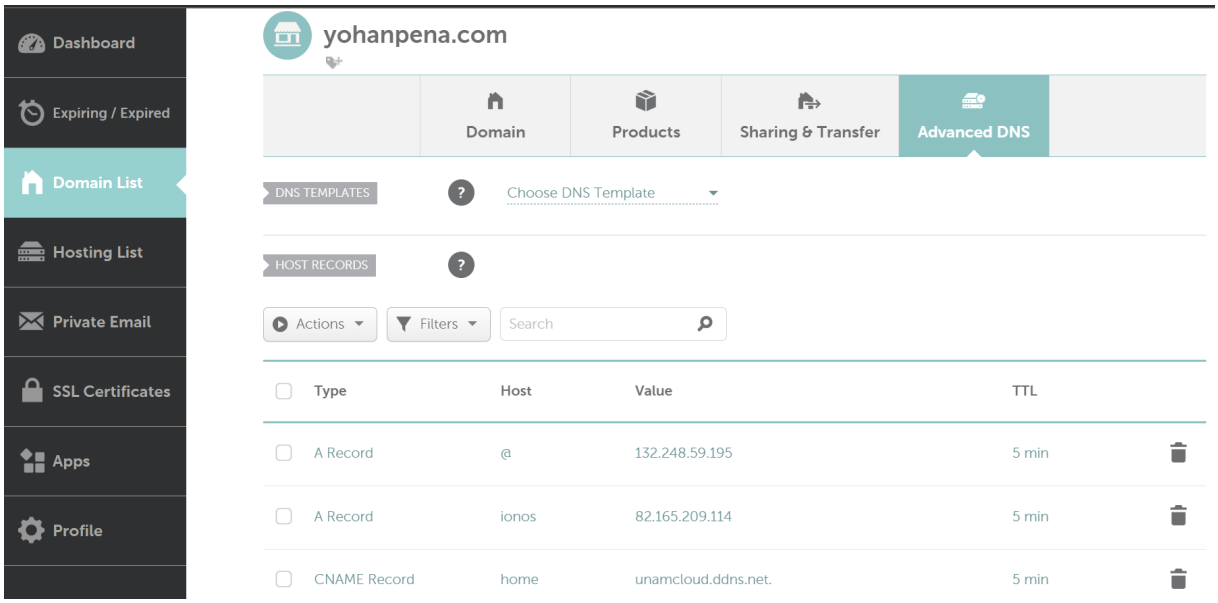

Figura 3.16: Panel de control de Namecheap.

3.5.3. Verificación del funcionamiento del dominio y subdominios

Se verifica el funcionamiento del dominio y de los subdominios utilizando el comando `dig` para consultar servidores *DNS*:

- yohanpena.com

  El dominio principal está apuntando a la *IP* de la sucursal CU como se muestra en la consola 3.1.

Consola 3.1: Comando dig para yohanpena.com.

```
root@servidorCU:# dig yohanpena.com
;; QUESTION SECTION:
;yohanpena.com.                 IN      A

;; ANSWER SECTION:
yohanpena.com.          300     IN      A       132.248.59.195
```

- ionos.yohanpena.com

  En la consola 3.2 se puede observar que el subdominio *ionos.yohanpena.com* está ligado a la *IP* de la sucursal IONOS.

Consola 3.2: Comando dig para ionos.yohanpena.com.

```
root@servidorCU:#dig ionos.yohanpena.com

;; QUESTION SECTION:
```

```
;ionos.yohanpena.com.            IN      A

;; ANSWER SECTION:
ionos.yohanpena.com.     300     IN      A       82.165.209.114
```

- home.yohanpena.com

  Finalmente, la consola 3.3 muestra la respuesta a la petición del dominio *home.yohanpena.com*.

Consola 3.3: Comando dig para home.yohanpena.com.

```
root@servidorCU:# dig home.yohanpena.com

;; QUESTION SECTION:
;home.yohanpena.com.            IN      A

;; ANSWER SECTION:
home.yohanpena.com.      300     IN      CNAME   unamcloud.ddns.net.
unamcloud.ddns.net.      60      IN      A       189.242.114.81
```

3.6. Obtención del certificado wildcard firmado por letsencrypt

Existen varias compañías que venden certificados *SSL/TLS*. Sin embargo, para esta tesis se optó por utilizar un certificado gratuito *wildcard* firmado por `letsencrypt`, esto con el fin de que tanto el panel de control (`Horizon`) de `OpenStack` como el servicio *SPICE* funcionen bajo *HTTPS*.

A continuación de describe el proceso para generar un certificado `wildcard` utilizando `certbot`:

1. Se instala `certbot` mediante el comando `snap` que se muestra en la consola 3.4.

Consola 3.4: Comando para instalar certbot

```
root@servidorCU:~# snap install --classic certbot
```

2. Una vez instalado `certbot` se utiliza una liga dinámica (ver consola 3.5) para que el binario de `certbot` sea visible para cualquier usuario en el directorio `/usr/bin`.

Consola 3.5: Comando ln

```
root@servidorCU:~# ln -s /snap/bin/certbot /usr/bin/certbot
```

3. Existen varios *plugins* de `certbot` que se pueden utilizar para generar certificados. Para este caso se realiza de forma manual, como se puede ver en la consola 3.6, porque se busca tener un solo certificado que ampare el dominio principal y cada uno de los subdominios. Si se hace de forma automática se genera un certificado por cada subdominio.

Consola 3.6: Generacion de certificado.

```
root@servidorCU:~# certbot certonly --manual -d *.yohanpena.com -d \
   yohanpena.com --register-unsafely-without-email --preferred-\
   challenges dns --agree-tos
```

donde,

- `-d *.yohanpena.com`: hace referencia a todos los subdominios de `yohanpena.com`
- `-d yohanpena.com`: indica el dominio principal.

4. Una vez hecha la petición del certificado es necesario validar la pertenecía del dominio. Para ello, es necesario superar una prueba que consisten en crear 2 registros `TXT` en el servidor *DNS*.

   En la consola 3.7 se muestra como `certbot` espera a que se completen los 2 registros.

Consola 3.7: Fragmento de solitud de desafios de certbot.

```
Requesting a certificate for *.yohanpena.com and yohanpena.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -\
    - - - - -
Please deploy a DNS TXT record under the name:

_acme-challenge.yohanpena.com.

with the following value:

CeMw_0r5sfwXvVjzOoStLBPSKSFI9Wqf6FKYsTEpPOA

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -\
    - - - - -
Press Enter to Continue

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -\
    - - - - -
Please deploy a DNS TXT record under the name:

_acme-challenge.yohanpena.com.

with the following value:

SMF_s2olgLVBRy5uQXGmX3IbEBXrPwZfqCeFyPQ6Ufk
```

Nota: la salida completa del comando se muestra en el apéndice D.2.

5. Una vez que se completa la prueba, se crean 2 registros `TXT` con los siguientes valores:

   - 1er registro:
     * Host : `_acme-challenge.irvingyohanan.com.`
     * Valor: `GqyOv2-Yr6roQg6v-gF9Mq5EqVEQA9n4d5jkTDO-xmg.`
   - 2do registro:
     * Host : `_acme-challenge.irvingyohanan.com.`
     * Valor: `RV9pmsT3vRdu35R5MtdBMnm-AGDA-YQ7j9BrAeQMmc`

6. Una vez completada la prueba se obtiene el certificado en formato PEM, cuya salida se muestra en la consola 3.8.

Consola 3.8: Fragmento de la salida de certbot.

```
Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/yohanpena.com/\
   fullchain.pem
Key is saved at:         /etc/letsencrypt/live/yohanpena.com/privkey.\
   pem
This certificate expires on 2024-04-25.
```

Nota: la fecha de expiración del certificado se muestra en la consola 3.8.

7. Tanto la llave privada como el certificado en formato PEM son guardados en el directorio `/etc/letsencrypt/live/yohanpena.com/`. Además de estos archivos, también se crean otros como se puede ver en la consola 3.9.

Consola 3.9: Archivos creados por certbot.

```
root@servidorCU:/etc/letsencrypt/live/yohanpena.com# ls
README  cert.pem  chain.pem  fullchain.pem  privkey.pem
```

El archivo `README` (ver apéndice D.1) explica el detalle de cada uno de los otros archivos.

Capítulo 4

OpenStack (IaaS)

En este capítulo se describe la implementación de una nube privada orquestada con `OpenStack 2023.1` (Antelope) bajo Ubuntu Server 22.04 LTS.

4.1. Configuración base

4.1.1. Host networking

Cada nodo debe ser capaz de identificar a los demás nodos por nombre y dirección *IP*, por lo tanto, se edita el archivo `/etc/hosts` de todos los nodos de `OpenStack` de la sucursal CU como se muestra en la consola 4.1.

Consola 4.1: Archivo /etc/hosts de los nodos de la sucursal CU.

```
127.0.0.1 localhost
# Nodo de control
192.168.10.200 controller
# Nodo de computo
192.168.10.205 compute
# Nodo de red
192.168.10.210 net-node

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

Igualmente, se edita el archivo `/etc/hosts` de todos los nodos de `OpenStack` de la sucursal HOME, tal como se muestra en la consola 4.2.

Consola 4.2: Archivo /etc/hosts de la sucursal HOME.

```
127.0.0.1 localhost
#controller
192.168.0.200    controller-home
#compute
192.168.0.205   compute-home
#net-node
192.168.0.210   netnode-home

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
```

```
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

4.1.2. Network Time Protocol (NTP)

Se debe garantizar que los nodos estén sincronizados mediante el protocolo *NTP*. Para ello, se realizan los siguientes pasos:

1. Se instala `chrony` en todos los nodos utilizando el comando que se muestra en la consola 4.3.

Consola 4.3: Instalación de chrony.

```
root@controllerCU:~# apt install chrony -y
```

2. Se configura *NTP* en el archivo `/etc/chrony/chrony.conf`.

   Nota: el archivo **chrony.conf** del nodo de control-CU se muestra en el apéndice B.1, mientras los archivos de configuración para los nodos computo-CU y red-CU se muestran en el apéndice B.2.

3. Se prueba la sincronización en el nodo de control-CU mediante el comando `chrony sources` (ver consola 4.4).

Consola 4.4: Prueba de sincronización del servidor NTP.

```
root@controllerCU:~# chronyc sources
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^- prod-ntp-4.ntp4.ps5.cano>     2  10   377  1103  -2870us[-2533us] +/-   67ms
^- alphyn.canonical.com          2  10   377   801  -2847us[-2475us] +/-   58ms
^- prod-ntp-5.ntp4.ps5.cano>     2  10   377   167  -3106us[-3106us] +/-   70ms
^- prod-ntp-3.ntp1.ps5.cano>     2  10   377   753  -2769us[-2391us] +/-   66ms
^- mail.broadbandcancun.mx       2  10   377   239  -5604us[-5604us] +/-  115ms
^- ntp2.flashdance.cx            2  10   377   496  -1132us[ -724us] +/-   83ms
^- Time100.Stupi.SE              1  10   377   627  -3937us[-3545us] +/-   88ms
^+ xalli.cie.unam.mx             1  10   377   746    -22us[ +356us] +/-   11ms
^* cronos.cenam.mx               1  10   363   327   +275us[ +702us] +/- 7870us
```

   Nota: el elemento de la columna `MS` marcado por un asterisco (*) indica el servidor con el que está sincronizado el servicio *NTP*.

4. Se verifica la sincronización *NTP* en los nodos de computo-CU y de red-CU mediante el comando `chronyc sources` (ver consola 4.5).

Consola 4.5: Prueba de sincronización del cliente NTP.

```
root@controllerCU:~#  chronyc sources
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* controller                    2   6   377     8  +6688ns[  +12us] +/- 8992us
```

Nota: este procedimiento se debe repetir en los nodos de la sucursal HOME.

4.1.3. OpenStack packages

Ubuntu 22.04 LTS tiene por defecto la versión `Yoga` de `OpenStack`. Sin embargo, esta versión no es la más reciente. Para habilitar el repositorio de la versión más reciente (`OpenStack 2023.1` Antílope, al momento de escribir esta tesis) se tiene que ejecutar el comando de la consola 4.6 en todos los nodos.

Consola 4.6: Comando para habilitar el archivo de OpenStack 2023.1 Antílope para Ubuntu 22.04 LTS.

```
root@controllerCU:~# add-apt-repository cloud-archive:antelope
```

Posteriormente, en cada nodo se actualiza la última versión del repositorio, así como también se actualiza cada paquete obsoleto y dependencia del sistema (consultar consola 4.7).

Consola 4.7: Comando para actualizar repositorio y dependencias del sistema.

```
root@controllerCU:~# apt update && apt upgrade -y
Hit:1 http://mx.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://mx.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://ubuntu-cloud.archive.canonical.com/ubuntu jammy-updates/\
   antelope InRelease
Hit:5 http://mx.archive.ubuntu.com/ubuntu jammy-backports InRelease
```

4.1.4. SQL database

La mayoría de los servicios de `OpenStack` utilizan una base de datos *SQL* para almacenar información [31]. En este proyecto se utiliza la base de datos `MariaDB` como gestor de base de datos.

A continuación se muestra el proceso para instalar y hacer la configuración inicial de la base de datos en el nodo de control-CU. Este procedimiento se debe repetir en el nodo de control-HOME:

1. Se instala `MariaDB` y las bibliotecas de Python para *MySQL* con el comando que se muestra en la consola 4.8.

   Consola 4.8: Instalación de MariaDB.

   ```
   root@controllerCU:~# apt install mariadb-server python3-pymysql -y
   ```

   Nota: `OpenStack` está escrito en Python por lo que son indispensables las bibliotecas de Python.

2. Utilizando el comando de la consola 4.9, se crea el archivo `99-openstack.cnf`.

   Consola 4.9: Comando para crear archivo /etc/mysql/mariadb.conf.d/99-openstack.cnf

   ```
   root@controllerCU:~# emacs /etc/mysql/mariadb.conf.d/99-openstack.cnf
   ```

   El contenido del archivo `99-openstack.cnf` del nodo de control-CU se muestra en la consola 4.10.

   Consola 4.10: Archivo /etc/mysql/mariadb.conf.d/99-openstack.cnf del nodo de control-CU.

   ```
   [mysqld]
   bind-address = 192.168.10.200

   default-storage-engine = innodb
   innodb_file_per_table = on
   max_connections = 4096
   collation-server = utf8_general_ci
   character-set-server = utf8
   ```

   El archivo `/etc/mysql/mariadb.conf.d/99-openstack.cnf` del nodo control-HOME es idéntico al de la consola 4.10, solo se debe modificar la *IP*.

3. Con el comando de la consola 4.11, se reinicia el servicio de base de datos.

Consola 4.11: Comando service mysql restart.

```
root@controllerCU:~# service mysql restart
```

4. Por último, se ejecuta el siguiente *script* que establece la contraseña del usuario *root* y el método de conexión, utilizando el comando mostrado en la consola 4.12.

Consola 4.12: Comando mysql secure installation.

```
root@controllerCU:~# mysql_secure_installation
```

Nota: La salida del comando `mysql_secure_installation` se muestra en el apéndice B.3.

4.1.5. Message queue

`OpenStack` utiliza un servicio de mensajes para coordinar operaciones e informar del estado entre servicios [32]. Para este proyecto se utiliza `RabbitMQ`, y a continuación se describe el proceso de instalación y configuración en el nodo de control-CU. Para el nodo de control-HOME, se repite el mismo proceso:

1. Se instala el paquete de `RabbitMQ` utilizando el comando que se muestra en la consola 4.13.

Consola 4.13: Comando para instalar RabbitMQ.

```
root@controllerCU:~# apt install rabbitmq-server -y
```

2. Se agrega el usuario `openstack` y se le asigna una contraseña, tal como se observa en la consola 4.14.

Consola 4.14: Comando -add user OpenStack- en Rabbitmq.

```
root@controllerCU:~# add_user openstack 316ac4
Adding user "openstack" ...
Done. Don-t forget to grant the user permissions to some virtual \
   hosts! See 'rabbitmqctl␣help␣set_permissions' to learn more.
```

3. Mediante el comando de la consola 4.15, se conceden al usuario `openstack` permisos de configuración, escritura y lectura.

Consola 4.15: Comando rabbitmqctl set permissions openstack.

```
root@controllerCU:~# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
Setting permissions for user "openstack" in vhost "/" ...
```

4.1.6. Memcached

El mecanismo de autenticación en `OpenStack` utiliza `Memcached` para crear los tokens [33].

A continuación se describe el proceso para instalar y configurar `Memcached` en el nodo de control-CU. Se repite el mismo proceso para el nodo de control-HOME.

1. Se instala `memcached` y las bibliotecas de Python para `memcached` con el comando que se muestra en la consola 4.16.

Consola 4.16: Comando para instalar memcached.

```
root@controllerCU:~# apt install memcached python3-memcache -y
```

2. Se configura el servicio en el archivo `/etc/memcached.conf`, como se muestra en la consola 4.17, para que utilice la *IP* del nodo de control, permitiendo así el acceso a otros nodos a través de la red.

Consola 4.17: Archivo /etc/memcached.conf.

```
-d
logfile /var/log/memcached.log
-m 64
-p 11211
-u memcache
-l 192.168.10.200
-P /var/run/memcached/memcached.pid
```

Nota: el archivo `memcached.conf` del nodo control-HOME es idéntico al de la consola 4.17, solo se modifica la *IP* del parámetro `-l`.

3. Con el comando de la consola 4.18, se reinicia el servicio `Memcached` para cargar la nueva configuración.

Consola 4.18: Comando para reinicia el servicio memcached.

```
root@controllerCU:~# service memcached restart
```

4.1.7. Etcd

`Etcd` es un almacén de claves y valores distribuidos de código abierto que permiten crear una configuración compartida [34].

A continuación se describe el proceso para instalar y configurar `Etcd` en el nodo control-CU. Se repite el mismo proceso para el nodo control-HOME:

1. Se instala el paquete de `Etcd` utilizando el comando que se muestra en la consola 4.19.

Consola 4.19: Comando para instalar Etcd.

```
root@controllerCU:~# apt install etcd -y
```

2. Se edita el archivo `/etc/default/etcd` mostrado en la consola 4.20. Se establecen los parámetros de la dirección *IP* del nodo de control con el fin de permitir el acceso de otros nodos a través de la red.

Consola 4.20: Archivo /etc/default/etcd.

```
ETCD_NAME="controller"
ETCD_DATA_DIR="/var/lib/etcd"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
ETCD_INITIAL_CLUSTER="controller=http://192.168.10.200:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://192.168.10.200:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://192.168.10.200:2379"
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
ETCD_LISTEN_CLIENT_URLS="http://192.168.10.200:2379"
```

3. Con el comando de la consola 4.21, se habilita y reinicia el servicio `Etcd` para cargar la nueva configuración.

Consola 4.21: Comandos para habilitar y reiniciar el servicio de etcd

```
root@controllerCU:~# systemctl enable etcd
root@controllerCU:~# systemctl restart etcd
```

4.2. Keystone: Identity service

El proceso para instalar y configurar `Keystone` en el nodo control-CU se repiten para el nodo control-HOME.

4.2.1. Base de datos para Keystone

Antes de instalar los paquetes de `Keystone` se debe de crear una base de datos:

1. Se crea un archivo *sql* llamado `a-keystoneDB.sql` utilizando el comando especificado en la consola 4.22.

Consola 4.22: Comando para crear archivo a-keystoneDB.sql.

```
root@controllerCU:~# emacs a-keystoneDB.sql
```

2. Dentro del archivo `a-keystoneDB.sql`, se escriben los comandos para crear la base de datos de `Keystone` y asignar al usuario `keystone` acceso completo a esa base de datos. Se puede visualizar el contenido completo del archivo `a-keystoneDB.sql` en la consola 4.23.

Consola 4.23: Archivo a-keystoneDB.sql.

```
1 CREATE DATABASE keystone;
2 GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
3 IDENTIFIED BY '316ac4';
4 GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
5 IDENTIFIED BY '316ac4';
```

3. Se ejecuta `a-keystoneDB.sql` con el comando de la consola 4.24.

Consola 4.24: Comando para ejecutar a-keystoneDB.sql.

```
root@controllerCU:~# cat a-keystoneDB.sql | mysql -p -u root
Enter password:
```

Nota: se debe de ingresar la contraseña de la base de datos para que el archivo se ejecute.

4.2.2. Instalación y configuración de Keystone

Para implementar el servicio de identidad se siguen los siguientes pasos:

1. Se instalan los paquetes de `Keystone` utilizando el comando que se muestra en la consola 4.25.

Consola 4.25: Comando para instalar los paquetes de Keystone

```
root@controllerCU:~# apt install keystone -y
```

2. Se edita el archivo `/etc/keystone/keystone.conf` para configurar el acceso a la base de datos y establecer el proveedor de token.

   Nota: el archivo `keystone.conf` (del nodo control-CU) se puede ver en el apéndice C.11

3. Se ejecuta el comando de la consola 4.26 para crear las tablas de la base de datos del servicio de identidad.

Consola 4.26: Comando para crear las tablas de la base de datos de Keystone.

```
root@controllerCU:~# su -s /bin/sh -c "keystone-manage db_sync" keystone
```

4. Se inicializan los repositorios de claves Fernet y se configuran las credenciales para agregar reglas y permisos utilizando el comando de la consola 4.27.

Consola 4.27: Comando para inicializar los repositorios de claves Fernet y configurar las credenciales.

```
root@controllerCU:~# keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
root@controllerCU:~# keystone-manage credential_setup --keystone-user keystone --keystone-group \
    keystone
```

5. Se ejecuta el comando de la consola 4.28 para iniciar el servicio de identidad y sus *endpoints* (URL que permiten conectar con otros servicios de `OpenStack`), permitiendo así que los demás servicios se puedan conectar a `Keystone`.

Consola 4.28: Comando para iniciar Keystone y sus endpoints.

```
root@controllerCU:~# keystone-manage bootstrap --bootstrap-password 316ac4 \
  --bootstrap-admin-url http://192.168.10.200:5000/v3/ \
  --bootstrap-internal-url http://192.168.10.200:5000/v3/ \
  --bootstrap-public-url http://192.168.10.200:5000/v3/ \
  --bootstrap-region-id RegionOne
```

Nota: `OpenStack` utiliza tres tipos de *endpoints* (*admin*, *internal* y *public*) que permiten acceder a un servicio con permisos diferentes.

4.2.3. Scripts de entorno para los clientes de OpenStack

El servicio de identidad proporciona servicios de autenticación para cada servicio `OpenStack`. El servicio de autenticación utiliza una combinación de proyectos, usuarios y roles [53]. Para ello, es necesario crear el nombre de usuario, la contraseña y la URL de autenticación para la cuenta *admin* como variables de entorno. El proceso para crear estas variables de entorno y cargar sus valores se puede ver en el apéndice C.1.1.

Antes de continuar, es necesario crear el proyecto service mediante el comando de la consola 4.29.

Consola 4.29: Comando para crear el proyecto service.

```
root@controllerCU:~# openstack project create --domain default \
  --description "Service Project" service
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Service Project                  |
| domain_id   | default                          |
| enabled     | True                             |
| id          | 9e3ac89092bd458da5d8afb7a49dce5c |
| is_domain   | False                            |
| name        | service                          |
| options     | {}                               |
| parent_id   | default                          |
| tags        | []                               |
+-------------+----------------------------------+
```

4.3. Glance: Image service

El servicio de imágenes (`Glance`) permite a los usuarios registrar y recuperar imágenes de máquinas virtuales [50]. El proceso de instalación y configuración de `Glance` en el nodo de control-CU debe replicarse en el nodo de control-HOME.

4.3.1. Requisitos previos del servicio de imágenes

Antes de instalar los paquetes de `Glance` se debe de crear una base de datos, credenciales de servicio y *API endpoints*.

4.3.1.1. Base de datos para Glance

1. Se crea un archivo sql llamado `b-glanceDB.sql` utilizando el comando especificado en la consola 4.30.

Consola 4.30: Comando para crear el archivo b-glanceDB.sql.

```
root@controllerCU:~# emacs b-glanceDB.sql
```

2. Dentro del archivo `b-glanceDB.sql`, se escriben los comandos para crear la base de datos de `Glance` y asignar al usuario `glance` acceso completo a esa base de datos. Se puede visualizar el contenido completo del archivo `b-glanceDB.sql` en la consola 4.31.

Consola 4.31: Archivo b-glanceDB.sql.

```
1 CREATE DATABASE glance;
2 GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
3   IDENTIFIED BY '316ac4';
4 GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
5   IDENTIFIED BY '316ac4';
```

3. Se ejecuta `b-glanceDB.sql` con el comando de la consola 4.32.

Consola 4.32: Comando para ejecutar b-glanceDB.sql.

```
root@controllerCU:~# cat b-glanceDB.sql | mysql -u root -p
Enter password:
```

Nota: se debe de ingresar la contraseña de la base de datos para que el archivo se pueda ejecutar.

4.3.1.2. Credenciales de servicio para Glance

Para crear las credenciales del servicio es necesario seguir los siguientes pasos:

1. Se adquieren las variables del cliente `admin`. Ver comando y requisitos en C.3.
2. Con el comando de la consola 4.33, se crea el usuario `glance` y se le asigna una contraseña.

Consola 4.33: Comando para crear el usuario glance.

```
root@controllerCU:~# openstack user create --domain default --password-prompt glance
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 8ab3cb82541540408e81bfc86ec8f330 |
| name                | nova                             |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

3. Se agrega al usuario `glance` la función de administrador del proyecto `service` utilizando el comando de la consola 4.34.

Consola 4.34: Comando para asignarle a glance el rol de administrador del proyecto service.

```
root@controllerCU:~# openstack role add --project service --user glance admin
```

4. Se crea la entidad del servicio de imágenes con el comando de la consola 4.35.

Consola 4.35: Comando para crear el servicio de cómputo.

```
root@controllerCU:~#  openstack service create --name glance \
  --description "OpenStack Image" image
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Image                  |
| enabled     | True                             |
| id          | 3adc160b0b4e4c0bb3ba2906572f7d78 |
| name        | glance                           |
| type        | image                            |
+-------------+----------------------------------+
```

4.3.1.3. API endpoints del servicio de imágenes

Los *endpoints* en OpenStack son URL que permiten conectar a los diferentes servicios como se mencionó anteriormente. Para crear los *API endpoints* público, interno y de administración en la RegionOne se deben ejecutar los comandos de la consola 4.36.

Consola 4.36: Comandos para la creación de los API endpoints del servicio de imágenes.

```
root@controllerCU:~# openstack endpoint create --region RegionOne \
  image public http://192.168.10.200:9292
openstack endpoint create --region RegionOne \
  image internal http://192.168.10.200:9292
openstack endpoint create --region RegionOne \
  image admin http://192.168.10.200:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | b1f61cabe41c4a03980613d00bfaca00 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 4e557d823a7b4694a8b5b2e8353ea377 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://192.168.10.200:9292       |
+--------------+----------------------------------+
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | ff550e7df55142c3bf89beb17158184f |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 4e557d823a7b4694a8b5b2e8353ea377 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://192.168.10.200:9292       |
+--------------+----------------------------------+
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 4b504018eaf04ff9a471ba7f5c25036e |
| interface    | admin                            |
```

```
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 4e557d823a7b4694a8b5b2e8353ea377 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://192.168.10.200:9292       |
+--------------+----------------------------------+
```

4.3.2. Instalación y configuración de Glance

Para implementar el servicio de imágenes se siguen los siguientes pasos:

1. Se instalan los paquetes de `Glance` utilizando el comando que se muestra en la consola 4.37.

   Consola 4.37: Comando para instalar los paquetes de Glance en el nodo de control.

   ```
   root@controllerCU:~# apt install glance -y
   ```

2. Se edita el archivo `/etc/glance/glance-api.conf` para:

   a) Configurar el acceso a la base de datos y la comunicación con `Keystone`.

   b) Indicar el lugar donde se almacena el sistema de archivos local y la ubicación de los archivos de imagen.

   Nota: el archivo `glance-api.conf` se muestra en el apéndice C.12

3. Se ejecuta el comando de la consola 4.38 para crear las tablas de la base de datos del servicio de imágenes.

   Consola 4.38: Comando para crear las tablas de la base de datos de Glance.

   ```
   root@controllerCU:~# glance-manage db_sync
   ```

4. Se reinicia el servicio de `Glance` utilizando el comando de la consola 4.39.

   Consola 4.39: Comando para reinciar el servicio de Glance.

   ```
   root@controllerCU:~# service glance-api restart
   ```

4.3.3. Establecimiento del repositorio de imágenes

En este proyecto de tesis se emplearon las imágenes de Ubuntu, Debian, Fedora y Windows, ya que son sistemas operativos comúnmente utilizados en entornos domésticos y educativos.

A continuación se detalla el proceso para guardar imágenes en el nodo de control:

1. Se adquieren las variables de entorno del usuario `admin`. Ver comando y requisitos en C.3.

2. Se descarga la imagen fuente de los sistemas operativos:

   a) Se ejecuta el comando de la consola 4.40 para descargar la imagen de Debian 12.

   Consola 4.40: Comando para descargar la imagen de Debian 12.

   ```
   root@controllerCU:~# wget https://cloud.debian.org/images/cloud/\
      bookworm/latest/debian-12-generic-amd64.qcow2
   ```

   b) Se ejecuta el comando de la consola 4.41 para descargar la imagen de Ubuntu 22.

Consola 4.41: Comando para descargar la imagen de Ubuntu Server 22.

```
root@controllerCU:~# wget https://cloud-images.ubuntu.com/jammy\
   /20240119.1/jammy-server-cloudimg-amd64-disk-kvm.img
```

c) Se ejecuta el comando de la consola 4.42 para descargar la imagen de Fedora 39.

Consola 4.42: Comando para descargar la imagen de Fedora Cloud 39.

```
root@controllerCU:~# wget https://download.fedoraproject.org/pub/\
   fedora/linux/releases/39/Cloud/x86_64/images/Fedora-Cloud-Base\
   -39-1.5.x86_64.qcow2
```

d) Para descargar Windows Server 2012, se sigue el proceso de la url `https://cloudbase.it/windows-cloud-images/`.

3. Se utiliza el comando de la consola 4.43 para cargar las imágenes al servicio de imágenes (`Glance`) utilizando el formato `QCOW2`. Este formato contiene una copia del sistema operativo, es decir, se puede ver como una plantilla para que con esta imagen se puedan crear múltiples instancias del sistema operativo.

Consola 4.43: Comando para cargar a Glance la imagen de Debian 12.

```
root@controllerCU:~# glance image-create --name "debian-12" \
  --file debian-12-generic-amd64.qcow2 \
  --disk-format qcow2 --container-format bare \
  --visibility=public
+------------------+--------------------------------------------------------------------------------------+
| Property         | Value                                                                                |
+------------------+--------------------------------------------------------------------------------------+
| checksum         | 23494e503eb40173ae67c45ca9256490                                                     |
| container_format | bare                                                                                 |
| created_at       | 2024-01-30T20:58:42Z                                                                 |
| disk_format      | qcow2                                                                                |
| id               | 4f1f58ac-4dfe-476e-a3ef-13b3bc877279                                                 |
| min_disk         | 0                                                                                    |
| min_ram          | 0                                                                                    |
| name             | debian-12                                                                            |
| os_hash_algo     | sha512                                                                               |
| os_hash_value    | ad93b29870f370b34b8bbd138b92608ec7a1817ee64d3b4602143dd68713e5e225a7fec0509e98e6     |
|                  | 4ed5d570c27f3a6864e7ab4aa99d7cdfad2158f4b84d364f                                     |
| os_hidden        | False                                                                                |
| owner            | 7285197c3e4e4709b46709a49a6f9d17                                                     |
| protected        | False                                                                                |
| size             | 379663872                                                                            |
| status           | active                                                                               |
| tags             | []                                                                                   |
| updated_at       | 2024-01-30T20:58:47Z                                                                 |
| virtual_size     | 2147483648                                                                           |
| visibility       | public                                                                               |
+------------------+--------------------------------------------------------------------------------------+
```

Para cargar las imágenes restantes se utiliza el mismo comando de la consola 4.43. Solo se modifica el nombre y la ruta del disco correspondiente a cada sistema operativo.

4. Se puede validar el contenido del repositorio mediante el comando de la consola 4.44.

Consola 4.44: Comando para validar el repositorio de Glance.

```
root@controllerCU:~#glance image-list
+--------------------------------------+------------+
| ID                                   | Name       |
+--------------------------------------+------------+
| 36e82b3e-e19d-4940-95f7-35cc6846714a | cirros     |
| 5cd75433-1783-469e-9e74-2a71d470c520 | debian-10  |
| 1bdae7ca-8756-4b7e-a743-ac56b4880b82 | debian12   |
| 59c3e554-3679-4e05-a3c5-4eb0b165fc81 | Windows    |
| 26af1409-9950-4960-b66a-89be84868951 | Windows 10 |
+--------------------------------------+------------+
```

4.4. Placement: Placement service

El servicio `Placement` proporciona una *API* HTTP que permite ver los inventarios y recursos existentes en la nube [49], es decir, este servicio le permite a otros servicios ver si

existen recursos como memoria RAM, espacio de almacenamiento, recursos de red, o *IP* disponibles. El proceso de instalación y configuración de `Placement` en el nodo de control-CU debe replicarse en el nodo de control-HOME.

4.4.1. Requisitos previos del servicio Placement

Antes de instalar los paquetes de `Placement` se debe de crear la base de datos, y las credenciales del servicio, así como los *API endpoints*.

4.4.1.1. Base de datos para Placement

1. Se crea un archivo sql llamado `c-placementDB.sql` utilizando el comando especificado en la consola 4.45.

Consola 4.45: Comando para crear archivo c-placementDB.sql.

```
root@controllerCU:~# emacs c-placementDB.sql
```

2. Dentro del archivo `c-placementDB.sql`, se escriben los comandos para crear la base de datos de `Placement` y asignar al usuario `placement` acceso completo a esa base de datos. Se puede visualizar el contenido completo del archivo `c-placementDB.sql` en la consola 4.46.

Consola 4.46: Archivo c-placementDB.sql.

```
1 CREATE DATABASE placement;
2 GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'localhost' \
3   IDENTIFIED BY '316ac4';
4 GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'%' \
5   IDENTIFIED BY '316ac4';
```

3. Se ejecuta `c-placementDB.sql` con el comando de la consola 4.47.

Consola 4.47: Comando para ejecutar c-placementDB.sql.

```
root@controllerCU:~# cat c-placementDB.sql | mysql -u root -p
Enter password:
```

Nota: se debe de ingresar la contraseña de la base de datos para que el archivo se ejecute.

4.4.1.2. Credenciales de servicio para Placement

Para crear las credenciales del servicio se realizan los siguientes pasos:

1. Se adquieren las variables del usuario `admin`. Ver comando y requisitos en C.3.
2. Con el comando de la consola 4.48, se crea el usuario `placement` y se le asigna una contraseña.

Consola 4.48: Comando para crear el usuario placement.

```
root@controllerCU:~# openstack user create --domain default --password-prompt placement
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | c8e68deeef084891978361538ca43bb3 |
| name                | placement                        |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

3. Se agrega al usuario `placement` la función de administrador del proyecto `service` utilizando el comando de la consola 4.49.

Consola 4.49: Comando para asignarle a placement el rol de administrador del proyecto service.

```
root@controllerCU:~# openstack role add --project service --user placement admin
```

4. Se crea la entidad `Placement` con el comando de la consola 4.50.

Consola 4.50: Comando para crear el servicio de ubicación.

```
root@controllerCU:~# openstack service create --name placement \
--description "Placement API" placement
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Placement API                    |
| enabled     | True                             |
| id          | 90b4b5c9cf1f4c66b4a2f8f24fd5bbf8 |
| name        | placement                        |
| type        | placement                        |
+-------------+----------------------------------+
```

4.4.1.3. API endpoints de Placement

Para crear los *API endpoints* público, interno y de administración en la `RegionOne` se deben ejecutar los comandos de la consola 4.51.

Consola 4.51: Comandos para la creacion de los API endpoints de Placement.

```
root@controllerCU:~# openstack endpoint create --region RegionOne \
placement public http://192.168.10.200:8778
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | f9706044c0474229ab87a491f351de16 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f2df0dc25e93471090f12d8e04c20592 |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://192.168.10.200:8778       |
+--------------+----------------------------------+
root@controllerCU:~# openstack endpoint create --region RegionOne \
placement internal http://192.168.10.200:8778
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 2c201317cc9d417cb6305e9782f36073 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f2df0dc25e93471090f12d8e04c20592 |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://192.168.10.200:8778       |
+--------------+----------------------------------+
root@controllerCU:~# openstack endpoint create --region RegionOne \
placement admin http://192.168.10.200:8778
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 9e060f91edc44a4f90af67933974a652 |
```

```
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f2df0dc25e93471090f12d8e04c20592 |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://192.168.10.200:8778       |
+--------------+----------------------------------+
```

4.4.2. Instalación y configuración de los componentes de Placement

1. Se instalan los paquetes de `Placement` utilizando el comando que se muestra en la consola 4.52.

Consola 4.52: Comando para instalar los paquetes de Placement en el nodo de control.

```
root@controllerCU:~# apt install placement-api -y
```

2. Se edita el archivo `/etc/placement/placement.conf` para configurar el acceso a la base de datos y a `Keystone`. Nota: El archivo `placement.conf` se muestra en C.13

3. Se ejecuta el comando de la consola 4.53 para crear las tablas de la base de datos de `Placement`.

Consola 4.53: Comando para crear las tablas de la base de datos de Placement.

```
root@controllerCU:~# placement-manage db sync
```

4. Se verifica el estado de `Placement` mediante el comando de la consola 4.54.

Consola 4.54: Comando para verificar el estado de Placement.

```
root@controllerCU:~# placement-status upgrade check
+----------------------------------------+
| Upgrade Check Results                  |
+----------------------------------------+
| Check: Missing Root Provider IDs       |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
| Check: Incomplete Consumers            |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
| Check: Policy File JSON to YAML Migration |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
```

5. Se instala el complemento `osc-placement`, utilizando el comando que se muestra en la consola 4.55.

Consola 4.55: Comando para instalar el complemento osc-placement.

```
root@controllerCU:~# apt install python3-pip -y
root@controllerCU:~# pip3 install osc-placement
```

6. Los recursos disponibles de `Placement` se pueden ver con el comando de la consola 4.56.

Consola 4.56: Comando para ver recursos disponibles de Placement.

```
root@controllerCU:~#  openstack --os-placement-api-version 1.2 \
    resource class list --sort-column name
+----------------------------------------+
| name                                   |
+----------------------------------------+
| DISK_GB                                |
| FPGA                                   |
| IPV4_ADDRESS                           |
| MEMORY_MB                              |
| MEM_ENCRYPTION_CONTEXT                 |
| NET_BW_EGR_KILOBIT_PER_SEC             |
| NET_BW_IGR_KILOBIT_PER_SEC             |
| NET_PACKET_RATE_EGR_KILOPACKET_PER_SEC |
| NET_PACKET_RATE_IGR_KILOPACKET_PER_SEC |
| NET_PACKET_RATE_KILOPACKET_PER_SEC     |
| NUMA_CORE                              |
| NUMA_MEMORY_MB                         |
| NUMA_SOCKET                            |
| NUMA_THREAD                            |
| PCI_DEVICE                             |
| PCPU                                   |
| PGPU                                   |
| SRIOV_NET_VF                           |
| VCPU                                   |
| VGPU                                   |
| VGPU_DISPLAY_HEAD                      |
+----------------------------------------+
```

Nota: la salida completa del comando de la consola 4.56 se muestran en el apéndice C.19.

4.5. Nova: Compute service

El proceso de instalación y configuración de `Nova` en el nodo de control-CU debe replicarse en el nodo de control-HOME.

4.5.1. Requisitos previos del servicio de cómputo

Antes de instalar los paquetes de `Nova` se debe de crear la base de datos, y las credenciales del servicio, así como los *API endpoints*.

4.5.1.1. Base de datos para Nova

1. Se crea un archivo sql llamado `d-computeDB.sql` utilizando el comando especificado en la consola 4.57.

Consola 4.57: Comando para crear el archivo d-computeDB.sql.

```
root@controllerCU:~# emacs d-computeDB.sql
```

2. Dentro del archivo `d-computeDB.sql`, se escriben los comandos para crear 3 bases de datos para `Nova` y asignar al usuario `nova` acceso completo a esas bases de datos. Se puede visualizar el contenido completo del archivo `d-computeDB.sql` en la consola 4.58.

Consola 4.58: Archivo d-computeDB.sql

```
CREATE DATABASE nova_api;
CREATE DATABASE nova;
CREATE DATABASE nova_cell0;
GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
IDENTIFIED BY '316ac4';
GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
IDENTIFIED BY '316ac4';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
IDENTIFIED BY '316ac4';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
IDENTIFIED BY '316ac4';
GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' \
IDENTIFIED BY '316ac4';
GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%' \
IDENTIFIED BY '316ac4';
```

3. Se ejecuta `d-computeDB.sql` con el comando de la consola 4.59.

Consola 4.59: Comando para ejecutar d-computeDB.sql.

```
root@controllerCU:~# cat d-computeDB.sql | mysql -u root -p
Enter password:
```

Nota: se debe de ingresar la contraseña de la base de datos para que el archivo se ejecute.

4.5.1.2. Credenciales del servicio Nova

Los siguientes pasos crean las credenciales del servicio:

1. Se requieren las variables de entorno del usuario `admin`. Ver comando y requisitos en el apéndice C.3.

2. Con el comando de la consola 4.60, se crea el usuario `nova` y se le asigna una contraseña.

Consola 4.60: Comando para crear el usuario nova.

```
root@controllerCU:~# openstack user create --domain default --password-prompt nova
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 8ab3cb82541540408e81bfc86ec8f330 |
| name                | nova                             |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

3. Se agrega al usuario `nova` la función de administrador del proyecto `service` utilizando el comando de la consola 4.61.

Consola 4.61: Comando para asignarle a nova el rol de administrador del proyecto service.

```
root@controllerCU:~# openstack role add --project service --user nova admin
```

4. Se crea la entidad del servicio de cómputo con el comando de la consola 4.62.

Consola 4.62: Comando para crear servicio de cómputo.

```
root@controllerCU:~# openstack service create --name nova \
--description "OpenStack Compute" compute
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Compute                |
| enabled     | True                             |
| id          | 5e8fc0238ab64251b87142c7b6cd48d4 |
| name        | nova                             |
| type        | compute                          |
+-------------+----------------------------------+
```

4.5.1.3. API endpoints del servicio de cómputo

Para crear los *API endpoints* público, interno y de administración en la `RegionOne` se deben ejecutar los comandos de la consola 4.63.

Consola 4.63: Comandos para la creacion de los API endpoints del servicio de cómputo.

```
root@controllerCU:~# openstack endpoint create --region RegionOne \
compute public http://192.168.10.200:8774/v2.1
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 07c8feb73e4e4643afd2bf044afc251a |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 5e8fc0238ab64251b87142c7b6cd48d4 |
| service_name | nova                             |
| service_type | compute                          |
| url          | http://192.168.10.200:8774/v2.1  |
+--------------+----------------------------------+
root@controllerCU:~# openstack endpoint create --region RegionOne \
compute internal http://192.168.10.200:8774/v2.1
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | f9de16260dcc47b5be56213e55821776 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 5e8fc0238ab64251b87142c7b6cd48d4 |
| service_name | nova                             |
| service_type | compute                          |
| url          | http://192.168.10.200:8774/v2.1  |
+--------------+----------------------------------+
root@controllerCU:~# openstack endpoint create --region RegionOne \
compute admin http://192.168.10.200:8774/v2.1
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | fb47d511a86e4735955fb6d2d44e3ed0 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 5e8fc0238ab64251b87142c7b6cd48d4 |
| service_name | nova                             |
| service_type | compute                          |
| url          | http://192.168.10.200:8774/v2.1  |
+--------------+----------------------------------+
```

4.5.1.4. Instalación y configuración de Nova

1. Se instalan los paquetes de Nova, en el nodo de control, utilizando el comando que se muestra en la consola 4.64.

Consola 4.64: Comando para intalar los paquetes de Nova en el nodo de control.

```
root@controllerCU:~# apt install nova-api nova-conductor nova-\
    spiceproxy nova-scheduler -y
```

2. Se edita el archivo /etc/nova/nova.conf para:

   a) Indicarle la *IP* del nodo de control al servicio de cómputo.

   b) Configurar el acceso a la base de datos, el acceso a Keystone, acceso al servicio RabbitMQ, acceso al servicio Glance, acceso al servicio Placement y al servicio Neutron.

   c) Habilitar la consola remota por medio de SPICE sobre SSL/TLS.

   Nota: el archivo nova.conf del nodo de control se muestra en el apéndice C.16

3. Se ejecuta el comando de la consola 4.65 para crear las tablas de la base de datos nova_api.

Consola 4.65: Comando para crear las tablas de la base de datos de nova_api.

```
root@controllerCU:~# nova-manage api_db sync
```

4. Se registra la base de datos cell0 con el comando mostrado en la consola 4.66.

Consola 4.66: Comando para registra la base de datos cell0.

```
root@controllerCU:~# nova-manage cell_v2 map_cell0
```

5. Se crea la celda cell1 con el comando de la consola 4.67.

Consola 4.67: Comando para crear la celda cell1.

```
root@controllerCU:~# nova-manage cell_v2 create_cell --name=cell1 --\
    verbose
```

6. Se ejecuta el comando de la consola 4.68 para crear las tablas de la base de datos Nova.

Consola 4.68: Comando para crear las tablas de la base de datos Nova.

```
root@controllerCU:~# nova-manage db sync
```

Nota: La salida del comando de la consola 4.68 se muestra en el apéndice C.20.

7. Como de muestra en la consola 4.69, se enumeran las celdas para verificar que cell0 y cell1 contengan la información correcta sobre la base de datos y RabbitMQ.

Consola 4.69: Comando para listar celdas.

```
root@controllerCU:~# nova-manage cell_v2 list_cells
+-------+--------------------------------------+----------------------------------------------------+-------------------------------------+----------+
|  Name |                 UUID                 |                   Transport URL                    |                Database Connection \
                    | Disabled |
+-------+--------------------------------------+----------------------------------------------------+-------------------------------------+----------+
| cell0 | 00000000-0000-0000-0000-000000000000 |                       none:/                       | mysql+pymysql://nova:****@192.168.10.200/\
      nova_cell0 |  False   |
| cell1 | 1399a0fa-5abd-4f19-9b26-4f2f795cc8b9 | rabbit://openstack:****@192.168.10.200:5672/ |    mysql+pymysql://nova:****@192.168.10.200/\
      nova    |  False   |
+-------+--------------------------------------+----------------------------------------------------+-------------------------------------+----------+
```

8. Se reinicia los siguientes servicios de Nova utilizando los comando de la consola 4.70.

Consola 4.70: Comandos para reiniciar servicios de Nova en el nodo de control.

```
root@controllerCU:~$ sudo systemctl restart nova-api
sudo systemctl restart nova-scheduler
sudo systemctl restart nova-conductor
sudo systemctl restart nova-spiceproxy.service
sudo systemctl stop nova-novncproxy
sudo systemctl enable nova-api
sudo systemctl enable nova-scheduler
sudo systemctl enable nova-conductor
sudo systemctl enable nova-spiceproxy.service
sudo systemctl disable nova-novncproxy
```

9. Los siguientes pasos se realizan en el nodo de cómputo.

4.5.2. Instalación y configuración en el nodo de cómputo

El proceso de instalación y configuración de Nova en el nodo de cómputo-CU debe replicarse en el nodo de cómputol-HOME.

1. Se instalan los paquetes de Nova, en el nodo de cómputo, utilizando el comando que se muestra en la consola 4.71.

Consola 4.71: Comando para instalar paquetes de Nova en el nodo de cómputo.

```
root@computeCU:~# apt install nova-compute -y
```

2. Al igual que en el nodo de control, se edita el archivo /etc/nova/nova.conf para:
   a) Indicarle la *IP* del nodo de cómputo a Nova.
   b) Configurar el acceso a Keystone, a RabbitMQ, a Glance, a Placement y a Neutron.
   c) Habilitar la consola remota por medio de SPICE sobre SSL/TLS.

   Nota: el archivo nova.conf del nodo de cómputo se muestra en el apéndice C.22.

3. Se reinicia el servicio de Nova utilizando el comando de la consola 4.72.

Consola 4.72: Comando para reiniciar el servicio de Nova en el nodo de cómputo.

```
root@computeCU:~# service nova-compute restart
```

4. Se regresa al nodo de control para finalizar la instalación.

4.5.3. Anexión del nodo de computo a la base de datos

Una vez dentro del nodo de control, se realiza lo siguiente:

1. Se adquieren las variables del usuario admin. Ver comando y requisitos en el apéndice C.3.

2. Con el propósito de verificar si el nodo de cómputo configurado previamente está operativo, se listan los nodos de cómputo disponibles utilizando el comando que se muestra en la consola 4.73.

Consola 4.73: Comando para enumeran los nodos de cómputo disponibles.

```
root@controllerCU:~# openstack compute service list --service nova-compute
+--------------------------------------+--------------+--------------+------+---------+-------+----------------------------+
| ID                                   | Binary       | Host         | Zone | Status  | State | Updated At                 |
+--------------------------------------+--------------+--------------+------+---------+-------+----------------------------+
| bb049c2f-8cd0-40be-957f-833197be0cfd | nova-compute |    computeCU | nova | enabled | up    | 2023-07-31T22:36:31.000000 |
+--------------------------------------+--------------+--------------+------+---------+-------+----------------------------+
```

3. Se agrega el nodo de cómputo a la celda mediante el comando de la consola 4.74.

Consola 4.74: Comando para agrega el nodo de cómputo a la celda.

```
root@controllerCU:~# nova-manage cell_v2 discover_hosts --verbose
Found 2 cell mappings.
Skipping cell0 since it does not contain hosts.
Getting computes from cell 'cell1': 1399a0fa-5abd-4f19-9b26-4f2f795cc8b9
Checking host mapping for compute host 'computeCU': 7ccdc919-0547-41ce-870c-\
    be41844577dd
Creating host mapping for compute host 'computeCU': 7ccdc919-0547-41ce-870c-\
    be41844577dd
Found 1 unmapped computes in cell: 1399a0fa-5abd-4f19-9b26-4f2f795cc8b9
```

4.5.4. Verificación del funcionamiento del servicio de cómputo

Para verificar el estado de Nova se requiere de los comandos de administración de OpenStack, por lo tanto, se debe estar en el nodo de control y ejecutar los siguientes comandos:

1. Se listan los componentes del servicio de cómputo para verificar que estén habilitados, utilizando el comando que se muestra en la consola 4.75.

Consola 4.75: Comando para enumeran los componentes del servicio de cómputo.

```
root@controllerCU:~# openstack compute service list
+--------------------------------------+----------------+-------------------+----------+---------+-------+----------------------------+
| ID                                   | Binary         | Host              | Zone     | Status  | State | Updated At                 |
+--------------------------------------+----------------+-------------------+----------+---------+-------+----------------------------+
| 14991043-afd4-401b-a794-f480f5b34d1b | nova-scheduler | ubuntu-controller | internal | enabled | up    | 2023-07-31T22:40:09.000000 |
| 339cd351-2a06-4b7d-873e-8f8e7164d0ef | nova-conductor | ubuntu-controller | internal | enabled | up    | 2023-07-31T22:40:05.000000 |
| bb049c2f-8cd0-40be-957f-833197be0cfd | nova-compute   | ubuntu-compute    | nova     | enabled | up    | 2023-07-31T22:40:11.000000 |
+--------------------------------------+----------------+-------------------+----------+---------+-------+----------------------------+
```

2. Utilizando el comando que se muestra en la consola 4.76, se enumeran los puntos *API endpoints* con el objetivo de comprobar la presencia de Nova:

Consola 4.76: Comando para listar los puntos API endpoints.

```
root@controllerCU:~# openstack catalog list
+-----------+-----------+-----------------------------------------+
| Name      | Type      | Endpoints                               |
+-----------+-----------+-----------------------------------------+
| keystone  | identity  | RegionOne                               |
|           |           |   admin: http://192.168.10.200:5000/v3/ |
|           |           | RegionOne                               |
|           |           |   public: http://192.168.10.200:5000/v3/|
|           |           | RegionOne                               |
|           |           |   internal: http://192.168.10.200:5000/v3/|
|           |           |                                         |
| glance    | image     | RegionOne                               |
|           |           |   admin: http://192.168.10.200:9292     |
|           |           | RegionOne                               |
|           |           |   public: http://192.168.10.200:9292    |
|           |           | RegionOne                               |
|           |           |   internal: http://192.168.10.200:9292  |
|           |           |                                         |
| nova      | compute   | RegionOne                               |
|           |           |   public: http://192.168.10.200:8774/v2.1|
|           |           | RegionOne                               |
|           |           |   internal: http://192.168.10.200:8774/v2.1|
|           |           | RegionOne                               |
|           |           |   admin: http://192.168.10.200:8774/v2.1|
|           |           |                                         |
| placement | placement | RegionOne                               |
|           |           |   internal: http://192.168.10.200:8778  |
```

```
|           |           | RegionOne                                    |
|           |           |   admin: http://192.168.10.200:8778          |
|           |           | RegionOne                                    |
|           |           |   public: http://192.168.10.200:8778         |
|           |           |                                              |
+-----------+-----------+----------------------------------------------+
```

4.6. Neutron: Networking service

El proceso de instalación y configuración de `Neutron` en el nodo de control-CU debe replicarse en el nodo de control-HOME.

4.6.1. Requisitos previos del servicio de redes

Antes de instalar los paquetes de `Neutron` se debe crear la base de datos, las credenciales de servicio, así como los *API endpoints*.

4.6.1.1. Base de datos para Neutron

1. Se crea un archivo *sql* llamado `e-networkingDB.sql` utilizando el comando especificado en la consola 4.77.

Consola 4.77: Comando para crear el archivo e-networkingDB.sql.

```
root@controllerCU:~# emacs e-networkingDB.sql
```

2. Dentro del archivo `e-networkingDB.sql`, se escriben los comandos para crear la base de datos para `Neutron` y asignar al usuario `neutron` acceso completo a esa base de datos. Se puede visualizar el contenido completo del archivo `e-networkingDB.sql` en la consola 4.78.

Consola 4.78: Archivo e-networkingDB.sql.

```
1 CREATE DATABASE neutron;
2 GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
3 IDENTIFIED BY '316ac4';
4 GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
5 IDENTIFIED BY '316ac4';
```

3. Se ejecuta `e-networkingDB.sql` con el comando de la consola 4.79.

Consola 4.79: Comando para ejecutar e-networkingDB.sql.

```
root@controllerCU:~# cat e-networkingDB.sql | mysql -u root -p
Enter password:
```

Nota: se debe de ingresar la contraseña de la base de datos para que el archivo se ejecute.

4.6.1.2. Credenciales de servicio para Neutron

1. Se adquieren las variables del usuario `admin`. Ver comando y requisitos en el apéndice C.3.

2. Con el comando de la consola 4.80, se crea el usuario `neutron` y se le asigna una contraseña.

Consola 4.80: Comando para crear usuario neutron.

```
root@controllerCU:~# openstack user create --domain default --password-prompt neutron
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 220fb19307834af49aea1888db08bbfe |
| name                | neutron                          |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

3. Se agrega al usuario `neutron` la función de administrador del proyecto `service` utilizando el comando de la consola 4.81.

Consola 4.81: Comando para asignarle a neutron el rol de administrador del proyecto service.

```
root@controllerCU:~# openstack role add --project service --user \
    neutron admin
```

4. Se crea la entidad del servicio de red con el comando de la consola 4.82.

Consola 4.82: Comando para crear servicio de red.

```
root@controllerCU:~# openstack service create --name neutron \
--description "OpenStack Networking" network
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Networking             |
| enabled     | True                             |
| id          | 6b6bc9deefc14b3e9191232003d8372c |
| name        | neutron                          |
| type        | network                          |
+-------------+----------------------------------+
```

4.6.1.3. API endpoints del servicio de red

Para crear los *API endpoints* público, interno y de administración en la `RegionOne` se deben ejecutar los comandos de la consola 4.83.

Consola 4.83: Comandos para la creacion de los API endpoints del servicio de red.

```
root@controllerCU:~# openstack endpoint create --region RegionOne \
network public http://192.168.10.200:9696
openstack endpoint create --region RegionOne \
network internal http://192.168.10.200:9696
openstack endpoint create --region RegionOne \
network admin http://192.168.10.200:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 470eda74d9ae485dadfb209197224fd3 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | ed85cdb90a104d51856dc0297625edad |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://192.168.10.200:9696       |
+--------------+----------------------------------+
+--------------+----------------------------------+
```

```
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 421948b238c14ea0aeacfca944b27f54 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | ed85cdb90a104d51856dc0297625edad |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://192.168.10.200:9696       |
+--------------+----------------------------------+
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 081abb8fa9d4488c92aecfb948913aea |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | ed85cdb90a104d51856dc0297625edad |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://192.168.10.200:9696       |
+--------------+----------------------------------+
```

4.6.1.4. Instalación y configuración de Neutron

1. Se instalan los paquetes de `Neutron`, en el nodo de control, utilizando el comando que se muestra en la consola 4.84.

   Consola 4.84: Comando para instalar los paquetes de Neutron en el nodo de control.

   ```
   root@controllerCU:~$ apt install neutron-server \
   neutron-plugin-ml2 -y
   ```

   donde,

   - `neutron-server`: encamina las solicitudes de la *API* al agente `Networking plug-in`.
   - `neutron-plugin-ml2`: permite que `Open vSwitch` pueda crear infraestructura de red.

2. Se edita el archivo `/etc/neutron/neutron.conf` para habilitar los componentes del servicio de red, base de datos, mecanismo de autenticación, `RabbitMQ` y las notificaciones de cambios de topología.

   Nota: el archivo `neutron.conf` se muestra en el apéndice C.14

3. Se modifica el archivo `/etc/neutron/plugins/ml2/ml2_conf.ini` para configurar el `plug-in Modular Layer 2 (ML2)` y permitir que `Open vSwitch` cree infraestructura de capa 2.

   Nota: el archivo `ml2_conf.ini` se muestra en el apéndice C.15.

4. Se habilita el soporte de puente (*bridge*) de red (ver procedimiento en el apéndice C.4) con el objetivo de implementar el *firewall* de `OpenStack`.

5. Se inicializa la base de datos con la configuración más reciente mediante el comando de la consola 4.85 que proporciona la configuración para *neutron* y *ml2*.

   Consola 4.85: Comando para crear las tablas de la base de datos de Neutron.

   ```
   root@controllerCU:~# neutron-db-manage --config-file /etc/neutron/\
       neutron.conf \
   --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head
   ```

Nota: la salida de la consola 4.85 se muestra en el apéndice. C.20.

6. Se reinician los siguientes servicios de Neutron utilizando el comando de la consola 4.86.

Consola 4.86: Comandos para reiniciar servicios de Neutron en el nodo de control-CU.

```
root@controllerCU:~# sudo service nova-api restart
sudo service neutron-server restart
```

4.6.2. Instalación y configuración de Neutron en el nodo de cómputo

El proceso de instalación y configuración de Neutron en el nodo de cómputo-CU debe replicarse en el nodo de cómputo-HOME.

1. Se instalan los paquetes de Neutron, en el nodo de cómputo, utilizando el comando que se muestra en la consola 4.87.

Consola 4.87: Comando para intalar paquetes de Neutron en el nodo de cómputo.

```
root@computeCU:~# apt install neutron-openvswitch-agent neutron-dhcp-\
    agent neutron-metadata-agent -y
```

donde:

- neutron-openvswitch-agent: configura Open vSwitch con el objetivo de crear infraestructura de red.
- neutron-dhcp-agent: proporciona servicio de DHCP a las redes creadas.
- neutron-metadata-agent: permite que las instancias accedan a información de configuración, como credenciales y datos de usuarios.

2. Al igual que en el nodo de control, se edita el archivo /etc/neutron/neutron.conf. El archivo neutron.conf del nodo de cómputo se muestra en el apéndice C.23.

3. Se modifica el archivo /etc/neutron/plugins/ml2/openvswitch_agent.ini para que Open vSwitch pueda crear infraestructura virtual de capa 2. El archivo openvswitch_agent.ini del nodo de cómputo se muestra en el apéndice C.24.

4. Se configura el servicio de DHCP en el archivo /etc/neutron/dhcp_agent.ini. El archivo dhcp_agent.ini se muestra en el apéndice C.25.

5. Se edita el archivo /etc/neutron/metadata_agent.ini para que el agente de metadatos pueda proveer a las instancias de los archivos de configuración y datos de usuarios. El archivo metadata_agent.ini se muestra en el apéndice C.26.

6. Se habilita el soporte de puente (*bridge*) de red (ver procedimiento en apéndice C.4) con el objetivo de implementar el firewall de OpenStack.

7. Se crea un *switch* con Open vSwitch como se indica en el apéndice C.5 para proveer de Internet a las instancias creadas por OpenStack.

8. Se reinician los siguientes servicios de Neutron utilizando el comando de la consola 4.88.

Consola 4.88: Comandos para reiniciar servicios de Neutron en el nodo de cómputo.

```
root@computeCU:~# sudo service nova-compute restart
sudo service neutron-openvswitch-agent restart
sudo service neutron-dhcp-agent restart
sudo service neutron-metadata-agent restart
```

4.6.3. Instalación y configuración de Neutron en el nodo de red

El proceso de instalación y configuración de Neutron en el nodo de red-CU debe replicarse en el nodo de red-HOME.

1. Se instalan los paquetes de Neutron, en el nodo de red, utilizando el comando que se muestra en la consola 4.89.

Consola 4.89: Comando para instalar componentes de Neutron en nodo de red.

```
root@netCU:~# apt install neutron-openvswitch-agent neutron-l3-agent \
    -y
```

donde:

- neutron-openvswitch-agent: configura Open vSwitch con el objetivo de crear infraestructura de red.
- neutron-l3-agent: ofrece servicios de capa 3, como encaminadores (*routers*) virtuales e *IP* flotantes. Proporciona reenvíos de capa 3 para dar acceso a la red externa a las máquinas virtuales.

2. Al igual que en los anteriores nodos, se edita el archivo /etc/neutron/neutron.conf. En este caso solo se habilita el servicio de Keystone, RabbitMQ, y se define el uso del plug-in Modular Layer 2 (ML2). El archivo neutron.conf del nodo de red se muestra en el apéndice C.27.

3. Se modifica el archivo /etc/neutron/plugins/ml2/openvswitch_agent.ini para que Open vSwitch pueda crear infraestructura virtual de capa 2 para las instancias. El archivo openvswitch_agent.ini del nodo de red se muestra en el apéndice C.28.

4. En el archivo /etc/neutron/l3_agent.ini se configura el agente de capa 3 para servicios como: *routers* virtuales e *IP* flotantes. El archivo l3_agent.ini se muestra en el apéndice C.29.

5. Al igual que en el nodo de cómputo, se habilita el soporte de *bridge* (ver procedimiento en apéndice C.4) con el objetivo de implementar el *firewall* de OpenStack.

6. De la misma forma que el nodo de cómputo, se crea un *switch* con OpenvSwitch como se indica en el apéndice C.5, esto con el fin de proveer de Internet a las instancias creadas por OpenStack.

7. Se reinician los siguientes servicios de Neutron utilizando el comando de la consola 4.90.

Consola 4.90: Comandos para reiniciar los servicios de Neutron en el nodo de red-CU.

```
root@netCU:~# sudo service neutron-openvswitch-agent restart
sudo service neutron-l3-agent restart
```

4.6.4. Verificación de la operación del servicio de red

1. Desde el nodo de control se adquieren las variables del usuario admin. Ver comando y requisitos en C.3.

2. Si la instalación y la configuración de Neutron se hizo de forma correcta, se debe obtener la salida mostrada en la consola 4.91.

Consola 4.91: Comando para verifica el funcionamiento de Neutron.

```
root@controllerCU:~# openstack network agent list
+--------------------------------------+--------------------+--------------------+-------------------+-------+-------+-----------------------\
| ID                                   | Agent Type         | Host               | Availability Zone | Alive | State | Binary \
                        |
+--------------------------------------+--------------------+--------------------+-------------------+-------+-------+-----------------------\
| 0891f49a-e976-4563-b740-d55db22aaadf | L3 agent           | ubuntu-controller | nova  | :-)   | UP    | neutron-l3-agent          |
| 58306acd-fc47-4b4c-b2a1-ebfc2218bdee | Metadata agent     | ubuntu-controller | None  | :-)   | UP    | neutron-metadata-agent    |
```

```
| 8171836f-77c6-451f-93e9-4cea6a15eb07 | Open vSwitch agent | ubuntu-controller | None  | :-)   | UP    | neutron-openvswitch-agent |
| e02c09a6-5277-4dd6-b510-b908885c721f | Open vSwitch agent | ubuntu-compute    | None  | :-)   | UP    | neutron-openvswitch-agent |
| e5d7783a-45b1-4237-98c5-f85cf33514fe | DHCP agent         | ubuntu-controller | nova  | :-)   | UP    | neutron-dhcp-agent        |
+--------------------------------------+--------------------+-------------------+-------+-------+-------+---------------------------+
```

4.7. Horizon: OpenStack Dashboard

`Horizon` es la plataforma web que permite administrar la nube de forma visual. Por defecto trabaja en el protocolo *HTTP* que transmite datos no cifrados, lo que representa un riesgo, ya que la información del usuario es enviada desde el navegador sin cifrar, y esta podría ser interceptada por cualquier persona que este monitorizando la sesión. En este proyecto se configuró el servicio para que funciones sobre *HTTPS* mediante un certificado que trabaje con *SSL/TLS* firmado por una autoridad certificadora (ver sección 3.6).

El proceso de instalación y configuración de `Horizon` en el nodo de control-CU debe replicarse en el nodo de control-HOME.

4.7.1. Instalación de Horizon

1. Se instala `Horizon` utilizando el comando que se muestra en la consola 4.92.

Consola 4.92: Comando para instalar Horizon.

```
root@controllerCU:~# apt install openstack-dashboard -y
```

2. Se edita el archivo `/etc/openstack-dashboard/local_settings.py` para habilitar las funciones del panel de control. El archivo completo `"local_settings.p"` se puede ver en el apéndice C.17.

3. Se configura el archivo `/etc/apache2/conf-available/openstack-dashboard.conf` (Ver en Apéndice C.18) para que todas las solicitudes al puerto 80 sean redirigidas al 443. Además, se indica la dirección del certificado para que el servidor `Apache` pueda habilitar el canal de cifrado.

4. Se habilita el modulo `"mod_ssl"`que brida soporte para cifrado *SSL/TLS* con el comando de la consola 4.93.

Consola 4.93: Modulo mod_ssl.

```
root@controllerCU:~# a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
```

5. El servidor apache se reinicia para que puedan aplicar los nuevos ajustes de configuración, utilizando el comando que se muestra en la consola 4.94

Consola 4.94: Comando para reiniciar el servidor apache.

```
root@controllerCU:~# systemctl restart apache2
```

4.8. Proyecto de OpenStack

Por cuestiones de seguridad se debe crear un proyecto y un usuario independiente al usuario `admin` para utilizar los servicios de `OpenStack`.

4.8.1. Creación del proyecto en la NUBE-HOME

1. Utilizando el comando de la consola 4.95, se crea el proyecto `home-workspace`.

Consola 4.95: Comando para crear el proyecto home-workspace.

```
root@controllerHOME:~# openstack project create --domain default \
  --description "Ambiente para usuarios en sucursal-home" home-\
     workspace
+-------------+-----------------------------------------+
| Field       | Value                                   |
+-------------+-----------------------------------------+
| description | Ambiente para usuarios en sucursal-home |
| domain_id   | default                                 |
| enabled     | True                                    |
| id          | 349212974faa419b80d5cbf2d1642bc4        |
| is_domain   | False                                   |
| name        | home-workspace                          |
| options     | {}                                      |
| parent_id   | default                                 |
| tags        | []                                      |
+-------------+-----------------------------------------+
```

2. Se crea el usuario `home-user` con el comando de la consola 4.96.

Consola 4.96: Comando para crear el usuario home-user.

```
root@controllerHOME:~# openstack user create --domain default \
  --password-prompt home-user
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 04719a218fe246c49a3fe66e7eceb450 |
| name                | home-user                        |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

3. Utilizando el comando de la consola 4.97, se agrega el rol `admin` al usuario `home-user` para el proyecto `home-workspace`.

Consola 4.97: Comando para agrega el rol admin al usuario home-user para el proyecto home-workspace.

```
root@controllerHOME:~#openstack role add --project home-workspace --\
    user home-user admin
```

4. Se crean las variables de entorno para el usuario `home-user` como se muestra en el apéndice C.1.2.

4.8.2. Creación del proyecto en la NUBE-CU

1. Utilizando el comando de la consola 4.98, se crea el proyecto `cu-workspace`.

Consola 4.98: Comando para crear el proyecto cu-workspace.

```
root@controllerCU:~# openstack project create --domain default \
--description "Ambiente para usuarios en sucursal-cu" cu-workspace
+-------------+-----------------------------------------+
| Field       | Value                                   |
+-------------+-----------------------------------------+
| description | Ambiente para usuarios en sucursal-cu |
| domain_id   | default                                 |
| enabled     | True                                    |
| id          | e62ca65f1e624346a692e3c9abd09a13        |
| is_domain   | False                                   |
| name        | cu-workspace                            |
| options     | {}                                      |
| parent_id   | default                                 |
| tags        | []                                      |
+-------------+-----------------------------------------+
```

2. Se crea el usuario `cu-user` con el comando de la consola 4.99.

Consola 4.99: Comando para crear el usuario cu-user.

```
root@controllerCU:~# openstack user create --domain default --\
   password-prompt cu-user
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | de9a7daeb49d4a169ff11a92b0f76e40 |
| name                | cu-user                          |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

3. Utilizando el comando de la consola 4.100, se agrega el rol `admin` al usuario `cu-user` para el proyecto `cu-workspace`.

Consola 4.100: Comando para agrega el rol admin al usuario cu-user para el proyecto cu-workspace.

```
root@controllerCU:~# openstack role add --project cu-workspace --user\
    cu-user admin
```

4. Se crean las variables de entorno para el usuario `cu-user` como se muestra en el apéndice C.1.3.

4.9. Creación de redes con OpenStack

`Horizon` permite crear redes de una manera gráfica, sin embargo, en esta tesis se muestra como crear infraestructura de red mediante consola de comandos.

4.9.1. Red de la NUBE-HOME

1. Se adquieren las variables del usuario `admin` con el comando de la consola C.3.

2. Se crea una `provider network`(red del provedor) a la cual se le nombra `home-proveedor1`, que servirá como punto de salida a internet para las redes internas, tal como se muestra en la consola 4.101.

Consola 4.101: Creación de red home-proveedor1.

```
root@controllerHOME:~# openstack network create --share --provider-physical-network provider \
  --provider-network-type flat home-proveedor1
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2024-01-30T20:25:00Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | cb5bbd25-d061-4571-81f7-4b416e26b1aa |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | False                                |
| is_vlan_transparent       | None                                 |
| mtu                       | 1500                                 |
| name                      | home-proveedor1                      |
| port_security_enabled     | True                                 |
| project_id                | 7285197c3e4e4709b46709a49a6f9d17     |
| provider:network_type     | flat                                 |
| provider:physical_network | provider                             |
| provider:segmentation_id  | None                                 |
| qos_policy_id             | None                                 |
| revision_number           | 1                                    |
| router:external           | Internal                             |
| segments                  | None                                 |
| shared                    | True                                 |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      |                                      |
| tenant_id                 | 7285197c3e4e4709b46709a49a6f9d17     |
| updated_at                | 2024-01-30T20:25:00Z                 |
+---------------------------+--------------------------------------+
```

3. Se crea una subred `IPv4` en la red `home-proveedor1` bajo el nombre `home-proveedor1-v4` cuya información detallada se presenta en la consola 4.102.

Consola 4.102: Creación de subred home-proveedor1-v4.

```
root@controllerHOME:~# openstack subnet create --subnet-range 192.168.1.0/24 --gateway \
    192.168.1.1 \
  --network home-proveedor1 --allocation-pool start=192.168.1.2,end=192.168.1.199 \
  --dns-nameserver 8.8.4.4 --dns-nameserver 8.8.8.8 home-proveedor1-v4
+----------------------+--------------------------------------+
| Field                | Value                                |
+----------------------+--------------------------------------+
| allocation_pools     | 192.168.1.2-192.168.1.199            |
| cidr                 | 192.168.1.0/24                       |
| created_at           | 2024-01-30T20:29:17Z                 |
| description          |                                      |
| dns_nameservers      | 8.8.4.4, 8.8.8.8                     |
| dns_publish_fixed_ip | None                                 |
| enable_dhcp          | True                                 |
| gateway_ip           | 192.168.1.1                          |
| host_routes          |                                      |
| id                   | 9c95506f-f3fe-4cf9-8623-87461da29980 |
| ip_version           | 4                                    |
| ipv6_address_mode    | None                                 |
| ipv6_ra_mode         | None                                 |
| name                 | home-proveedor1-v4                   |
| network_id           | cb5bbd25-d061-4571-81f7-4b416e26b1aa |
| project_id           | 7285197c3e4e4709b46709a49a6f9d17     |
| revision_number      | 0                                    |
| segment_id           | None                                 |
| service_types        |                                      |
| subnetpool_id        | None                                 |
| tags                 |                                      |
| updated_at           | 2024-01-30T20:29:17Z                 |
```

```
+---------------------+--------------------------------------+
```

4. Se establece que la red `home-proveedor1` pueda admitir conectividad externa para las `self-service network`, tal como se muestra en la consola 4.103.

Consola 4.103: Comando network set –external home-proveedor1.

```
root@controllerHOME:~# openstack network set --external home-\
   proveedor1
```

5. En la figura 4.1 se muestra el panel de control de `OpenStack` dentro de la NUBE-HOME.

Figura 4.1: Red home-proveedor desde Horizon.

6. Se cargan las variables de entorno del cliente `home-user` mediante el comando de la consola C.7.

7. Se establece otra `self-service network`, esta vez para el cliente `home-user`, la cual se denomina `home-red1`, según se visualiza en la consola 4.104.

Consola 4.104: Comando para crear red llamada home-red1.

```
root@controllerHOME:~# openstack network create home-red1
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2024-01-30T20:39:48Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | 965b781d-bac9-498b-9521-d61e8fb58cc1 |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | False                                |
| is_vlan_transparent       | None                                 |
| mtu                       | 1450                                 |
```

```
| name                        | home-red1                            |
| port_security_enabled       | True                                 |
| project_id                  | 7285197c3e4e4709b46709a49a6f9d17     |
| provider:network_type       | vxlan                                |
| provider:physical_network   | None                                 |
| provider:segmentation_id    | 333                                  |
| qos_policy_id               | None                                 |
| revision_number             | 1                                    |
| router:external             | Internal                             |
| segments                    | None                                 |
| shared                      | False                                |
| status                      | ACTIVE                               |
| subnets                     |                                      |
| tags                        |                                      |
| tenant_id                   | 7285197c3e4e4709b46709a49a6f9d17     |
| updated_at                  | 2024-01-30T20:39:48Z                 |
+-----------------------------+--------------------------------------+
```

8. Se crea una subred `IPv4` en la red `home-red1` bajo el nombre `home-red1-v4` cuya información detallada se presenta en la consola 4.105.

Consola 4.105: Creación de subred home-red1-v4.

```
root@controllerHOME:~#openstack subnet create --subnet-range 192.168.2.0/24 \
  --network home-red1 --dns-nameserver 8.8.4.4 \
  --dns-nameserver 8.8.8.8 home-red1-v4
+----------------------+--------------------------------------+
| Field                | Value                                |
+----------------------+--------------------------------------+
| allocation_pools     | 192.168.2.2-192.168.2.254            |
| cidr                 | 192.168.2.0/24                       |
| created_at           | 2024-01-30T20:41:53Z                 |
| description          |                                      |
| dns_nameservers      | 8.8.4.4, 8.8.8.8                     |
| dns_publish_fixed_ip | None                                 |
| enable_dhcp          | True                                 |
| gateway_ip           | 192.168.2.1                          |
| host_routes          |                                      |
| id                   | 95efb6ee-efa9-4b3a-a838-1a88bce5607e |
| ip_version           | 4                                    |
| ipv6_address_mode    | None                                 |
| ipv6_ra_mode         | None                                 |
| name                 | home-red1-v4                         |
| network_id           | 965b781d-bac9-498b-9521-d61e8fb58cc1 |
| project_id           | 7285197c3e4e4709b46709a49a6f9d17     |
| revision_number      | 0                                    |
| segment_id           | None                                 |
| service_types        |                                      |
| subnetpool_id        | None                                 |
| tags                 |                                      |
| updated_at           | 2024-01-30T20:41:53Z                 |
+----------------------+--------------------------------------+
```

9. Se crea un router y se le nombra `home-router1`, como se indica en la consola 4.106.

Consola 4.106: Creación de home-router1.

```
root@controllerHOME:~# openstack router create home-router1
+-------------------------+--------------------------------------+
| Field                   | Value                                |
```

```
+-------------------------+--------------------------------------+
| admin_state_up          | UP                                   |
| availability_zone_hints |                                      |
| availability_zones      |                                      |
| created_at              | 2024-01-30T20:42:14Z                 |
| description             |                                      |
| distributed             | False                                |
| enable_ndp_proxy        | None                                 |
| external_gateway_info   | null                                 |
| flavor_id               | None                                 |
| ha                      | False                                |
| id                      | 77911275-d3bf-4763-afce-27691c2db5d3 |
| name                    | home-router1                         |
| project_id              | 7285197c3e4e4709b46709a49a6f9d17     |
| revision_number         | 1                                    |
| routes                  |                                      |
| status                  | ACTIVE                               |
| tags                    |                                      |
| tenant_id               | 7285197c3e4e4709b46709a49a6f9d17     |
| updated_at              | 2024-01-30T20:42:14Z                 |
+-------------------------+--------------------------------------+
```

10. Se agregan la subredes `IPv4` como interfaces del *router* y la red `home-proveedor1` como puerta de enlace, tal como se observa en la consola 4.107.

Consola 4.107: Comando openstack router add subnet home-router1 home-red1-v4.

```
root@controllerHOME:~# openstack router add subnet home-router1 home-red1-v4
openstack router set --external-gateway home-proveedor1 home-router1
```

11. Tras la creación de la primera red interna, se puede visualizar en `Horizon` la nueva topología de la NUBE-HOME. En la figura 4.2 se observa el *router*(`home-router1`), la red del proveedor (`home-proveedor1`) y una `self-service network` (`home-red1`).

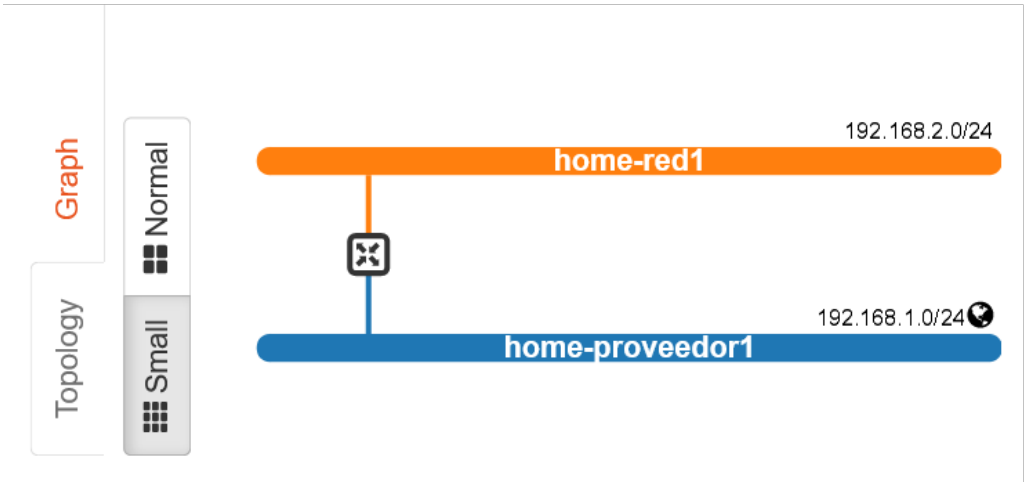

Figura 4.2: Primera topología de la red NUBE-HOME desde Horizon.

12. De forma similar a como se creó la red `home-red1` y el *router* `home-router1`. Se crean 3 redes y 2 *routers* adicionales. En la figura 4.3 se observan tres *routers*, la red del proveedor (`home-proveedor1`) y cuatro `self-service network`.

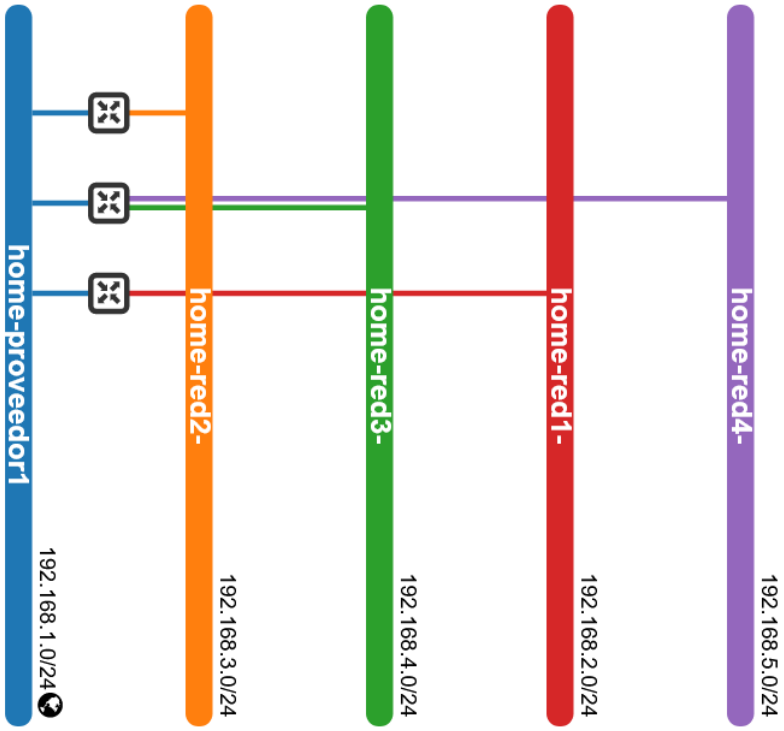

Figura 4.3: Topología completa de la NUBE-HOME.

4.9.2. Red de la NUBE-CU

El proceso para crear redes en la NUBE-CU sigue un patrón similar al llevado a cabo anteriormente en la NUBE-HOME, con la diferencia de que en la NUBE-CU solo se requiere una `provider network` (red del proveedor).

1. Se cargan las variables de entorno del usuario `admin` mediante el comando de la consola C.3.

2. Se crea una `provider network` (red del proveedor) a la cual se le nombra `cu-proveedor1`, tal como se muestra en la consola 4.108.

Consola 4.108: Creación de red cu-proveedor1.

```
root@controllerCU:~# openstack network create --share --provider-physical-network provider \
  --provider-network-type flat cu-proveedor1
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2024-02-01T01:50:17Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | 0acea49a-3e7c-4ed3-8a10-e70569c15c7d |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | False                                |
| is_vlan_transparent       | None                                 |
| mtu                       | 1500                                 |
| name                      | cu-proveedor1                        |
| port_security_enabled     | True                                 |
| project_id                | 7a0ef556d0de484d85baab7d4656edda     |
| provider:network_type     | flat                                 |
| provider:physical_network | provider                             |
| provider:segmentation_id  | None                                 |
| qos_policy_id             | None                                 |
| revision_number           | 1                                    |
| router:external           | Internal                             |
```

```
| segments                  | None                                 |
| shared                    | True                                 |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      |                                      |
| tenant_id                 | 7a0ef556d0de484d85baab7d4656edda     |
| updated_at                | 2024-02-01T01:50:17Z                 |
+---------------------------+--------------------------------------+
```

3. Se crea una subred `IPv4` en la red `cu-proveedor1` bajo el nombre `cu-proveedor1-v4` cuya información detallada se presenta en la consola 4.109.

Consola 4.109: Creación de subred cu-proveedor1-v4.

```
root@controllerCU:~# openstack subnet create --subnet-range 192.168.10.0/24 --gateway 192.168.10.1 \
  --network cu-proveedor1 --allocation-pool start=192.168.10.10,end=192.168.10.100 \
  --dns-nameserver 8.8.4.4 --dns-nameserver 8.8.8.8 cu-proveedor1-v4
+----------------------+--------------------------------------+
| Field                | Value                                |
+----------------------+--------------------------------------+
| allocation_pools     | 192.168.10.10-192.168.10.100         |
| cidr                 | 192.168.10.0/24                      |
| created_at           | 2024-02-01T01:51:48Z                 |
| description          |                                      |
| dns_nameservers      | 8.8.4.4, 8.8.8.8                     |
| dns_publish_fixed_ip | None                                 |
| enable_dhcp          | True                                 |
| gateway_ip           | 192.168.10.1                         |
| host_routes          |                                      |
| id                   | 541cf373-23ca-4db2-8fa4-e6a260208c49 |
| ip_version           | 4                                    |
| ipv6_address_mode    | None                                 |
| ipv6_ra_mode         | None                                 |
| name                 | cu-proveedor1-v4                     |
| network_id           | 0acea49a-3e7c-4ed3-8a10-e70569c15c7d |
| project_id           | 7a0ef556d0de484d85baab7d4656edda     |
| revision_number      | 0                                    |
| segment_id           | None                                 |
| service_types        |                                      |
| subnetpool_id        | None                                 |
| tags                 |                                      |
| updated_at           | 2024-02-01T01:51:48Z                 |
+----------------------+--------------------------------------+
```

4. Se establece que la red `cu-proveedor1` pueda admitir conectividad externa, tal como se ve en la consola 4.110.

Consola 4.110: Comando openstack network set –external cu-proveedor1.

```
root@controllerCU:~# openstack network set --external cu-proveedor1
```

5. Dentro del panel de control de `OpenStack` se tiene hasta el momento la topología en la NUBE-CU mostrada en la figura 4.4.

Figura 4.4: Primera topología de la red NUBE-CU desde Horizon.

4.10. Creación de máquinas virtuales con OpenStack

Antes de generar máquinas virtuales es importante mencionar que dentro del entorno de Openstack existen conceptos como *flavor*, *keypair* y *security group* que buscan definir y organizar los atributos que se le otorgan a las máquinas virtuales.

Para establecer los valores de *flavor*, *keypair* y *security group* se utiliza la línea de comandos en el nodo de control de cada una de las nubes de la siguiente forma.

4.10.1. Flavor

En OpenStack, *flavor* define la capacidad informática de memoria y de almacenamiento de las instancias de Nova [52].

La forma en cómo se crean los *flavors* en OpenStack es la siguiente:

1. Se cargan las variables de entorno del usuario admin con el comando de la consola C.3.
2. Se definen varios *flavors* en la consola 4.111, con el fin de tener disponible varios perfiles al momento de crear máquinas virtuales.

Consola 4.111: Creación de flavors.

```
root@controllerHOME:~#openstack flavor create m1.tiny --id 1 --ram 512 --disk 1 --vcpus 1
root@controllerHOME:~#openstack flavor create m1.small --id 2 --ram 2048 --disk 20 --vcpus 1
root@controllerHOME:~#openstack flavor create m1.medium --id 3 --ram 4096 --disk 40 --vcpus 2
root@controllerHOME:~#openstack flavor create m1.large --id 4 --ram 8192 --disk 80 --vcpus 4
root@controllerHOME:~#openstack flavor create m1.xlarge --id 5 --ram 16384 --disk 160 --vcpus 8
root@controllerHOME:~#openstack flavor create m1.debian --id 6 --ram 1024 --disk 10 --vcpus 1
+----------------------------+-----------+
| Field                      | Value     |
+----------------------------+-----------+
| OS-FLV-DISABLED:disabled   | False     |
| OS-FLV-EXT-DATA:ephemeral  | 0         |
| description                | None      |
| disk                       | 10        |
| id                         | 6         |
| name                       | m1.debian |
| os-flavor-access:is_public | True      |
| properties                 |           |
| ram                        | 1024      |
| rxtx_factor                | 1.0       |
| swap                       |           |
| vcpus                      | 1         |
+----------------------------+-----------+
+----------------------------+---------+
| Field                      | Value   |
+----------------------------+---------+
| OS-FLV-DISABLED:disabled   | False   |
| OS-FLV-EXT-DATA:ephemeral  | 0       |
| description                | None    |
| disk                       | 1       |
| id                         | 1       |
| name                       | m1.tiny |
| os-flavor-access:is_public | True    |
| properties                 |         |
| ram                        | 512     |
| rxtx_factor                | 1.0     |
| swap                       |         |
| vcpus                      | 1       |
+----------------------------+---------+
+----------------------------+----------+
| Field                      | Value    |
+----------------------------+----------+
| OS-FLV-DISABLED:disabled   | False    |
| OS-FLV-EXT-DATA:ephemeral  | 0        |
| description                | None     |
| disk                       | 20       |
| id                         | 2        |
| name                       | m1.small |
| os-flavor-access:is_public | True     |
| properties                 |          |
| ram                        | 2048     |
| rxtx_factor                | 1.0      |
| swap                       |          |
| vcpus                      | 1        |
+----------------------------+----------+
+----------------------------+-----------+
```

```
| Field                       | Value     |
+-----------------------------+-----------+
| OS-FLV-DISABLED:disabled    | False     |
| OS-FLV-EXT-DATA:ephemeral   | 0         |
| description                 | None      |
| disk                        | 40        |
| id                          | 3         |
| name                        | m1.medium |
| os-flavor-access:is_public  | True      |
| properties                  |           |
| ram                         | 4096      |
| rxtx_factor                 | 1.0       |
| swap                        |           |
| vcpus                       | 2         |
+-----------------------------+-----------+
+-----------------------------+----------+
| Field                       | Value    |
+-----------------------------+----------+
| OS-FLV-DISABLED:disabled    | False    |
| OS-FLV-EXT-DATA:ephemeral   | 0        |
| description                 | None     |
| disk                        | 80       |
| id                          | 4        |
| name                        | m1.large |
| os-flavor-access:is_public  | True     |
| properties                  |          |
| ram                         | 8192     |
| rxtx_factor                 | 1.0      |
| swap                        |          |
| vcpus                       | 4        |
+-----------------------------+----------+
+-----------------------------+-----------+
| Field                       | Value     |
+-----------------------------+-----------+
| OS-FLV-DISABLED:disabled    | False     |
| OS-FLV-EXT-DATA:ephemeral   | 0         |
| description                 | None      |
| disk                        | 160       |
| id                          | 5         |
| name                        | m1.xlarge |
| os-flavor-access:is_public  | True      |
| properties                  |           |
| ram                         | 16384     |
| rxtx_factor                 | 1.0       |
| swap                        |           |
| vcpus                       | 8         |
+-----------------------------+-----------+
```

4.10.2. Keypair

Keypair se refiere a la clave pública para acceder a los VPS por medio del protocolo SSH. La forma de agregar una keypair en `OpenStack` es la siguiente:

1. Se cargan las variables de entorno del proyecto respectivo, es decir, en la NUBE-HOME, las del cliente `home-user` con el comando de la consola C.7, y en la NUBE-CU, las del cliente `cu-user` con el comando de la consola C.10.

2. Se generan la clave pública y privada mediante el comando que se muestra en la consola 4.112.

Consola 4.112: Comando para genera la clave pública y privada.

```
root@controllerHOME:~# ssh-keygen -q -N ""
Enter file in which to save the key (/root/.ssh/id_rsa):
```

3. Utilizando el comando de la consola 4.113, se agrega la clave pública al servicio de cómputo y se le nombra *mykey* dentro del entorno de `OpenStack`.

Consola 4.113: Comando para agrega la clave pública al servicio de cómputo.

```
root@controllerHOME:~# openstack keypair create --public-key ~/.ssh/\
   id_rsa.pub mykey
+-------------+-------------------------------------------------+
```

```
| Field       | Value                                                |
+-------------+------------------------------------------------------+
| created_at  | None                                                 |
| fingerprint | 80:40:8a:e8:12:bd:7f:53:db:fa:23:a0:6c:3c:5f:69      |
| id          | mykey                                                |
| is_deleted  | None                                                 |
| name        | mykey                                                |
| type        | ssh                                                  |
| user_id     | ce95c67c0d2943db9e15b21433fd0679                     |
+-------------+------------------------------------------------------+
```

4. Para verificar que *mykey* se ha creado correctamente, se enumeran los *keypair* utilizando el comando que se muestra en la consola 4.114.

Consola 4.114: Comando para listar las keypair.

```
root@controllerHOME:~# openstack keypair list
+-------+-------------------------------------------------+------+
| Name  | Fingerprint                                     | Type |
+-------+-------------------------------------------------+------+
| mykey | 80:40:8a:e8:12:bd:7f:53:db:fa:23:a0:6c:3c:5f:69 | ssh  |
+-------+-------------------------------------------------+------+
```

4.10.3. Security Group

Los *Security Groups* son un conjunto de reglas de filtrado que definen el acceso a la red. Los grupos de seguridad niegan todo de forma predeterminada. Una regla de grupo de seguridad tiene los siguientes atributos: un protocolo como *IP*, ICMP, TCP o UDP, un puerto o rango de puertos destino , y un rango de *IP* remoto en formato CIDR [51].

Por razones de seguridad no es una buena práctica abrir todos lo puertos de un VPS. Sin embargo, esta tesis se construyó bajo un ambiente controlado donde los servidores físicos cuentan con un *firewall* que bloquean todas las conexiones externas. Por tanto se creó un *Security Group* que permite todo el tráfico desde cualquier puerto. No obstante, se pueden crear reglas que solo dejen pasar la información necesaria. A continuación se muestra como se genera un grupo de seguridad:

1. Al igual que con las *keypair*, se cargan las variables de entorno del proyecto, para este ejemplo, las variables del cliente `home-user` con el comando de la consola C.7. O de la misma forma se cargan las variables de entorno para el cliente `cu-user`.

2. Se crea un nuevo *Security Group* llamado *open* utilizando el comando que se muestra en la consola 4.115.

Consola 4.115: Creación de un Security Group.

```
root@controllerHOME:~# openstack security group create open --description "All open"
+-----------------+-----------------------------------------------------------------------------------------------------+
| Field           | Value
                                                                                                      |
+-----------------+-----------------------------------------------------------------------------------------------------+
| created_at      | 2023-09-20T21:20:36Z
                                                                                                      |
| description     | All open
                                                                                                      |
| id              | e90286c1-ad0c-4f87-a936-dc408581e316
                                                                                                      |
| name            | open
                                                                                                      |
| project_id      | 7a0ef556d0de484d85baab7d4656edda
                                                                                                      |
| revision_number | 1
                                                                                                      |
| rules           | created_at='2023-09-20T21:20:36Z', direction='egress', ethertype='IPv4', id='dc3db79c-2c0c-40df-9d3b-231d02cdf2ad', \
      standard_attr_id='11', updated_at='2023-09-20T21:20:36Z' |
|                 | created_at='2023-09-20T21:20:36Z', direction='egress', ethertype='IPv6', id='ff0c8ce2-382f-487a-9b3f-68e8347ec1e9', \
      standard_attr_id='12', updated_at='2023-09-20T21:20:36Z' |
| shared          | False
                                                                                                      |
| stateful        | True
                                                                                                      |
```

```
| tags              | []                                                                                                |
| updated_at        | 2023-09-20T21:20:36Z                                                                              |
+-------------------+---------------------------------------------------------------------------------------------------+
```

3. Con el comando de la consola 4.116, se abren los puertos para los protocolos TCP y UDP.

Consola 4.116: Comando para abrir los puertos para los protocolos TCP y UDP.

```
root@controllerHOME:~# openstack security group rule create open \
--protocol tcp --dst-port 1:65535  --remote-ip 0.0.0.0/0
+--------------------------+--------------------------------------+
| Field                    | Value                                |
+--------------------------+--------------------------------------+
| created_at               | 2023-09-20T21:20:42Z                 |
| description              |                                      |
| direction                | ingress                              |
| ether_type               | IPv4                                 |
| id                       | e5d63094-266e-4a17-9ac3-65289aa60db3 |
| name                     | None                                 |
| normalized_cidr          | 0.0.0.0/0                            |
| port_range_max           | None                                 |
| port_range_min           | None                                 |
| project_id               | 7a0ef556d0de484d85baab7d4656edda     |
| protocol                 | tcp                                  |
| remote_address_group_id  | None                                 |
| remote_group_id          | None                                 |
| remote_ip_prefix         | 0.0.0.0/0                            |
| revision_number          | 0                                    |
| security_group_id        | e90286c1-ad0c-4f87-a936-dc408581e316 |
| tags                     | []                                   |
| updated_at               | 2023-09-20T21:20:42Z                 |
+--------------------------+--------------------------------------+
root@controllerHOME:~# openstack security group rule create open \
--protocol udp --dst-port 1:65535  --remote-ip 0.0.0.0/0
+--------------------------+--------------------------------------+
| Field                    | Value                                |
+--------------------------+--------------------------------------+
| created_at               | 2023-09-20T21:20:48Z                 |
| description              |                                      |
| direction                | ingress                              |
| ether_type               | IPv4                                 |
| id                       | f6e05276-1d9f-46d1-a527-c20debde7389 |
| name                     | None                                 |
| normalized_cidr          | 0.0.0.0/0                            |
| port_range_max           | None                                 |
| port_range_min           | None                                 |
| project_id               | 7a0ef556d0de484d85baab7d4656edda     |
| protocol                 | udp                                  |
| remote_address_group_id  | None                                 |
| remote_group_id          | None                                 |
| remote_ip_prefix         | 0.0.0.0/0                            |
| revision_number          | 0                                    |
| security_group_id        | e90286c1-ad0c-4f87-a936-dc408581e316 |
| tags                     | []                                   |
| updated_at               | 2023-09-20T21:20:48Z                 |
+--------------------------+--------------------------------------+
```

4. Se establece una regla para permitir el protocolo ICMP (`ping`) utilizando el comando que se muestra en la consola 4.117.

Consola 4.117: Comando para establecer la regla para permitir el protocolo ICMP.

```
root@controllerHOME:~# openstack security group rule create --proto icmp open
+-------------------------+--------------------------------------+
| Field                   | Value                                |
+-------------------------+--------------------------------------+
| created_at              | 2023-09-20T21:20:54Z                 |
| description             |                                      |
| direction               | ingress                              |
| ether_type              | IPv4                                 |
| id                      | 6a8a8b85-6384-46a6-a36e-a95242db33e8 |
| name                    | None                                 |
| normalized_cidr         | 0.0.0.0/0                            |
| port_range_max          | None                                 |
| port_range_min          | None                                 |
| project_id              | 7a0ef556d0de484d85baab7d4656edda     |
| protocol                | icmp                                 |
| remote_address_group_id | None                                 |
| remote_group_id         | None                                 |
| remote_ip_prefix        | 0.0.0.0/0                            |
| revision_number         | 0                                    |
| security_group_id       | e90286c1-ad0c-4f87-a936-dc408581e316 |
| tags                    | []                                   |
| updated_at              | 2023-09-20T21:20:54Z                 |
+-------------------------+--------------------------------------+
```

5. Se verifica la presencia de las nuevas reglas del security group open utilizando el comando que se muestra en la consola 4.118.

Consola 4.118: Enumeración de las reglas del security group open.

```
root@controllerHOME:~# openstack security group rule list open
+--------------------------------------+-------------+-----------+-----------+------------+-----------+-----------------------+--------+
| ID                                   | IP Protocol | Ethertype | IP Range  | Port Range | Direction | Remote Security Group | Remote \
      Address Group |
+--------------------------------------+-------------+-----------+-----------+------------+-----------+-----------------------+------+
| 04a75010-4bfc-4f42-a278-49b36885ba5c | None        | IPv6      | ::/0      |            | egress    | None                  | None \
                    |
| 8ced14ed-86b9-4eec-aafc-5358ce8d7d12 | udp         | IPv4      | 0.0.0.0/0 |            | ingress   | None                  | None \
                    |
| ccfaa94d-9c74-4fab-aa9c-920341b13266 | icmp        | IPv4      | 0.0.0.0/0 |            | ingress   | None                  | None \
                    |
| df4828cb-3ac9-44df-85b4-68a33bba1b62 | tcp         | IPv4      | 0.0.0.0/0 |            | ingress   | None                  | None \
                    |
| e70836c4-9a0f-4bdb-9757-f0e1bceff987 | None        | IPv4      | 0.0.0.0/0 |            | egress    | None                  | None \
                    |
+--------------------------------------+-------------+-----------+-----------+------------+-----------+-----------------------+--------+
```

En la consola 4.118 se observan las reglas del *security group* recién creado llamando *open* que permite el egreso y el ingreso de los protocolos TCP, UDP, ICMP para cualquier *IP*.

4.10.4. Script de inicio de sesión

Las imágenes creadas en el repositorio (`Glance`) de la sección 4.3.3 tienen por defecto un usuario cuyo nombre es el nombre de la distribución (Debian, Ubuntu, etc), y por defecto no lleva contraseña. Estas imágenes están diseñadas para ingresar por medio de *ssh* mediante la llave creada en la sección 4.10.2. Sin embargo, es posible configurar una contraseña mediante el script(ver consola 4.119) que se guarda como archivo de texto llamado `userdata.txt`.

Consola 4.119: Script de inicio de sesión.

```
#cloud-config
password: mypasswd
chpasswd: { expire: False }
ssh_pwauth: True
```

Nota: la contraseña de la consola 4.119 solo es un ejemplo representativo.

4.10.5. Máquinas virtuales de la NUBE-HOME

Una vez definidos los valores de *flavor*, *keypair* y *security group*, se pueden crear *VM* y asignarles atributos específicos.

`Horizon` permite crear *VM* de una manera gráfica, sin embargo, en este trabajo se presenta la creación de *VM* mediante consola de comandos.

A continuación se muestra como se genera una máquina virtual:

1. Se exportan las variables del cliente `home-user` mediante el comando de la consola C.7.

2. Se crea la máquina virtual utilizando el comando que se muestra en la consola 4.120.

Consola 4.120: Comando para crea una máquina virtual.

```
root@controllerHOME:~# openstack server create --flavor m1.debian --image debian-12 \
  --nic net-id=home-red1 --security-group open \
  --key-name mykey --user-data=userdata.txt HOME-pc1
+-------------------------------------+-----------------------------------------------------+
| Field                               | Value                                               |
+-------------------------------------+-----------------------------------------------------+
| OS-DCF:diskConfig                   | MANUAL                                              |
| OS-EXT-AZ:availability_zone         |                                                     |
| OS-EXT-SRV-ATTR:host                | None                                                |
| OS-EXT-SRV-ATTR:hypervisor_hostname | None                                                |
| OS-EXT-SRV-ATTR:instance_name       |                                                     |
| OS-EXT-STS:power_state              | NOSTATE                                             |
| OS-EXT-STS:task_state               | scheduling                                          |
| OS-EXT-STS:vm_state                 | building                                            |
| OS-SRV-USG:launched_at              | None                                                |
| OS-SRV-USG:terminated_at            | None                                                |
| accessIPv4                          |                                                     |
| accessIPv6                          |                                                     |
| addresses                           |                                                     |
| adminPass                           | z9zHcjE735My                                        |
| config_drive                        |                                                     |
| created                             | 2024-02-13T19:57:00Z                                |
| flavor                              | m1.debian (6)                                       |
| hostId                              |                                                     |
| id                                  | e2d08fe0-11f3-47a9-88d1-990166baaa53                |
| image                               | debian-12 (4f1f58ac-4dfe-476e-a3ef-13b3bc877279)    |
| key_name                            | mykey                                               |
| name                                | HOME-pc1                                            |
| progress                            | 0                                                   |
| project_id                          | 349212974faa419b80d5cbf2d1642bc4                    |
| properties                          |                                                     |
| security_groups                     | name='93243f23-2572-47fc-9a6a-58b89107f21c'         |
| status                              | BUILD                                               |
| updated                             | 2024-02-13T19:57:00Z                                |
| user_id                             | 04719a218fe246c49a3fe66e7eceb450                    |
| volumes_attached                    |                                                     |
+-------------------------------------+-----------------------------------------------------+
```

donde:

- `HOME-pc1`: representa el nombre de la máquina virtual.
- `--flavor m1.debian`: es el valor creado en la consola 4.111. La máquina se genera con 1 vcpus, 1 GB de ram y 10 GB de disco.
- `--image debian-12`: representa el sistema operativo elegido. Todos los sistemas que se cargaron a `Glance` se pueden ver en la sección 4.3.3.
- `--nic net-id home-red1` : red del segmento 2 creada en la sección 4.9.1.
- `--security-group open` : representa las reglas de `Firewall` establecidas en la sección 4.10.3.
- `--key-name mykey`: es la Llave pública para el servicio de SSH generada en la sección4.10.2.
- `--user-data=userdata.txt`: es un archivo para asignación de contraseñada creado en la sección 4.10.4.

3. Se verifica el funcionamiento utilizando la consola *SPICE* (ver figura 4.5).

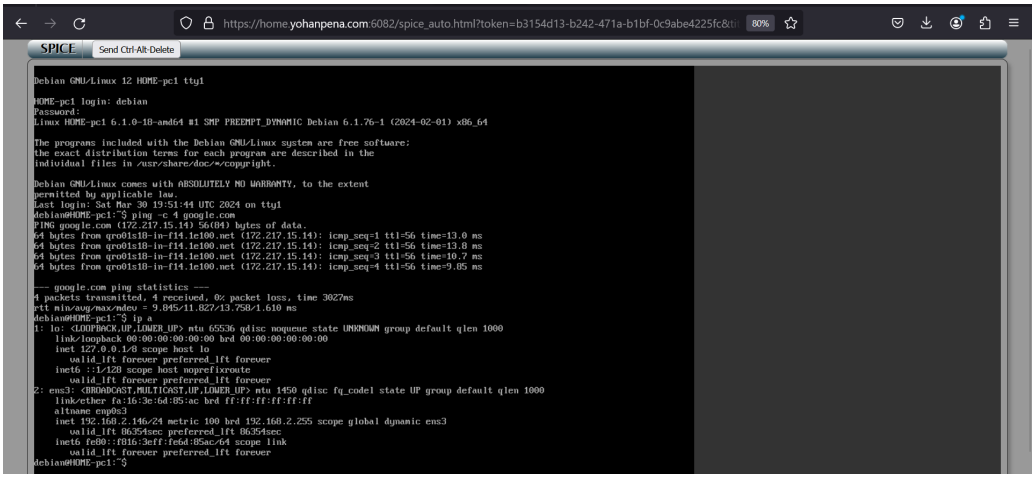

Figura 4.5: Consola SPICE de la máquina virtual HOME-pc1.

Es posible acceder a `HOME-pc1` mediante SSH, sin embargo, se decidió mostrar el funcionamiento mediante *SPICE* para presentar esta funcionalidad que tienen las nubes `OpenStack`.

En la consola *SPICE* de la figura 4.5, se visualiza que se ingresó a la máquina con el usuario `debian` y con la contraseña del archivo `userdata.txt`.

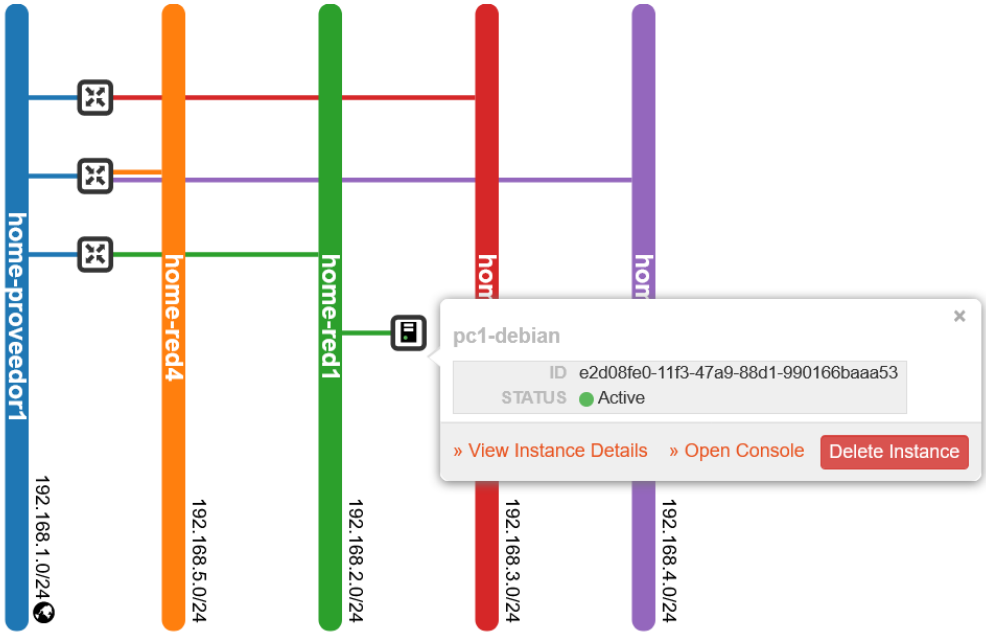

Figura 4.6: HOME-pc1 dentro de la red de la NUBE-HOME.

La figura 4.6 muestra gráficamente que la máquina virtual HOME-pc1 se encuentra en una red interna de `OpenStack` y pude salir a Internet por toda la infraestructura de red creada en la sección 4.9.1.

Empleando un método similar al usado para crear la máquina virtual HOME-pc1 se generaron un total de `4` máquinas, cuyas características se detallan en la tabla 4.1.

Nombre	OS	vCPs	Disco[GB]	RAM[GB]	RED
HOME-pc1	Debian 12	1	10	1	192.168.2.146
HOME-pc2	Debian 12	1	25	3	▪ 192.168.3.189 ▪ 192.168.4.40
HOME-pc3	Fedora 39	1	25	3	192.168.3.227
HOME-pc4	Ubuntu 22	1	25	3	192.168.4.234

Tabla 4.1: Máquinas virtuales de la NUBE-HOME.

La representación completa desde `Horizon` de la NUBE-HOME, incluyendo la infraestructura de red y las 4 máquinas virtuales (ver tabla 4.1), se puede observar en la figura 4.7.

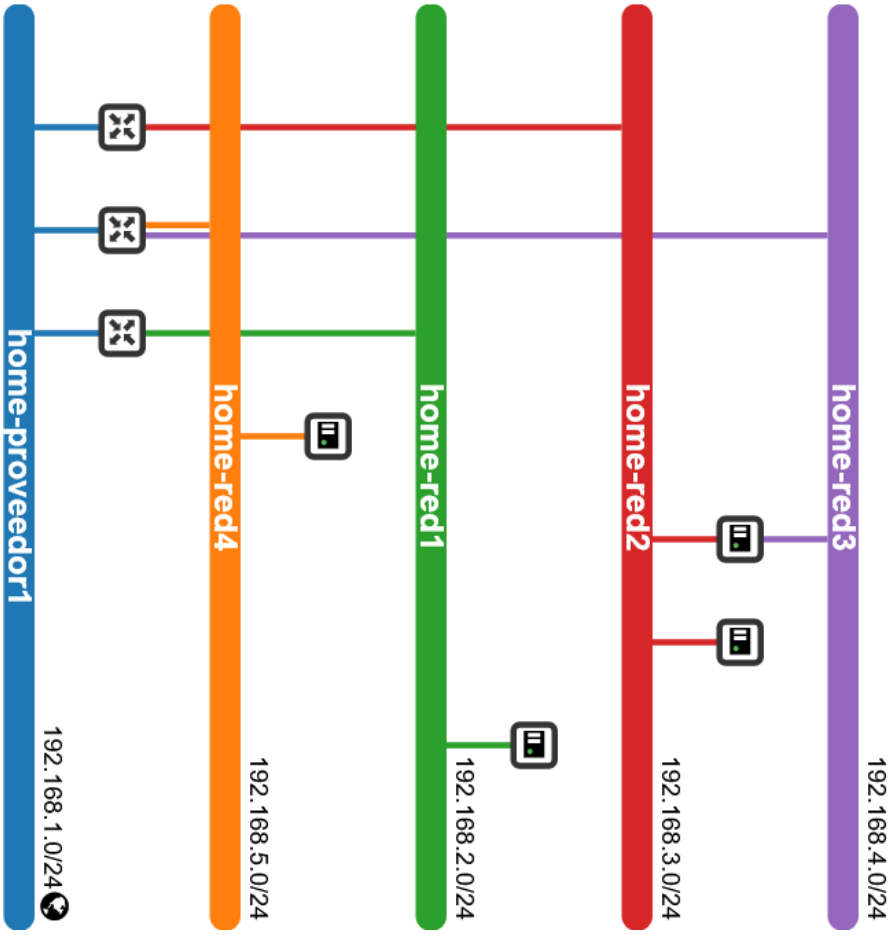

Figura 4.7: NUBE-HOME desde Horizon.

4.10.6. Máquinas virtuales de la NUBE-CU

La metodología para crear las máquinas virtuales, su funcionamiento y los parámetros requeridos son idénticos a los de la NUBE-HOME. Por lo tanto, se presenta un resumen de las características de las máquinas creadas en la NUBE-CU en la tabla 4.2.

Nombre	OS	vCPs	Disco[GB]	RAM[GB]	RED
CU-pc1	Debian 12	4	20	16	192.168.10.26
CU-pc2	Windows 10	4	50	16	192.168.10.80
CU-pc3	Windows-server 2012	2	40	4	192.168.10.29
CU-pc4	Fedora 39	4	20	16	192.168.10.25

Tabla 4.2: Máquinas virtuales de la NUBE-CU.

Gracias a la capacidad y recursos del servidor CU, se crearon máquinas virtuales con interfaz gráficas para demostrar la capacidad de `OpenStack`. A continuación, se muestran imágenes que ilustran cómo se visualizan las máquinas virtuales utilizando la consola *SPICE* dentro de `Horizon`:

- CU-pc1: En la figura 4.8 se aprecia la interfaz gráfica de Debian 12 correspondiente a la máquina virtual CU-pc1.

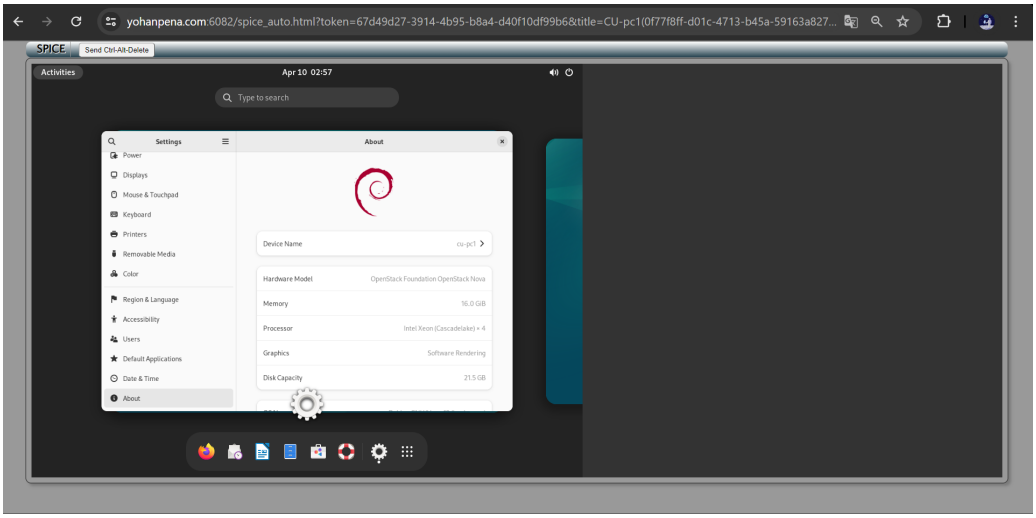

Figura 4.8: Consola SPICE de la máquina virtual CU-pc1 de la NUBE-CU.

- CU-pc2: En la figura 4.9 se observa la interfaz gráfica de Windows 10 correspondiente a la máquina virtual CU-pc2.

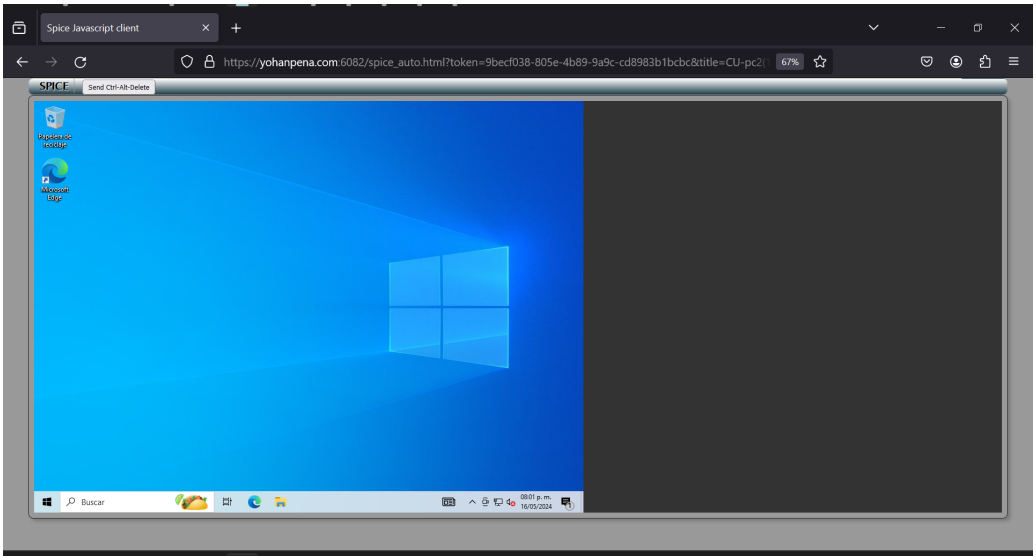

Figura 4.9: Consola SPICE de la máquina virtual CU-pc2 de la NUBE-CU.

- CU-pc3: En la figura 4.10 se aprecia la interfaz gráfica de Windows-server 2012 correspondiente a la máquina virtual CU-pc3.

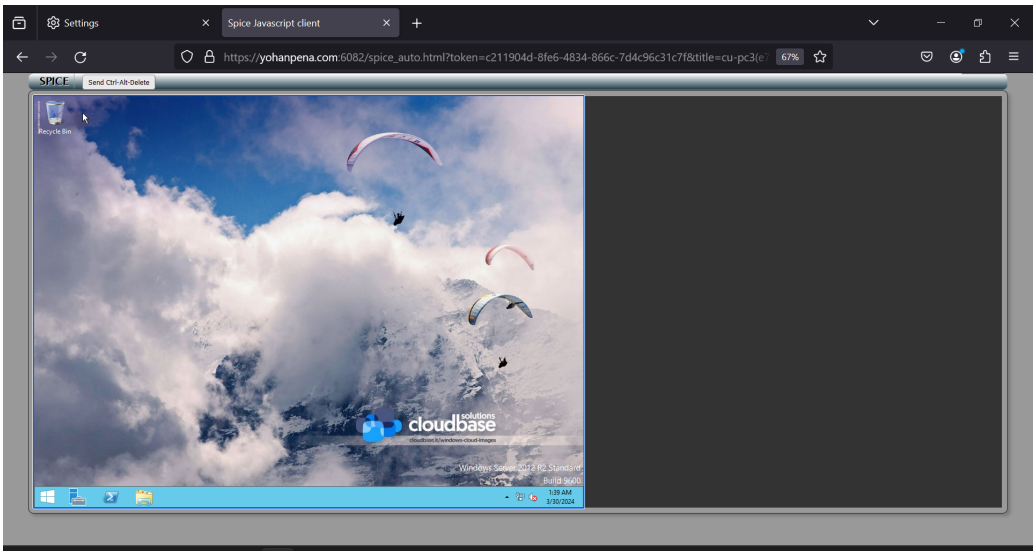

Figura 4.10: Consola SPICE de la máquina virtual CU-pc3 de la NUBE-CU.

- CU-pc4: En la figura 4.10 se observa la interfaz gráfica de Fedora 39 correspondiente a la máquina virtual CU-pc4.

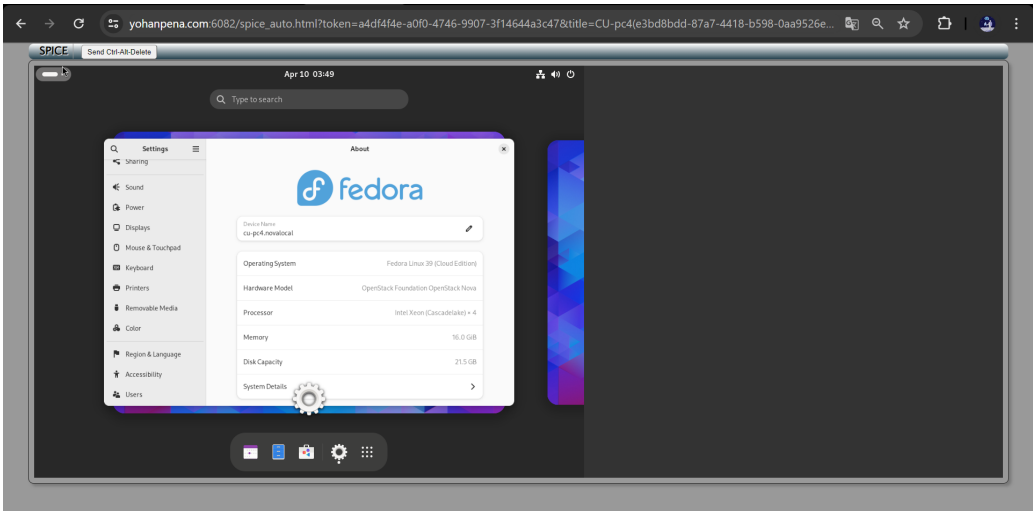

Figura 4.11: Consola SPICE de la máquina virtual CU-pc4 de la NUBE-CU.

La representación completa desde `Horizon` de la NUBE-CU, incluyendo la infraestructura de red y las 4 máquinas virtuales (ver tabla 4.2), se puede observar en la figura 4.12.

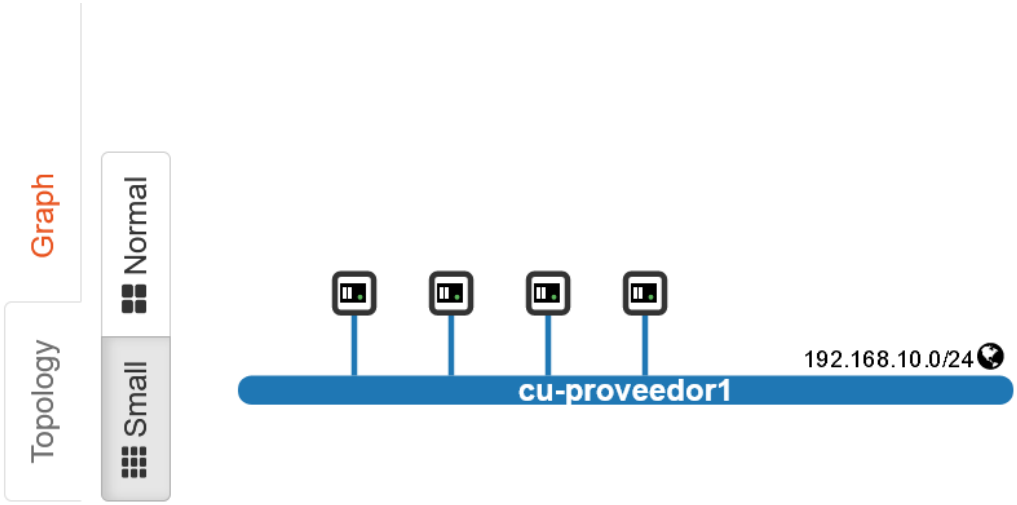

Figura 4.12: NUBE-CU desde Horizon.

Capítulo 5

WireGuard

En este capítulo se da una explicación detallada de cómo implementar el servidor *VPN* (servidor CU) y los clientes *VPN* (servidor HOME y VPS IONOS). Aunque, primero se explica la razón por la que se eligió `WireGuard` como protocolo *VPN* en esta tesis.

Dentro de la sección *Performance* de la documentación de `WireGuard` [43] existe un análisis de desempeño en el cual comparan a `WireGuard` con respecto a `IPsec` y a `OpenVPN`. Para ello utilizan el generador de tráfico *iperf3* entre una máquina Intel Core i7-3820QM y un Intel Core i7-5200U con tarjetas Gigabit Ethernet Intel 82579LM e Intel 1218LM, respectivamente. El resultado durante treinta minutos se muestra en la figura 5.1.

Protocol	Configuration
WireGuard	256-bit ChaCha20, 128-bit Poly1305
IPsec #1	256-bit ChaCha20, 128-bit Poly1305
IPsec #2	256-bit AES, 128-bit GCM
OpenVPN	256-bit AES, HMAC-SHA2-256, UDP mode

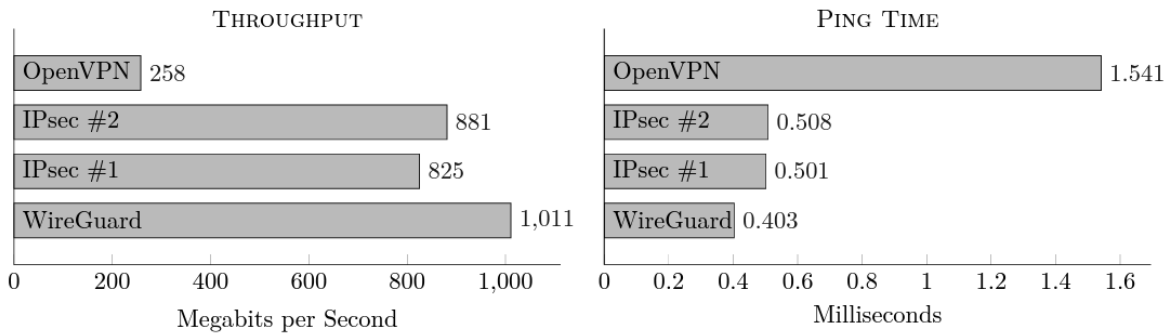

Figura 5.1: Desempeño de WireGuard en comparación con IPsec y OpenVPN. Ver cita [43].

Para ambas métricas, `WireGuard` superó a `OpenVPN` y a ambos modos de `IPsec`. La CPU tuvo una utilización del 100 % durante las pruebas de rendimiento de `OpenVPN` e `IPsec`, sin embargo, no se utilizó por completo para la prueba de `WireGuard`, lo que sugiere que `WireGuard` no satura el enlace *gigabit Ethernet* [43]. Es de esperarse la enorme brecha entre `OpenVPN` y `WireGuard` debido a que `OpenVPN` se ejecuta en el espacio de usuario, lo que significa que existe una latencia y sobrecarga adicional debido a la copia de paquetes entre el espacio de usuario y el espacio de kernel [43].

Actualmente, `WireGuard` ha permitido a varias empresas desarrollar sus propios protocolos *VPN* como es el caso de `NordVPN` con el protocolo `NordLynx` [44]. Todo lo anterior nos brinda una idea clara de la importancia que tiene en la industria `WireGuard`. Por ello, se eligió como protocolo para la creación de la *VPN* en esta tesis.

5.1. Topología de la VPN

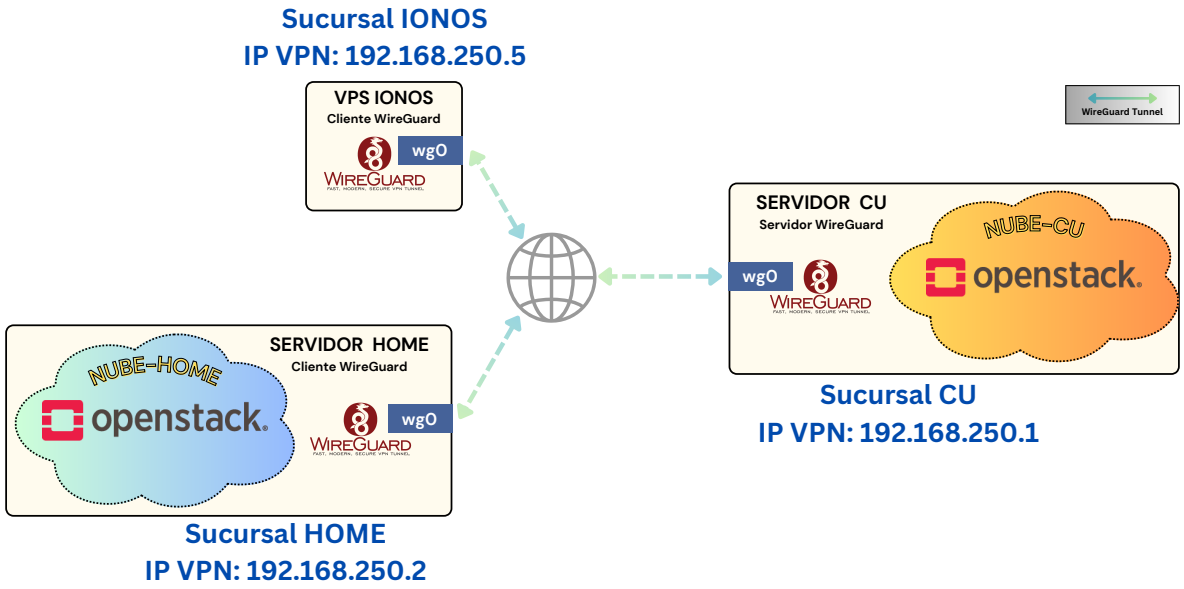

Figura 5.2: Topología de la VPN.

La topología de la figura 5.2 indica la asignación de *IP* para la *VPN* y también recalca lo mencionado en la sección 3.1, es decir, la sucursal CU es el servidor `WireGuard` y las otras sucursales son clientes.

El método de instalación y configuración de `WireGuard` en el servidor y en los clientes *VPN* se describe a continuación.

5.2. Servidor CU: servidor WireGuard

1. Se instala `WireGuard` utilizando el comando que se muestra en la consola 5.1.

Consola 5.1: Comando para instalar WireGuard.

```
# apt install wireguard -y
```

2. Como se observa en la consola 5.2, en el servidor CU se crea un directorio llamado `server` para guardar las claves del servidor *VPN*.

Consola 5.2: Comando para crear directorio server.

```
root@servidorCU:/etc/wireguard# mkdir server
```

3. Tal como se muestra en la consola 5.3, se crea una llave privada para el servidor *VPN* con su correspondiente llave pública y se modifican los permisos por defecto para que solo el usuario `root` tenga acceso a las claves.

Consola 5.3: Comando para generar llaves de servidor WireGuard.

```
root@servidorCU:/etc/wireguard/server# umask 077; wg genkey > \
   privatekey ; cat privatekey | wg pubkey > publickey
root@servidorCU:/etc/wireguard/server# ls -la
```

```
total 16
drwxr-xr-x 2 root root 4096 Oct 11 17:05 .
drwx------ 3 root root 4096 Oct 11 17:02 ..
-rw------- 1 root root   45 Oct 11 17:05 privatekey
-rw------- 1 root root   45 Oct 11 17:05 publickey
```

donde:

- `umask 077`: asigna permisos para que solo `root` tenga acceso a lectura/escritura/, por lo tanto, todos los demás usuarios no tienen permisos de acceso a los archivos.
- `privatekey`: llave privada codificada en `base64`.
- `publickey`: llave pública codificada en `base64`.

4. El archivo `/etc/sysctl.conf` se modifica según lo mostrado en la consola 5.4, con el fin de habilitar el *ip_forwarding* y permitir que el servidor `WireGuard` pueda reenviar paquetes.

Consola 5.4: Archivo /etc/sysctl.conf.

```
net.ipv4.ip_forward=1
```

Esta modificación permite que el servidor `WireGuard` encamine todo el tráfico desde el `WireGuard Peer` a través de la dirección *IP* del servidor *VPN*, con el fin de que la dirección *IP* pública de los cliente se oculte.

5. Se carga la nueva configuración con el comando de la consola 5.5.

Consola 5.5: Comando para cargan la nueva configuracion de /etc/sysctl.conf.

```
root@servidorCU:~#  sysctl -p
net.ipv4.ip_forward = 1
```

6. Conforme se evidencia en la consola 5.6, para configurar el servidor *VPN* se crea un nuevo archivo llamado `/etc/wireguard/wg0.conf` y se restringen los permisos para que solo `root` pueda tener acceso.

Consola 5.6: Comando para crear archivo wg0.conf y restringir permisos en el servidor CU.

```
root@servidorCU:/etc/wireguard# umask 077; touch /etc/wireguard/wg0.\
   conf
```

7. El contenido del archivo `/etc/wireguard/wg0.conf` se muestra en la consola 5.7.

Consola 5.7: Archivo wg0.conf del servidor VPN.

```
1  [Interface]
2  Address = 192.168.250.1/24
3  SaveConfig = true
4  PostUp = iptables -A FORWARD -i wg0 -j ACCEPT
5  PostUp = iptables -t nat -I POSTROUTING -o br0 -j MASQUERADE
6  PreDown = iptables -D FORWARD -i wg0 -j ACCEPT
7  PreDown = iptables -t nat -D POSTROUTING -o br0 -j MASQUERADE
8  ListenPort = 7777
9  PrivateKey = +OeEEW6jdixcyb3Am3zkS07f6IGFL5n8/c0e7wnuLGA=
10
11 [Peer]
12 PublicKey = xubnB23fpy/dH9yiuzB92iq+mLb91dLpTtdtn3UvbA4=
13 AllowedIPs = 192.168.250.2/32
14
```

```
[Peer]
PublicKey = dH6Oy/DhsYXFcrRH1lfgneuhJJxoXTMpTdaq6v61zlU=
AllowedIPs = 192.168.250.5/32
```

Los parámetro de `wg0.conf` se describen a continuación:

a) Sección `[Interface]`
- `Address`: dirección *IP* para la interfaz del túnel (`wg0`) del servidor *VPN*.
- `PostUp`: reglas que se ejecutan cuando el servidor `WireGuard` inicia el túnel.
  - `iptables -A FORWARD -i wg0 -j ACCEPT`: acepta todos los paquetes de la interfaz del túnel (`wg0`).
  - `iptables -t nat -I POSTROUTING -o br0 -j MASQUERADE`: configura el enmascaramiento y reescribe el tráfico IPv4 que ingresa a la interfaz `wg0` con el fin de que parezca que se origina directamente desde la dirección IPv4 pública del servidor `WireGuard`.
- `PreDown`: las reglas se aplican cuando se detiene el túnel y cumple la función de deshacer las reglas de `PostUp`.
- `ListenPort`: puerto que utiliza el servidor *VPN* para escuchar las conexiones entrantes.
- `PrivateKey`: llave privada generada en la consola 5.3.

b) Sección `[Interface]`: este apartado describe la configuración de los clientes `WireGuard`:
- `PublicKey`: la llave pública del cliente correspondiente.
- `AllowedIPs`: las direcciones *IP* del túnel del cliente que puede enviar datos al servidor `WireGuard`.

8. Se habilita e inicia la conexión `WireGuard` con el comando de la consola 5.8.

Consola 5.8: Inicio de la conexión WireGuard desde el servidor CU.

```
root@servidorCU:~# wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 192.168.250.1/24 dev wg0
[#] ip link set mtu 1420 up dev wg0
[#] iptables -A FORWARD -i wg0 -j ACCEPT
[#] iptables -t nat -I POSTROUTING -o br0 -j MASQUERADE
```

5.3. Servidor HOME: cliente WireGuard

La forma de crear un cliente WireGuard es similar a la del servidor *VPN*, sin embargo, existen cierta diferencias significativas que requieren una explicación detallada.

1. Se instala `WireGuard` con el comando de la consola 5.1.
2. Tal como se muestra en la consola 5.9, en el servidor HOME se crea un directorio llamado `client` y dentro se genera una llave privada, para el cliente *VPN*, con su correspondiente llave pública y se modifican los permisos para que solo `root` tenga acceso a las llaves.

Consola 5.9: Comando para generar llaves del cliente WireGuard del servidor HOME.

```
root@servidorHOME:~# mkdir /etc/wireguard/client
root@servidorHOME:~#/etc/wireguard/client# umask 077; wg genkey > \
privatekey ; cat privatekey | wg pubkey > publickey
root@servidorHOME:~#/etc/wireguard/client# ls -la
total 16
drwxr-xr-x 2 root root 4096 Oct 11 17:05 .
```

```
drwx------ 3 root root 4096 Oct 11 17:02 ..
-rw------- 1 root root 45 Oct 11 17:05 privatekey
-rw------- 1 root root 45 Oct 11 17:05 publickey
```

3. Conforme se evidencia en la consola 5.10, para configurar el cliente *VPN* se crea un nuevo archivo llamado `/etc/wireguard/wg0.conf` y se limitan los permisos:

Consola 5.10: Comando para crear archivo wg0.conf y restringir permisos en el servidor HOME.

```
root@servidorHOME:/etc/wireguard# umask 077; touch /etc/wireguard/wg0\
    .conf
```

4. El contenido del archivo `/etc/wireguard/wg0.conf` se muestra en la consola 5.11.

Consola 5.11: Archivo wg0.conf del cliente VPN de la sucursal HOME.

```
1  [Interface]
2  Address = 192.168.250.2/24
3  SaveConfig = true
4  PostUp = ip rule add table 200 from 192.168.0.65
5  PostUp = ip route add table 200 default via 192.168.0.1
6  PreDown = ip rule delete table 200 from 192.168.0.65
7  PreDown = ip route delete table 200 default via 192.168.0.1
8  PrivateKey = OFI0SIdqUxvmwy/e1REtbcixKp72aVXQopl0+6tZZVI=
9
10 [Peer]
11 PublicKey = 5OnkpmMz6S5sHE9W6VpIqLHA+SABnekYnfvXuB4aHBI=
12 AllowedIPs = 0.0.0.0/0
13 Endpoint = 132.248.59.195:7777
14 PersistentKeepalive = 25
```

Los parámetro de `wg0.conf` se describen a continuación:

a) Sección `[Interface]`

- `Address`: dirección *IP* para la interfaz del túnel (`wg0`) del cliente *VPN* de la sucursal HOME.
- `PostUp`: reglas que se ejecutan cuando el cliente `WireGuard` inicia el túnel. El objetivo de estas reglas es garantizar que se pueda acceder al servidor HOME desde afuera cuando esté activo el túnel. De lo contrario, cuando se establezca el túnel, el *router* TELMEX que le otorga visibilidad por medio de la *IP* Dinámica mediante `Port Mapping` no podrá acceder el servidor HOME.
  - `ip rule add table 200 from 192.168.0.65`: esta regla verifica si hay entradas de encaminamiento en la tabla 200 cuando la *IP* coincide con la dirección 192.168.0.65 (*IP* privada del servidor HOME). El número de tabla fue eligió de manera arbitrario.
  - `ip route add table 200 default via 192.168.0.1`: cualquier tráfico procesado por la tabla 200 utilizará la *IP* 192.168.0.1 como *gateway* en lugar de la interfaz `WireGuard`.
- `PreDown`: las reglas se aplican cuando se detiene el túnel y cumple la función de deshacer las reglas de `PostUp`.
- `PrivateKey`: llave privada del cliente *VPN* generada en la consola 5.9.

b) Sección `[Peer]`

- `PublicKey`: llave pública del servidor *VPN* generada en la consola 5.3.
- `AllowedIPs`: el parámetro `0.0.0.0/0` permite que cualquier dirección IPv4 se comunique con este cliente. Se utiliza este parámetro para que el cliente encamine todo el tráfico a través del túnel `WireGuard`, es decir, se utiliza el servidor *VPN* como el *gateway* predeterminado.

- `Endpoint`: indica la *IP* pública del servidor *VPN* y el puerto de `WireGuard` que se definió en el parámetro `ListenPort` del archivo `wg0.conf` (ver consola 5.7) del servidor `Wireguard`.
- `PersistentKeepalive`: intervalo en segundos en el que `WireGuard` envía un paquete *keepalive* al servidor *VPN*.

5. De la misma manera que en el servidor CU, se habilita la conexión `WireGuard` con el comando de la consola 5.12.

Consola 5.12: Inicio de la conexión WireGuard desde el servidor HOME.

```
root@servidorHOME:~# wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 192.168.250.2/24 dev wg0
[#] ip link set mtu 1420 up dev wg0
[#] ip -4 route add 0.0.0.0/0 dev wg0 table 51820
[#] ip -4 rule add not fwmark 51820 table 51820
[#] ip -4 rule add table main suppress_prefixlength 0
[#] sysctl -q net.ipv4.conf.all.src_valid_mark=1
[#] nft -f /dev/fd/63
[#] ip rule add table 200 from 192.168.0.65
[#] ip route add table 200 default via 192.168.0.1
```

5.4. VPS IONOS: cliente WireGuard

La configuración inicial de `WireGuard` para VPS IONOS sigue el mismo procedimiento que en el servidor HOME. Primero, se instala `WireGuard` utilizando el comando de la consola 5.1. Luego, se generan tanto la clave pública como la privada de manera similar a lo ilustrado en la consola 5.9. Finalmente, se crea el archivo de configuración `wg0.conf` siguiendo el procedimiento mostrado en la consola 5.10.

El archivo `/etc/wireguard/wg0.conf` (ver consola 5.13) contiene la configuración para la VPS IONOS.

Consola 5.13: Archivo wg0.conf del cliente VPN de la sucursal IONOS.

```
1  [Interface]
2  Address = 192.168.250.5/24
3  SaveConfig = true
4  PostUp = ip rule add table 200 from 82.165.209.114
5  PostUp = ip route add table 200 default via 82.165.209.1
6  PreDown = ip rule delete table 200 from 82.165.209.114
7  PreDown = ip route delete table 200 default via 82.165.209.1
8  PrivateKey = UOGE5N4zHJ/DxKfobaYixayq/TskV86bE3dDa7qsU3A=
9
10 [Peer]
11 PublicKey = 5OnkpmMz6S5sHE9W6VpIqLHA+SABnekYnfvXuB4aHBI=
12 AllowedIPs = 0.0.0.0/0
13 Endpoint = 132.248.59.195:7777
14 PersistentKeepalive = 25
```

Los parámetro de `wg0.conf`, tal como se muestran en la consola 5.13, se describen a continuación:

1. Sección `[Interface]`

   - `Address`: dirección *IP* para la interfaz del túnel (`wg0`) del cliente *VPN* de la sucursal IONOS.

- `PostUp`: reglas que se ejecutan cuando el cliente `WireGuard` inicia el túnel. El objetivo de estas reglas es garantizar que se pueda acceder al servidor IONOS desde afuera cuando esté activo el túnel.
  - `ip rule add table 200 from 82.165.209.114`: esta regla verifica si hay entradas de encaminamiento en la tabla 200 cuando la *IP* coincide con la dirección 82.165.209.114 (*IP* privada del servidor IONOS). El número de tabla fue eligió de manera arbitrario.
  - `ip route add table 200 default via 82.165.209.1`: cualquier tráfico procesado por la tabla 200 utilizará la *IP* 82.165.209.1 como *gateway* en lugar de la interfaz `WireGuard`.
- `PreDown`: las reglas se aplican cuando se detiene el túnel y cumple la función de deshacer las reglas de `PostUp`.
- `PrivateKey`: llave privada del cliente *VPN* de la sucursal IONOS.

2. Sección `[Peer]`
   - `PublicKey`: llave pública del servidor *VPN* generada mediante el comando de la consola 5.3.
   - `AllowedIPs`: el parámetro `0.0.0.0/0` permite que cualquier dirección IPv4 se comunique con este cliente. Se utiliza dicho parámetro para que el cliente encamine todo el tráfico a través del túnel `WireGuard`, es decir, se utiliza el servidor *VPN* como el *gateway* predeterminado.
   - `Endpoint`: indica la *IP* pública del servidor *VPN* y el puerto de `WireGuard` que se definió en el parámetro `ListenPort` del archivo `wg0.conf` (ver consola 5.7) del servidor `Wireguard`.
   - `PersistentKeepalive`: intervalo en segundos en el que `WireGuard` envía un paquete *keepalive* al servidor *VPN*.

Se habilita la conexión de `WireGuard` con el comando de la consola 5.14.

Consola 5.14: Inicio de la conexión WireGuard desde el VPS IONOS.

```
root@vpsIONOS:~#  wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 192.168.250.5/24 dev wg0
[#] ip link set mtu 1420 up dev wg0
[#] wg set wg0 fwmark 51820
[#] ip -4 route add 0.0.0.0/0 dev wg0 table 51820
[#] ip -4 rule add not fwmark 51820 table 51820
[#] ip -4 rule add table main suppress_prefixlength 0
[#] sysctl -q net.ipv4.conf.all.src_valid_mark=1
[#] iptables-restore -n
[#] ip rule add table 200 from 82.165.209.114
[#] ip route add table 200 default via 82.165.209.1
```

5.5. Verificación del estatus de WireGuard

Se puede monitorizar el estatus de la *VPN* con el comando que se muestra en la consola 5.15.

Consola 5.15: Comando para verificar del estatus de WireGuard en el servidor VPN.

```
root@servidorCU:~# wg
interface: wg0
```

```
  public key: 5OnkpmMz6S5sHE9W6VpIqLHA+SABnekYnfvXuB4aHBI=
  private key: (hidden)
  listening port: 7777

peer: dH6Oy/DhsYXFcrRH1lfgneuhJJxoXTMpTdaq6v61zlU=
  endpoint: 82.165.209.114:7777
  allowed ips: 192.168.250.5/32
  latest handshake: 15 seconds ago
  transfer: 6.75 KiB received, 2.33 KiB sent

peer: xubnB23fpy/dH9yiuzB92iq+mLb91dLpTtdtn3UvbA4=
  endpoint: 187.159.244.93:7777
  allowed ips: 192.168.250.2/32
  latest handshake: 1 minute, 14 seconds ago
  transfer: 3.24 KiB received, 620 B sent
```

Capítulo 6

Evaluación de la red

En este capítulo se realiza una evaluación de las funcionalidades de la red V-WAN planteadas en la sección 3.1.

6.1. Función A

En la sección 3.1, se mencionó que la función A tiene como objetivo que: "*todas las máquinas virtuales de NUBE-HOME (consultar figura 3.2) se conecten a las máquinas virtuales de NUBE-CU (consultar figura 3.3), utilizando un túnel VPN. De forma similar, el VPS IONOS establece conexión con las máquinas virtuales de la NUBE-CU (ver figura 3.3) mediante un túnel* WireGuard."

Para validar el funcionamiento de la NUBE-HOME, se emplea la máquina virtual HOME-pc1 como agente representativo. Por otro lado, en la NUBE-CU se toma como referencia a CU-pc1. Es importante destacar que, dado que la configuración de red de todas las máquinas es similar, se espera un comportamiento homogéneo. Por lo tanto, las capacidades de red de HOME-pc1 son comparables a las de cualquier otra máquina virtual de NUBE-HOME, y lo mismo ocurre con CU-pc1 en NUBE-CU.

Se proponen los siguientes casos para analizar el comportamiento de la función A:

1. Conexión entre HOME-pc1 y CU-pc1:

   En la tabla 6.1 de muestra la ubicación y dirección IP de HOME-pc1 y de CU-pc1.

HOME-pc1	CU-pc1
▪ Ubicación: NUBE-HOME ▪ Dirección IP: 192.168.2.146	▪ Ubicación: NUBE-CU ▪ Dirección IP: 192.168.10.39

Tabla 6.1: Características de HOME-pc1 y CU-pc1.

La consola 6.1 muestra la conectividad entre la máquina HOME-pc1 y la máquina CU-pc1 de la NUBE-CU.

Consola 6.1: Ping desde HOME-pc1 a CU-pc1.

```
root@HOME-pc1:~# ping -c 5 192.168.10.39
PING 192.168.10.39 (192.168.10.39) 56(84) bytes of data.
64 bytes from 192.168.10.39: icmp_seq=1 ttl=61 time=33.4 ms
64 bytes from 192.168.10.39: icmp_seq=2 ttl=61 time=28.9 ms
64 bytes from 192.168.10.39: icmp_seq=3 ttl=61 time=27.4 ms
64 bytes from 192.168.10.39: icmp_seq=4 ttl=61 time=30.0 ms
64 bytes from 192.168.10.39: icmp_seq=5 ttl=61 time=28.5 ms
```

```
--- 192.168.10.39 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 27.411/29.645/33.355/2.039 ms
```

Del mismo modo, en la consola 6.2, se muestra que desde HOME-pc1 es posible conectarse a CU-pc1 utilizando el protocolo *ssh*.

Consola 6.2: Conexión desde HOME-pc1 hacia CU-pc1 utilizando SSH.

```
root@HOME-pc1:~# ssh debian@192.168.10.39
debian@192.168.10.39 s password:
Linux cu-pc1-debian 6.1.0-17-amd64 #1 SMP PREEMPT_DYNAMIC Debian \
    6.1.69-1 (2023-12-30) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Feb 15 01:01:23 2024 from 192.168.10.2
debian@CU-pc1:~$
```

La figura 6.1 muestra el enlace de conexión entre la máquina HOME-pc1 y CU-pc1.

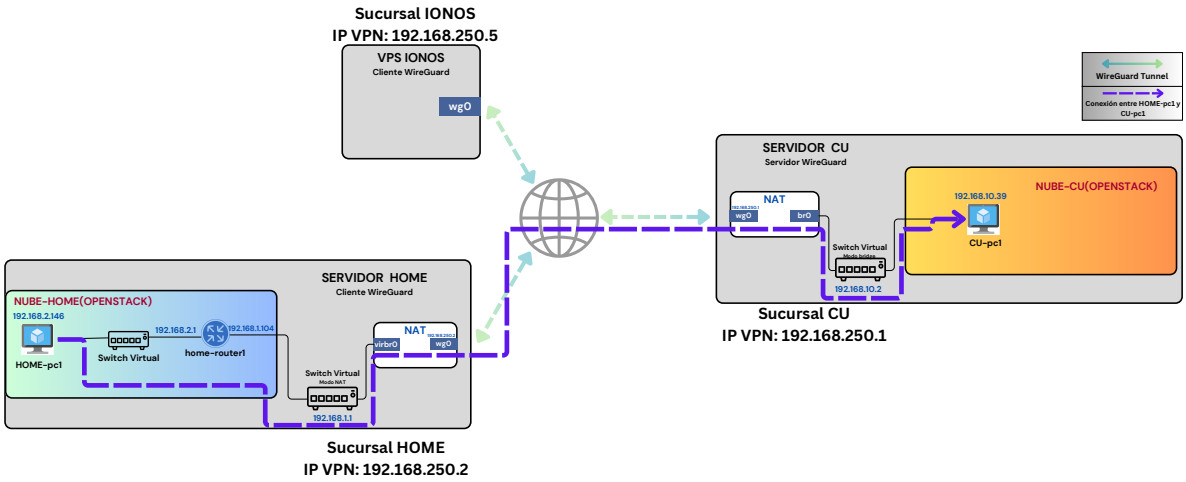

Figura 6.1: Conexión entre HOME-pc1 y CU-pc1.

Con el fin de explicar el flujo de datos de la figura 6.1 se utiliza la herramienta `tcpdump` para analizar el tráfico de la tarjeta de red `virbr0` y `wg0` del servidor HOME, así como las tarjetas `br0` y `wg0` del servidor CU, y la tarjeta `ens3` de la máquina CU-pc1. La prueba de conectividad consiste de enviar constantemente paquetes ICMP desde el HOME-pc1 (192.168.2.146) hacia CU-pc1 (192.168.10.39). Los resultados se muestran a continuación:

a) virbr0 del servidor HOME

En la consola 6.3 se ve el registro del comando `ping` proveniente de la *IP* del *router* home-router1. Por lo tanto, se comprueba que el home-router1 enmascara el paquete y lo reenvía a la interfaz virtual `virbr0` del servidor HOME.

Consola 6.3: Comando *tcpdump -i wirbr0 icmp* desde el servidor HOME.

```
root@servidorHOME:~# tcpdump -i virbr0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on virbr0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
19:49:01.669300 IP 192.168.1.104 > 192.168.10.39: ICMP echo request, id 39468, seq 1429, length\
     64
19:49:01.693912 IP 192.168.10.39 > 192.168.1.104: ICMP echo reply, id 39468, seq 1429, length \
    64
19:49:02.670984 IP 192.168.1.104 > 192.168.10.39: ICMP echo request, id 39468, seq 1430, length\
     64
19:49:02.694436 IP 192.168.10.39 > 192.168.1.104: ICMP echo reply, id 39468, seq 1430, length \
    64
```

b) wg0 del servidor HOME

Entre las tarjeta de red `virbr0` y `wg0` existe una NAT, por ello, en la consola 6.4 se observa que los paquete ICMP ya tiene como fuente la *IP VPN* 192.168.250.2.

Consola 6.4: Comando *tcpdump -i wg0 icmp* desde el servidor HOME.

```
root@servidorHOME:~# tcpdump -i wg0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on wg0, link-type RAW (Raw IP), snapshot length 262144 bytes
19:19:42.570920 IP 192.168.250.2 > 192.168.10.39: ICMP echo request, id 21952, seq \
    37, length 64
19:19:42.600909 IP 192.168.10.39 > 192.168.250.2: ICMP echo reply, id 21952, seq \
    37, length 64
19:19:43.572078 IP 192.168.250.2 > 192.168.10.39: ICMP echo request, id 21952, seq \
    38, length 64
19:19:43.604352 IP 192.168.10.39 > 192.168.250.2: ICMP echo reply, id 21952, seq \
    38, length 64
```

c) wg0 del servidor CU

En la tarjeta de red `wg0` los paquetes llevan la misma *IP* destino y origen, sin embargo, ahora están del otro lado de la *VPN*, por tanto se puede confirmar que los paquetes están llegando correctamente a través de la *VPN* (ver consola 6.5).

Consola 6.5: Comando *tcpdump -i wg0 icmp* desde el servidor CU para HOME-pc1.

```
root@servidorCU:~# tcpdump -i wg0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on wg0, link-type RAW (Raw IP), snapshot length 262144 bytes
21:35:06.522734 IP 192.168.250.2 > 192.168.10.39: ICMP echo request, id 36386, seq \
    24, length 64
21:35:06.523228 IP 192.168.10.39 > 192.168.250.2: ICMP echo reply, id 36386, seq \
    24, length 64
21:35:07.523463 IP 192.168.250.2 > 192.168.10.39: ICMP echo request, id 36386, seq \
    25, length 64
21:35:07.527277 IP 192.168.10.39 > 192.168.250.2: ICMP echo reply, id 36386, seq \
    25, length 64
```

d) br0 del servidor CU

Entre las tarjetas de red `wg0` y `br0` existe una NAT, por tanto los paquetes ICMP tienen como fuente la *IP* del servidor CU, esto se puede ver en la consola 6.6.

Consola 6.6: Comando *tcpdump -i br0 icmp* desde el servidor CU para HOME-pc1.

```
root@servidorCU:~# tcpdump -i br0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on br0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
21:35:57.619495 IP servidor > 192.168.10.39: ICMP echo request, id 36386, seq 75, \
    length 64
21:35:57.623421 IP 192.168.10.39 > servidor: ICMP echo reply, id 36386, seq 75, \
    length 64
```

```
21:35:58.619297 IP servidor > 192.168.10.39: ICMP echo request, id 36386, seq 76, \
    length 64
21:35:58.619675 IP 192.168.10.39 > servidor: ICMP echo reply, id 36386, seq 76, \
    length 64
```

e) ens3 de CU-pc1

Finalmente, en la consola 6.7 se muestra como los paquetes ICMP llegan a su destino y tiene como *IP* origen la *IP* del servidor NAT.

Consola 6.7: Comando *tcpdump -i br0 icmp* desde CU-pc1 para HOME-pc1.

```
root@servidorCU:~# tcpdump -i br0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on br0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
21:35:57.619495 IP servidor > 192.168.10.39: ICMP echo request, id 36386, seq 75, \
    length 64
21:35:57.623421 IP 192.168.10.39 > servidor: ICMP echo reply, id 36386, seq 75, \
    length 64
21:35:58.619297 IP servidor > 192.168.10.39: ICMP echo request, id 36386, seq 76, \
    length 64
21:35:58.619675 IP 192.168.10.39 > servidor: ICMP echo reply, id 36386, seq 76, \
    length 64
```

2. Conexión entre el VPS IONOS y CU-pc1:

En la tabla 6.2 de muestra la ubicación y dirección IP del VPS IONOS y de CU-pc1.

VPS-IONOS	CU-pc1
▪ Ubicación: Sucursal IONOS ▪ Dirección IP: 192.168.250.5	▪ Ubicación: NUBE-CU ▪ Dirección IP: 192.168.10.39

Tabla 6.2: Características de VPS-IONOS y CU-pc1.

Análogo al proceso para HOME-pc1, se demuestra el flujo de datos del VPS-IONOS con la máquina virtual CU-pc1 de la NUBE-CU utilizando el comando `ping`:

Consola 6.8: Ping desde VPS-IONOS hacia CU-pc1.

```
root@vpsIONOS:~# ping -c 4 192.168.10.39
PING 192.168.10.39 (192.168.10.39) 56(84) bytes of data.
64 bytes from 192.168.10.39: icmp_seq=1 ttl=63 time=66.5 ms
64 bytes from 192.168.10.39: icmp_seq=2 ttl=63 time=65.8 ms
64 bytes from 192.168.10.39: icmp_seq=3 ttl=63 time=65.9 ms
64 bytes from 192.168.10.39: icmp_seq=4 ttl=63 time=65.9 ms

--- 192.168.10.39 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 65.837/66.034/66.466/0.252 mss
```

En al consola 6.8 se pude observar que VPS-IONOS alcanza a la máquina CU-pc1 de la NUBE-CU.

En la consola 6.9 se puede ver que VPS-IONOS puede conectarse a CU-pc1 utilizando el protocolo *ssh*.

Consola 6.9: Conexión desde el VPS-IONO hacia CU-pc1 utilizando SSH.

```
root@HOME-pc1:~# ssh debian@192.168.10.39
debian@192.168.10.39 s password:
```

```
Linux cu-pc1-debian 6.1.0-17-amd64 #1 SMP PREEMPT_DYNAMIC Debian \
    6.1.69-1 (2023-12-30) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Feb 15 01:01:23 2024 from 192.168.10.2
debian@CU-pc1:~$
```

El comportamiento de la conexión entre el VPS-IONOS y CU-pc1 se representa gráficamente en la figura 6.2.

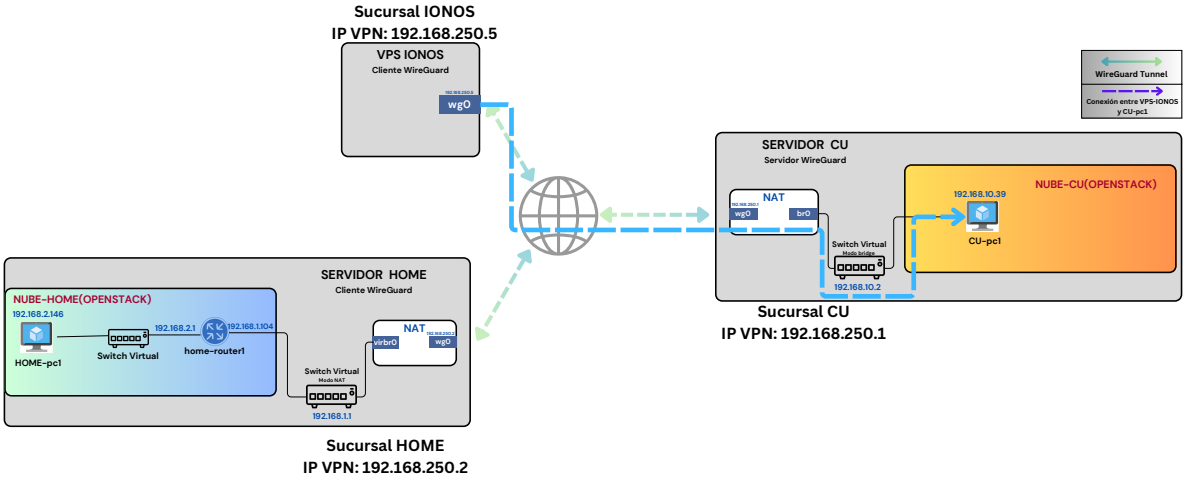

Figura 6.2: Conexión entre VPS-IONOS y CU-pc1.

Con el fin de explicar el flujo de datos, mostrado en la figura 6.2, entre el VPS-IONOS (192.168.250.5) y la máquina CU-pc1 (192.168.10.39), se envía un flujo de paquetes `ICMP` entre ellas. El resultado del tráfico es analizado con el comando `tcpdump`.

a) wg0 del VPS IONOS

La tarjeta de red `wg0` del VPS IONOS crea los paquetes con destinó CU-pc1 (192.168.10.39) (ver la consola 6.10).

Consola 6.10: Comando *tcpdump -i wg0 icmp* desde VPS-IONOS.

```
root@vpsIONOS:~# tcpdump -i wg0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on wg0, link-type RAW (Raw IP), snapshot length 262144 bytes
20:31:43.702196 IP 192.168.250.5 > 192.168.10.39: ICMP echo request, id 59212, seq 74, length \
    64
20:31:43.767940 IP 192.168.10.39 > 192.168.250.5: ICMP echo reply, id 59212, seq 74, length 64
20:31:44.703269 IP 192.168.250.5 > 192.168.10.39: ICMP echo request, id 59212, seq 75, length \
    64
20:31:44.769157 IP 192.168.10.39 > 192.168.250.5: ICMP echo reply, id 59212, seq 75, length 64
```

b) wg0 del servidor CU

La consola 6.11 muestra la tarjeta de red `wg0` del servidor CU, la cual conserva la *IP* origen y destino mostrados en la consola 6.10. Sin embargo, en este momento se encuentran del lado del servidor CU, lo que garantiza que viajaron correctamente sobre la *VPN*.

Consola 6.11: Comando *tcpdump -i wg0 icmp* desde servidor CU para VPS-IONOS.

```
root@servidorCU:~# tcpdump -i wg0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on wg0, link-type RAW (Raw IP), snapshot length 262144 bytes
15:09:06.627711 IP 192.168.250.5 > 192.168.10.39: ICMP echo request, id 59212, seq 2314, length\
     64
15:09:06.628645 IP 192.168.10.39 > 192.168.250.5: ICMP echo reply, id 59212, seq 2314, length \
    64
15:09:07.629571 IP 192.168.250.5 > 192.168.10.39: ICMP echo request, id 59212, seq 2315, length\
     64
15:09:07.630503 IP 192.168.10.39 > 192.168.250.5: ICMP echo reply, id 59212, seq 2315, length \
    64
```

c) br0 del servidor CU

Entre las tarjeta de red `wg0` y `br0` existe una NAT. La consola 6.12 muestra que los paquete ICMP tiene como fuente la *IP* del servidor CU.

Consola 6.12: Comando *tcpdump -i br0 icmp* desde servidor CU para VPS-IONOS.

```
root@servidorCU:~# tcpdump -i br0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on br0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
21:35:57.619495 IP servidor > 192.168.10.39: ICMP echo request, id 36386, seq 75, length 64
21:35:57.623421 IP 192.168.10.39 > servidor: ICMP echo reply, id 36386, seq 75, length 64
21:35:58.619297 IP servidor > 192.168.10.39: ICMP echo request, id 36386, seq 76, length 64
21:35:58.619675 IP 192.168.10.39 > servidor: ICMP echo reply, id 36386, seq 76, length 64
```

d) ens3 de CU-pc1

Finalmente, la consola 6.13 muestra como los paquetes ICMP llegan a su destino.

Consola 6.13: Comando *tcpdump -i ens3 icmp* desde CU-pc1 para VPS-IONOS.

```
root@CU-pc1:~# tcpdump -i ens3 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
21:37:05.759091 IP 192.168.10.2 > cu-pc1-debian: ICMP echo request, id 59212, seq \
    3991, length 64
21:37:05.759134 IP cu-pc1-debian > 192.168.10.2: ICMP echo reply, id 59212, seq \
    3991, length 64
21:37:06.760065 IP 192.168.10.2 > cu-pc1-debian: ICMP echo request, id 59212, seq \
    3992, length 64
21:37:06.760123 IP cu-pc1-debian > 192.168.10.2: ICMP echo reply, id 59212, seq \
    3992, length 64
```

6.2. Función B

En la sección 3.1, se mencionó que la función B tiene como objetivo que: *"las máquinas virtuales de la NUBE-HOME (consultar figura 3.2) tienen salida a Internet a través del servidor CU (servidor* `WireGuard`*). Igualmente, el VPS IONOS cuenta con acceso a Internet a través del servidor CU (servidor* `WireGuard`*)."*

Para demostrar que la función B opera correctamente, se utiliza el comando `curl` en las máquinas HOME-pc1 y en el VPS-IONOS para obtener el código de la página web *ipinfo.io*. Para la *VPN* activa como sin la VPN, se verifica con qué dirección *IP* pública están accediendo a Internet. Además, con la VPN activa, se utiliza el comando `mtr` para verificar que el tráfico proveniente de las máquinas de la NUBE-CU y el VPS-IONOS salte primero a la sucursal CU antes de salir a Internet.

1. HOME-pc1:

Al igual que la Función A, se toma como referencia la máquina virtual HOME-pc1 de la NUBE-HOME. Se desactiva la *VPN* y se ejecuta el comando `curl` para reproducir el código de la página web *ipinfo.io* como se muestra a continuación:

Consola 6.14: Comando curl ipinfo.io sin VPN activa en HOME-pc1.

```
root@HOME-pc1:~$ curl ipinfo.io
{
  "ip": "189.149.12.31",
  "hostname": "dsl-189-149-12-31-dyn.prod-infinitum.com.mx",
  "city": "Ciudad Nezahualcoyotl",
  "region": "Mexico",
  "country": "MX",
  "loc": "19.4006,-99.0148",
  "org": "AS8151 UNINET",
  "postal": "57700",
  "timezone": "America/Mexico_City",
  "readme": "https://ipinfo.io/missingauth"
```

En la consola 6.14 se muestra que sin la *VPN*, el tráfico de datos sale directamente por la infraestructura del ISP de la sucursal HOME, y por ello se ve la *IP* pública de HOME-pc1 (189.149.12.31) y los datos que se describen son del ISP.

Posteriormente se activa la *VPN* y se vuelve a ejecuta el comando `curl`:

Consola 6.15: Comando curl ipinfo.io con VPN activa en HOME-pc1.

```
root@HOME-pc1:~$ curl ipinfo.io
{
  "ip": "132.248.59.195",
  "city": "Mexico City",
  "region": "Mexico City",
  "country": "MX",
  "loc": "19.4285,-99.1277",
  "org": "AS278 Universidad Nacional Autonoma de Mexico",
  "postal": "03020",
  "timezone": "America/Mexico_City",
  "readme": "https://ipinfo.io/missingauth"
```

La consola 6.15 muestra que con la *VPN* activa la *IP* pública para HOME-pc1 es la *IP* 132.248.59.195 de la sucursal CU en lugar de la *IP* de la sucursal HOME como se había mostrado en la consola 6.14.

Por último, se utiliza el comando `mtr` para imprimir la ruta y todos lo saltos que realiza un paquete. El parámetro `-4b` fuerza a utilizar IPv4 y a mostrar tanto los nombres de *host* como las direcciones *IP*. La consola 6.16 muestra cómo se usa el comando `mtr` para llegar al host de Google (142.251.218.142) desde HOME-pc1 y también imprime detalles sobre todos los saltos que visita en el medio:

Consola 6.16: Comando mtr sobre HOME-pc1 hacia google.com.

```
root@HOME-pc1:~#  mtr -4b google.com
                                 My traceroute  [v0.95]
HOME-pc1 (192.168.2.146) -> google.com (142.251.34.46)                2024-02-28T02:40:26+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                                      Packets               Pings
 Host                                                Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. _gateway (192.168.2.1)                            0.1%  1678    0.8   1.7   0.5  49.9   2.8
 2. 192.168.1.1 (192.168.1.1)                         0.0%  1678    1.0   2.0   0.6  52.0   1.9
 3. 192.168.250.1 (192.168.250.1)                     0.0%  1678    6.7   7.5   3.8  27.9   1.9
 4. 192.168.10.1 (192.168.10.1)                       0.0%  1678    5.5   7.5   3.9  45.5   2.4
 5. ve59.iimas-dist.unam.mx (132.248.59.254)          0.0%  1678    4.6  11.6   4.0  55.9   5.5
 6. 1010-iimas.redunam.unam.mx (132.247.237.101)      0.0%  1678    7.1  10.7   4.0  59.5   5.0
 7. 132.248.133.33 (132.248.133.33)                   0.0%  1678    5.9   7.9   4.1  39.3   2.1
 8. 132.247.254.30 (132.247.254.30)                   0.0%  1678    4.5   7.8   4.0  49.5   2.4
 9. fixed-186-96-10-169.totalplay.net (186.96.10.169)  2.3%  1677    7.7   8.8   4.9  47.2   2.5
```

```
10. 10.180.59.133 (10.180.59.133)                    0.0%  1677    7.0   9.1   5.3  46.9   2.5
11. 10.180.200.172 (10.180.200.172)                  0.1%  1677   12.4  15.9  12.0  56.8   2.5
12. 72.14.215.106 (72.14.215.106)                    0.0%  1677   16.3  16.5  12.6  54.1   2.3
13. 142.250.215.89 (142.250.215.89)                  0.0%  1677   17.0  19.0  14.6  94.9   5.8
14. 142.251.78.55 (142.251.78.55)                    0.0%  1677   13.5  16.3  12.5  56.9   2.6
15. qro01s28-in-f14.1e100.net (142.251.34.46)        0.0%  1677   27.0  27.4  23.7  68.1   2.9
```

El comportamiento de los paquetes mostrados en la consola 6.16 se representan gráficamente en la figura 6.3.

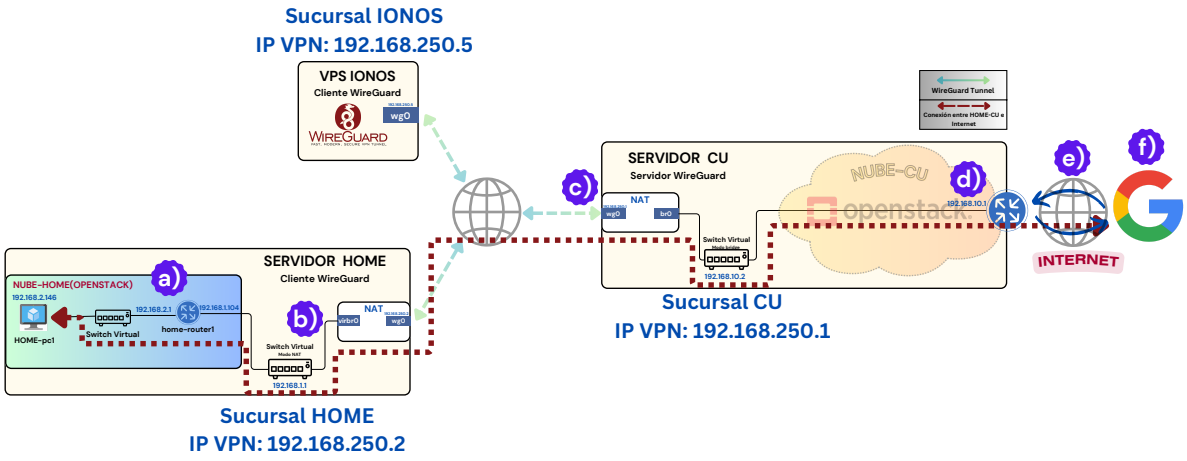

Figura 6.3: Comando mtr sobre HOME-pc1 hacia google.com.

A continuación se describe la ruta y los saltos mostrados en la consola 6.16 y en la figura 6.3:

a) Los paquetes llegan al router virtual (192.168.2.1) creado dentro del ambiente `OpenStack`.
b) El tráfico de datos atraviesa por el *switch* virtual (192.168.1.1) de la NAT.
c) Por medio de la *VPN* se conducen los paquetes y llega al servidor `WireGuard` (192.168.250.1).
d) El tráfico de datos pasa al *gateway* de la sucursal HOME (192.168.10.1).
e) Los paquetes salen a Internet a través infraestructura de la red del ISP de la sucursal CU.
f) Finalmente llegan al *host* de Google (142.251.218.142).

2. VPS-IONOS:

Se ejecuta el comando `curl`, se desactiva la *VPN* para reproducir el código de la página web ipinfo.io:

Consola 6.17: Comando curl ipinfo.io sin VPN activa en VPS-IONOS.

```
root@vpsIONOS:~# curl ipinfo.io
{
  "ip": "82.165.209.114",
  "hostname": "ip82-165-209-114.pbiaas.com",
  "city": "Piscataway",
  "region": "New Jersey",
  "country": "US",
```

```
  "loc": "40.4993,-74.3990",
  "org": "AS54548 IONOS Cloud Inc.",
  "postal": "08854",
  "timezone": "America/New_York",
  "readme": "https://ipinfo.io/missingauth"
}
```

La consola 6.17 muestra que sin la *VPN* los paquetes sale directamente por la infraestructura de la sucursal IONOS y debido a ello al utilizar el comando `curl` se imprime la *IP* pública (82.165.209.114) de la sucursal IONOS correspondiente a una región de Estados Unidos debido a que al momento de contratar el VPS se eligió como zona el centro de datos de IONOS en Estados Unidos.

A continuación se activa la *VPN* en la sucursal IONOS y se vuelve a ejecuta el comando `curl`:

Consola 6.18: Comando curl ipinfo.io con VPN activa en VPS-IONOS.

```
root@vpsIONOS:~# curl ipinfo.io
{
  "ip": "132.248.59.195",
  "city": "Mexico City",
  "region": "Mexico City",
  "country": "MX",
  "loc": "19.4285,-99.1277",
  "org": "AS278 Universidad Nacional Autonoma de Mexico",
  "postal": "03020",
  "timezone": "America/Mexico_City",
  "readme": "https://ipinfo.io/missingauth"
}
```

La consola 6.18 muestra que con la *VPN* activa, la *IP* pública del VPS-IONOS es ahora la *IP* (132.248.59.195) de la sucursal CU y debido a ello ahora el país correspondiente a la *IP* pública es México.

Finalmente, se utiliza el comando `mtr` para imprimir la ruta y todos lo saltos que realiza un paquete desde el VPS-IONOS hasta llegar al *host* de Google (142.251.49.213):

Consola 6.19: Comando mtr sobre VPS IONOS hacia google.com.

```
root@vpsIONOS:~# mtr -4b google.com
                                 My traceroute  [v0.94]
vpsIONOS (192.168.250.5) -> google.com                          2024-02-29T03:25:50+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                                    Packets               Pings
 Host                                             Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 192.168.250.1 (192.168.250.1)                  0.0%    34   64.8  65.0  64.8  65.9   0.2
 2. 192.168.10.1 (192.168.10.1)                    0.0%    34   64.9  65.1  64.9  65.4   0.1
 3. ve59.iimas-dist.unam.mx (132.248.59.254)       0.0%    34   65.5  65.9  65.2  69.9   1.2
 4. 1010-iimas.redunam.unam.mx (132.247.237.101)   0.0%    34   71.0  67.9  65.1  76.2   3.5
 5. 132.248.133.33 (132.248.133.33)                0.0%    34   65.2  65.4  65.2  65.8   0.1
 6. 132.247.254.30 (132.247.254.30)                0.0%    34   65.3  65.5  65.3  66.1   0.2
 7. fixed-186-96-10-169.totalplay.net (186.96.10.169)  0.0%    33   66.3  66.1  65.7  66.6   0.2
 8. 10.180.59.133 (10.180.59.133)                  0.0%    33   66.5  66.6  66.3  67.5   0.2
 9. 10.180.200.172 (10.180.200.172)                0.0%    33   73.2  73.5  73.2  74.8   0.3
10. 72.14.215.106 (72.14.215.106)                  0.0%    33   73.6  74.0  73.6  74.6   0.2
11. 74.125.243.33 (74.125.243.33)                  0.0%    33   85.3  86.1  85.3  86.9   0.3
12. 74.125.243.41 (74.125.243.41)                  0.0%    33   74.7  75.1  74.4  79.0   0.7
13. 142.251.77.252 (142.251.77.252)                0.0%    33  103.0 112.4 102.6 195.9  20.9
14. 108.170.231.7 (108.170.231.7)                  0.0%    33  101.6 101.6 101.2 102.2   0.2
15. 142.251.67.134 (142.251.67.134)                0.0%    33  123.3 124.3 121.9 145.5   4.4
16. 172.253.68.51 (172.253.68.51)                  0.0%    33  125.2 126.3 125.2 127.3   0.6
17. 192.178.81.146 (192.178.81.146)                0.0%    33  132.5 135.3 132.2 153.6   5.0
18. 142.251.226.100 (142.251.226.100)              0.0%    33  132.8 132.9 132.3 135.3   0.5
19. 142.251.49.213 (142.251.49.213)                0.0%    33  217.0 217.2 217.0 217.6   0.1
```

El comportamiento de los paquetes mostrados en la consola 6.19 se representa gráficamente en la siguiente figura:

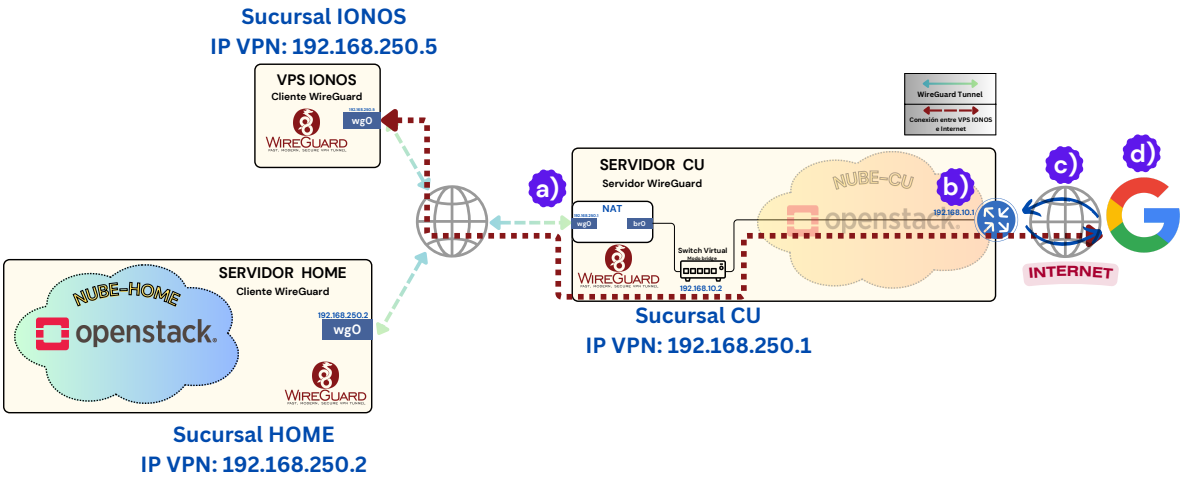

Figura 6.4: Comando mtr sobre VPS IONOS hacia google.com.

A continuación se describe la ruta y los saltos mostrados en la consola 6.19 y en la figura 6.4:

a) Los paquetes se conducen mediante la *VPN* y llega al servidor `WireGuard` (192.168.250.1).
b) El tráfico de datos pasa al *gateway* de la sucursal HOME (192.168.10.1).
c) Los paquetes salen a Internet a través de la infraestructura de red del ISP de la sucursal CU.
d) Finalmente llegan al *host* de Google (142.251.49.213).

6.3. Función C

En la sección 3.1, se mencionó que la función C tiene como objetivo que: *"la comunicación es bidireccional entre el servidor CU, el servidor HOME y el VPS-IONOS."* Para verificar la función C, se comprueba la conectividad (a nivel capa *IP*) bidireccionalmente mediante el comando `ping`. Se envían paquetes ICMP desde el servidor CU hacia el VPS-IONOS y hacia el servidor HOME. Asimismo, se realizan pruebas mediante el comando `ping` desde el VPS-IONOS hacia el servidor HOME y hacia el servidor CU.

La consola 6.20 muestra que existe conexión entre el servidor CU (192.168.250.1) con el servidor HOME (192.168.250.2) y VPS-IONOS (192.168.250.5).

Consola 6.20: Ping desde el servidor CU hacia el VPS IONOS y hacia la servidor HOME.

```
root@servidorCU:~# ping -c 3 192.168.250.2
PING 192.168.250.2 (192.168.250.2) 56(84) bytes of data.
64 bytes from 192.168.250.2: icmp_seq=1 ttl=64 time=4.29 ms
64 bytes from 192.168.250.2: icmp_seq=2 ttl=64 time=3.70 ms
64 bytes from 192.168.250.2: icmp_seq=3 ttl=64 time=6.39 ms

--- 192.168.250.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 3.703/4.792/6.385/1.151 ms
root@servidor:~# ping -c 3 192.168.250.5
```

```
PING 192.168.250.5 (192.168.250.5) 56(84) bytes of data.
64 bytes from 192.168.250.5: icmp_seq=1 ttl=64 time=64.9 ms
64 bytes from 192.168.250.5: icmp_seq=2 ttl=64 time=65.1 ms
64 bytes from 192.168.250.5: icmp_seq=3 ttl=64 time=65.4 ms

--- 192.168.250.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 64.904/65.123/65.358/0.185 ms
```

De la misma manera en el servidor CU se verifica la conexión del VPS-IONOS con el servidor CU y el servidor HOME como se muestra en la consola 6.21.

Consola 6.21: Ping desde VPS-IONOS hacia el servidor CU y el servidor HOME.

```
root@vpsIONOS:~# ping -c 3 192.168.250.1
PING 192.168.250.1 (192.168.250.1) 56(84) bytes of data.
64 bytes from 192.168.250.1: icmp_seq=1 ttl=64 time=64.7 ms
64 bytes from 192.168.250.1: icmp_seq=2 ttl=64 time=65.6 ms
64 bytes from 192.168.250.1: icmp_seq=3 ttl=64 time=64.8 ms

--- 192.168.250.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 64.749/65.019/65.552/0.376 ms
root@debian-ionos:~# ping -c 3 192.168.250.2
PING 192.168.250.2 (192.168.250.2) 56(84) bytes of data.
64 bytes from 192.168.250.2: icmp_seq=1 ttl=63 time=68.0 ms
64 bytes from 192.168.250.2: icmp_seq=2 ttl=63 time=71.4 ms
64 bytes from 192.168.250.2: icmp_seq=3 ttl=63 time=70.5 ms

--- 192.168.250.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 67.950/69.951/71.412/1.464 ms
```

6.4. Evaluación de rendimiento

Se realizó una prueba de rendimiento en el VPS-IONOS, el servidor HOME y la HOME-cp1 utilizando la herramienta `speedtest`, la cual permite evaluar la velocidad de descarga, carga y latencia de una conexión a Internet. Para llevar a cabo esta evaluación, se utilizó el comando de la consola 6.22, el cual ejecuta el comando cada 20 segundos durante una hora y registra los resultados en un archivo de texto.

Consola 6.22: Comando para evaluación de rendimiento.

```
root@servidorHOME:~#timeout 3600 watch -n 20 "speedtest-cli --simple >> speedtest-resul.txt"
```

Nota: el comando `watch` se utiliza para ejecutar el comando `speedtest-cli --simple` cada 20 s, mientras el comando `timeout` delimita el tiempo de la operación. Para filtrar los datos, se utiliza el comando `awk` mostrado en la consola 6.23, con el fin de dividir los valores de descarga, carga y latencia en tres archivos diferentes para su posterior análisis.

Consola 6.23: Comando para filtrar los datos de la evaluación de rendimiento.

```
root@servidorHOME:~# awk '/Ping:/ {print $2}' speedtest-resul.txt > sp-ping.txt
awk '/Download:/ {print $2}' speedtest-resul.txt > sp-down.txt
awk '/Upload:/ {print $2}' speedtest-resul.txt > sp-up.txt
```

Para obtener valores promedios, se desarrolló un *script* en *Python* (ver consola 6.24) que utiliza los datos obtenidos anteriormente y la biblioteca *pandas*. Específicamente, se emplea la función *mean()* para calcular el promedio y la desviación estándar mediante la función *std()*.

Consola 6.24: Script de python para calcular promedio y desviación estándar.

```
import pandas as pd

# Cargamos los datos del archivo txt en un DataFrame de Pandas
df = pd.read_csv('sp-ionos-ping.txt', header=None)

# Calculamos el promedio
promedio = df.mean()

# Calculamos la desviacion estandar
desviacion_estandar = df.std()

print("El promedio de los datos es:", promedio[0])  # Se imprime promedio
print("La desviacion estandar de los datos es:", desviacion_estandar[0])  # Se \
    imprime desviacion estandar
```

Los resultados obtenidos para cada equipo se muestran a continuación:

1. Servidor HOME:

   a) Sin *VPN*: en la consola 6.25 se visualiza un fragmento de la prueba de rendimiento (ver consola 6.22) del servidor HOME sin VPN activa.

Consola 6.25: Fragmento de la prueba de rendimiento del servidor HOME sin VPN activa.

```
root@servidorHOME:~# speedtest
Retrieving speedtest.net configuration...
Testing from Telmex (189.149.12.31)...
Retrieving speedtest.net server list...
Selecting best server based on ping...
Hosted by INFINITUM (Naucalpan) [19.09 km]: 7.061 ms
Testing download speed.................................
Download: 98.92 Mbit/s
Testing upload speed...................................
Upload: 97.54 Mbit/s
```

Los resultados del servidor HOME sin VPN activa fueron los siguientes:

1) Upload
   - El promedio de la velocidad de carga fue: 96.610 Mbps
   - La desviación estándar de la velocidad de carga fue: 2.54 Mbps
2) Download:
   - El promedio de la velocidad de descarga fue: 98.857 Mbps
   - La desviación estándar de la velocidad de descarga fue: 1.536 Mbps
3) Ping:
   - El promedio de la latencia fue: 6.128 ms
   - La desviación estándar de la latencia: 0.660 ms

b) *VPN* activa: en la consola 6.26 se visualiza un fragmento de la prueba de rendimiento (ver consola 6.22) del servidor HOME con VPN activa.

Consola 6.26: Fragmento de la prueba de rendimiento del servidor HOME con VPN activa.

```
root@servidorHOME:~# speedtest
Retrieving speedtest.net configuration...
Testing from Universidad Nacional Autonoma de Mexico \
   (132.248.59.195)...
Retrieving speedtest.net server list...
Selecting best server based on ping...
Hosted by INFINITUM (Mexico City) [11.88 km]: 6.105 ms
Testing download speed.....................................
```

```
Download: 98.00 Mbit/s
Testing upload speed.....................................
Upload: 96.99 Mbit/s
```

Los resultados fueron los siguientes:

1) Upload:
   - El promedio de la velocidad de carga fue: 82.373 Mbps
   - La desviación estándar de la velocidad de carga fue: 10.155 Mbps
2) Download:
   - El promedio de la velocidad de descarga fue: 95.662 Mbps
   - La desviación estándar de la velocidad de descarga fue: 2.973 Mbps
3) Ping:
   - El promedio de la latencia fue: 42.074 ms
   - La desviación estándar de la latencia: 3.05 ms

A continuación, se presenta el análisis de los resultados del servidor HOME con y sin VPN:

1) Upload: Se registra una caída del 14.74 % en la velocidad de carga al emplear la VPN, y en el mismo escenario, la desviación estándar aumento 7.61 Mbps. Este comportamiento indica una pequeña pérdida de estabilidad en el servicio.
2) Download: Se registró una disminución del 3.23 % en la velocidad de descarga al utilizar la VPN. Además, en el mismo escenario, se observó una reducción de 1.434 Mbps en la desviación estándar.
3) Ping: Se tiene una latencia 6.86 veces más grande con VPN y su desviación estándar aumentó 2.39 ms. Este resultado evidencia que la VPN aumenta la latencia al introducir un tiempo de recorrido adicional para las solicitudes y respuestas. Al mantenerse debajo de 100 ms se puede decir que se tiene un tiempo promedio de `ping` (ver referencia [47]).

2. HOME-pc1:

a) Sin *VPN*: en la consola 6.27 se visualiza un fragmento de la prueba de rendimiento (ver consola 6.22) de HOME-pc1 sin VPN activa.

Consola 6.27: Fragmento de la prueba de rendimiento de HOME-pc1 sin VPN activa.

```
root@HOME-pc1:~# speedtest
Retrieving speedtest.net configuration...
Testing from Telmex (189.149.12.31)...
Retrieving speedtest.net server list...
Selecting best server based on ping...
Hosted by Comunicalo de Mexico, S.A. de C.V. (Mexico City) [7.19 \
   km]: 9.171 ms
Testing download speed.....................................
Download: 94.11 Mbit/s
Testing upload speed.....................................
Upload: 98.47 Mbit/s
```

Los resultados del HOME-pc1 sin VPN activa fueron los siguientes:

1) Upload
   - El promedio de la velocidad de carga fue: 98.980 Mbps
   - La desviación estándar de la velocidad de carga fue: 7.504 Mbps
2) Download:
   - El promedio de la velocidad de descarga fue: 97.061 Mbps
   - La desviación estándar de la velocidad de descarga fue: 5.228 Mbps
3) Ping:

- El promedio de la latencia fue: 11.472 ms
- La desviación estándar de la latencia: 5.095 ms

A continuación, se presenta el análisis de los resultados al comparar el servidor HOME con la MV HOME-pc1:

1) Upload: el aumento de 2.4 % de la velocidad promedio de carga y la disminución de 2.65 Mbps de la desviación estándar, se atribuye a las condiciones cambiantes del enlace público.
2) Download: el aumento de 1.9 % de la velocidad promedio de descarga y el aumento 2.25 Mbps de la desviación estándar, se atribuye a las condiciones cambiantes del enlace público.
3) Ping: se observa un aumento en la latencia de 1.87 veces en comparación con el valor obtenido en el servidor HOME. Este incremento se debe a la virtualización anidada de la que depende HOME-pc1.

b) *VPN* activa: En la consola 6.28 se visualiza un fragmento de la prueba de rendimiento (ver consola 6.22) de HOME-pc1 con VPN activa.

Consola 6.28: Fragmento de la prueba de rendimiento de HOME-pc1 con VPN activa.

```
root@HOME-pc1:~# speedtest
Retrieving speedtest.net configuration...
Testing from Universidad Nacional Autonoma de Mexico \
    (132.248.59.195)...
Retrieving speedtest.net server list...
Selecting best server based on ping...
Hosted by INFINITUM (Tlalnepantla) [4.89 km]: 9.082 ms
Testing download speed.....................................
Download: 76.24 Mbit/s
Testing upload speed.......................................
Upload: 83.36 Mbit/s
```

Los resultados del HOME-pc1 con VPN activa fueron los siguientes:

1) Upload
   - El promedio de la velocidad de carga fue: 96.519 Mbps
   - La desviación estándar de la velocidad de carga fue: 2.678 Mbps
2) Download:
   - El promedio de la velocidad de descarga fue: 92.962 Mbps
   - La desviación estándar de la velocidad de descarga fue: 3.90 Mbps
3) Ping:
   - El promedio de la latencia fue:21.872 ms
   - La desviación estándar de la latencia: 1.757 ms

A continuación, se presenta el análisis de los resultados del HOME-pc1 al comparar su comportamiento con y sin VPN:

1) Upload: Se registra una caída del 2.461 % en la velocidad de subida al emplear la VPN, y en el mismo escenario, la desviación estándar disminuyó 4.485 Mbs.
2) Download: Se registró una disminución del 3.23 % en la velocidad de descarga al utilizar la VPN. Además, en el mismo escenario, se observó una reducción de 1.328 Mbs en la desviación estándar.
3) Ping: Se tiene una latencia 1.9 veces más grande con *VPN* y su desviación estándar disminuyó 3.338 Mbs.

3. VPS-IONOS:

a) Sin *VPN*: En la consola 6.29 se visualiza un fragmento de la prueba de rendimiento (ver consola 6.22) del VPS IONOS sin *VPN* activa.

Consola 6.29: Fragmento de la prueba de rendimiento de VPS IONOS sin VPN activa.

```
root@vpsIONOS:~# speedtest
Retrieving speedtest.net configuration...
Testing from 1&1 Internet AG (82.165.209.114)...
Retrieving speedtest.net server list...
Selecting best server based on ping...
Hosted by Cox - Nova (Fairfax, VA) [6548.05 km]: 54.572 ms
Testing download speed................................
Download: 1463.79 Mbit/s
Testing upload speed..................................
Upload: 679.77 Mbit/s
```

Los resultados del VPS-IONOS sin *VPN* activa fueron los siguientes:

1) Upload
   - El promedio de la velocidad de carga fue: 481.946 Mbps
   - La desviación estándar de la velocidad de carga fue: 98.46 Mbps
2) Download:
   - El promedio de la velocidad de descarga fue: 1438.284 Mbps
   - La desviación estándar de la velocidad de descarga fue: 184.002 Mbps
3) Ping:
   - El promedio de la latencia fue: 64.310 ms
   - La desviación estándar de la latencia: 4.174 ms

La desviación estándar en el VPS-IONOS de subida tuvo un valor de 98.46 Mbps, a su vez, la desviación estándar de descarga fue 184.002 Mbps. Estos son valores muy grandes, lo que sugiere que la conexión ofrecida por el proveedor IONOS no es estable. El valor de la latencia promedio se debe a que el servidor desde donde se esta tomando la medición de `ping` esta a una gran distancia (6548.05 km) como se puede ver en la consola 6.29.

b) *VPN* activa: en la consola 6.30 se visualiza un fragmento de la prueba de rendimiento (ver consola 6.22) del VPS-IONOS con *VPN* activa.

Consola 6.30: Fragmento de la prueba de rendimiento del VPS IONOS con VPN activa.

```
root@vpsIONOS:~# speedtest
Retrieving speedtest.net configuration...
Testing from Universidad Nacional Autonoma de Mexico \
   (132.248.59.195)...
Retrieving speedtest.net server list...
Selecting best server based on ping...
Hosted by BESTEL (Mexico City) [11.55 km]: 174.115 ms
Testing download speed................................
Download: 161.92 Mbit/s
Testing upload speed\
   ..............................................
Upload: 96.72 Mbit/s
```

Los resultados del VPS-IONOS con *VPN* activa fueron los siguientes:

1) Upload:
   - El promedio de la velocidad de carga fue: 94.943 Mbps
   - La desviación estándar de la velocidad de carga fue: 10.606 Mbps
2) Download:
   - El promedio de la velocidad de descarga fue: 95.398 Mbps
   - La desviación estándar de la velocidad de descarga fue: 71.578 Mpbs
3) Ping:

- El promedio de la latencia fue: $180.731\,\mathrm{ms}$
- La desviación estándar de la latencia: $3.863\,\mathrm{mbps}$

A continuación, se presenta el análisis de los resultados del VPS-IONOS al comparar su comportamiento con y sin *VPN*:

1) Upload: Se tiene una caída del $80.3\,\%$ debido a que el VPS-IONOS se ve limitado por las capacidades del servidor *VPN*.
2) Download: Se registra una caída del $93.37\,\%$, nuevamente debido a las capacidades del del servidor *VPN*, es decir, existe un cuello de botella entre el VPS-IONOS y el servidor *VPN*.
3) Ping: El aumento de 2.81 veces más del valor de latencia con la *VPN* se debe a que el servidor IONOS se encuentra en Estados Unidos, como se mostró en la consola 6.17, lo que implica que los paquetes tienen que recorrer una mayor distancia y por lo tanto se necesita de un mayor número de saltos.

Capítulo 7

Conclusiones

En este capítulo se describen las conclusiones generales y los trabajos futuros.

7.1. Conclusiones generales

Esta tesis tuvo como finalidad implementar una red de área amplia bajo el paradigma de redes virtuales. Las funciones de la V-WAN planteadas en la sección 3.1 se evaluaron en el capítulo 6 y como resultado se obtuvo que operan de manera correcta, por lo tanto, se puede afirmar que se construyó satisfactoriamente una V-WAN con base al concepto Virtual Network Function (VNF), ya que los servicios de red como son *routers*, *switches* y *firewalls* se ejecutan como instancias de software y se apoyan de una *virtual infrastructure manager* (VIM), que le permite asignar recursos de procesamiento, de almacenamiento y de red a la VNF de manera sencilla.

La V-WAN construida en esta tesis se basa en herramientas de código abierto que ofrece una alternativa a las soluciones de paga, lo cual no quiere decir que se busca competir contra esas soluciones, sino que esta implementación es una opción más a tomar en cuenta.

Las dos nubes privadas que se construyeron en este trabajo de tesis proveen infraestructura como servicio (IasS), de manera simple y fácil de administrar mediante el servicio web Horizon de OpenStack. Esto le permitiría a las instituciones de educación superior (IES) tener sus sistemas en la nube y poder acceder a sus recursos de manera segura desde cualquier parte del mundo. Más aún, alumnos y profesores se podrían ver beneficiados de los recursos de las IaaS, ya que solo requieren de una conexión a Internet. Adicionalmente, optar por herramientas de código abierto puede facilitar que otras instituciones de educación superior implementen y desarrollen este tipo de proyecto con el fin de fomentar un entorno colaborativo entre distintas universidades, contribuyendo así al desarrollo del país.

Por otro lado, la implementación de nubes privadas en las IES se hace relevante en situaciones de paro de actividades, ya sea por conflictos internos o por fuerza mayor como lo fue la pandemia COVID-19 que trajo muchos inconvenientes al interior de las instituciones de educación. Por ello, esta propuesta da una solución que permite la conexión de los recursos entre las diferentes sedes de las IES mediante una conexión a Internet.

En las pruebas de rendimiento de la V-WAN se concluyó que la *VPN* puede aumentar la latencia al introducir un tiempo de recorrido adicional para las solicitudes y respuestas. Esto se evidenció al realizar mediciones en el servidor HOME, donde se observó que la latencia fue 6.86 veces mayor con la *VPN*. En relación con HOME-pc, durante la evaluación de la velocidad de carga y descarga con y sin *VPN*, se observó que el proceso de virtualización resultó en un aumento de la latencia. HOME-pc1 fue la entidad que mostró los valores de desviación estándar más bajos al activar la *VPN*, lo que sugiere una mayor estabilidad en la conexión. En cuanto a la carga y descarga en HOME-pc, este registró valores cercanos a los 100 Mbps, lo cual representa la capacidad máxima del enlace. Esto confirma que WireGuard cumple con su característica de ser un protocolo veloz y estable. En el VPS de IONOS se identificó un cuello de botella en las mediciones de carga y descarga, tanto con y sin *VPN*, debido a

que la velocidad de transferencia del servidor IONOS es considerablemente mayor que la del servidor WireGuard. Otro aspecto importante identificado en el servidor de IONOS es que con la *VPN* activa se registró un ping promedio elevado (180.732 ms), debido a que dicho servidor se encuentra en Estados Unidos, lo que implica que los paquetes deben recorrer una mayor distancia.

Es importante resaltar que el tráfico de MPLS generalmente recibe mayor prioridad en las redes troncales en comparación con el tráfico de Internet. Por lo tanto, al comparar directamente una V-WAN contra MPLS, es claro que este último ofrece una mayor calidad de servicio, dado que utiliza enlaces dedicados y no está sujeto a las fluctuaciones de los enlaces públicos. También es fundamental tener en cuenta que la virtualización conlleva una disminución del rendimiento en comparación con un equipo físico. Sin embargo, una vez concluida la segunda fase de esta propuesta de tesis (SD-WAN), que incluye un controlador y un orquestador inteligentes, se espera que la SD-WAN tenga mejor disponibilidad y mayor flexibilidad, y podrá competir directamente con la solución tradicional de MPLS, y más importante con soluciones SD-WAN de paga.

7.2. Trabajo futuro

La evolución natural de una red virtual de área amplia (V-WAN) es una red de área amplia bajo el paradigma de redes definidas por software (SD-WAN), lo que lograría que las conexiones de red sean dinámicas y elásticamente escalables. Para construir una SD-WAN se puede tomar como base este trabajo de tesis. Para ello, solo haría falta agregar un controlador que regule el tráfico en los canales *VPN* bajo demanda o carga mediante balanceadores.

La V-WAN desarrollada en esta primera etapa cumple la función de ser un entorno de producción sobre el cual se puedan realizar pruebas y establecer comunicación entre sedes distantes. Este *backbone* es necesario para la segunda etapa en la creación de la SD-WAN, ya que establece todos los elementos necesarios que permiten introducir un controlador y un orquestador virtualizado que den lugar a una SD-WAN.

Por otro lado, para mejorar el rendimiento de las nubes implementadas en esta tesis, se puede configurar un nodo de cómputo directamente en un equipo físico en lugar de uno virtual. Esta configuración mejoraría el rendimiento de la VPS, ya que elimina las limitaciones de la virtualización anidada.

Apéndice A

Máquinas virtuales QEMU/KVM

A.1. Nodo de control-CU

En la consola A.1 se muestra el archivo de configuración XML del nodo de control-CU.

Consola A.1: Archivo de configuración XML del nodo de control-CU.

```
<domain type='kvm' id='18'>
  <name>10-IRVING-Controller-Antelope</name>
  <uuid>7c7246f9-c46f-4427-adb5-d2d6a14d38a1</uuid>
  <description></description>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://ubuntu.com/ubuntu/20.10"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit='KiB'>263781376</memory>
  <currentMemory unit='KiB'>263781376</currentMemory>
  <vcpu placement='static'>24</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
    <type arch='x86_64' machine='pc-q35-5.2'>hvm</type>
    <loader readonly='yes' secure='yes' type='pflash'>/usr/share/OVMF/OVMF_CODE_4M.secboot.fd</loader>
    <nvram>/var/lib/libvirt/qemu/nvram/10-IRVING-Controller-Antelope_VARS.fd</nvram>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <vmport state='off'/>
    <smm state='on'/>
  </features>
  <cpu mode='host-passthrough' check='partial' migratable='on'/>
  <clock offset='utc'>
    <timer name='rtc' tickpolicy='catchup'/>
    <timer name='pit' tickpolicy='delay'/>
    <timer name='hpet' present='no'/>
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <pm>
    <suspend-to-mem enabled='no'/>
    <suspend-to-disk enabled='no'/>
  </pm>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='native'/>
      <source dev='/dev/vg0/IRVING-Controller-Antelope' index='1'/>
      <backingStore/>
      <target dev='vda' bus='virtio'/>
      <alias name='virtio-disk0'/>
      <address type='pci' domain='0x0000' bus='0x04' slot='0x00' function='0x0'/>
    </disk>
    <controller type='usb' index='0' model='ich9-ehci1'>
      <alias name='usb'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1d' function='0x7'/>
    </controller>
    <controller type='usb' index='0' model='ich9-uhci1'>
      <alias name='usb'/>
      <master startport='0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1d' function='0x0' multifunction='on'/>
    </controller>
    <controller type='usb' index='0' model='ich9-uhci2'>
      <alias name='usb'/>
      <master startport='2'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1d' function='0x1'/>
    </controller>
    <controller type='usb' index='0' model='ich9-uhci3'>
      <alias name='usb'/>
      <master startport='4'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1d' function='0x2'/>
    </controller>
    <controller type='sata' index='0'>
      <alias name='ide'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1f' function='0x2'/>
```

```
    </controller>
    <controller type='pci' index='0' model='pcie-root'>
      <alias name='pcie.0'/>
    </controller>
    <controller type='pci' index='1' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='1' port='0x10'/>
      <alias name='pci.1'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0' multifunction='on'/>
    </controller>
    <controller type='pci' index='2' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='2' port='0x11'/>
      <alias name='pci.2'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x1'/>
    </controller>
    <controller type='pci' index='3' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='3' port='0x12'/>
      <alias name='pci.3'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x2'/>
    </controller>
    <controller type='pci' index='4' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='4' port='0x13'/>
      <alias name='pci.4'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x3'/>
    </controller>
    <controller type='pci' index='5' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='5' port='0x14'/>
      <alias name='pci.5'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x4'/>
    </controller>
    <controller type='pci' index='6' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='6' port='0x15'/>
      <alias name='pci.6'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x5'/>
    </controller>
    <controller type='pci' index='7' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='7' port='0x16'/>
      <alias name='pci.7'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x6'/>
    </controller>
    <controller type='virtio-serial' index='0'>
      <alias name='virtio-serial0'/>
      <address type='pci' domain='0x0000' bus='0x03' slot='0x00' function='0x0'/>
    </controller>
    <interface type='bridge'>
      <mac address='52:54:00:cc:04:c7'/>
      <source bridge='br0'/>
      <target dev='vnet24'/>
      <model type='virtio'/>
      <alias name='net0'/>
      <address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0'/>
    </interface>
    <interface type='network'>
      <mac address='52:54:00:4d:9e:6a'/>
      <source network='management' portid='612e9d2f-007f-4247-a3ca-452f0d7331e0' bridge='virbr2'/>
      <target dev='vnet25'/>
      <model type='virtio'/>
      <alias name='net1'/>
      <address type='pci' domain='0x0000' bus='0x02' slot='0x00' function='0x0'/>
    </interface>
    <interface type='network'>
      <mac address='52:54:00:8a:7b:63'/>
      <source network='management' portid='12a31b30-9271-446d-8ee0-0adfldfee77e' bridge='virbr2'/>
      <target dev='vnet26'/>
      <model type='virtio'/>
      <alias name='net2'/>
      <address type='pci' domain='0x0000' bus='0x07' slot='0x00' function='0x0'/>
    </interface>
    <serial type='pty'>
      <source path='/dev/pts/6'/>
      <target type='isa-serial' port='0'>
        <model name='isa-serial'/>
      </target>
      <alias name='serial0'/>
    </serial>
    <console type='pty' tty='/dev/pts/6'>
      <source path='/dev/pts/6'/>
      <target type='serial' port='0'/>
      <alias name='serial0'/>
    </console>
    <channel type='unix'>
      <source mode='bind' path='/var/lib/libvirt/qemu/channel/target/domain-18-10-IRVING-Controller/org.qemu.guest_agent.0'/>
      <target type='virtio' name='org.qemu.guest_agent.0' state='disconnected'/>
      <alias name='channel0'/>
      <address type='virtio-serial' controller='0' bus='0' port='1'/>
    </channel>
    <channel type='spicevmc'>
      <target type='virtio' name='com.redhat.spice.0' state='disconnected'/>
      <alias name='channel1'/>
      <address type='virtio-serial' controller='0' bus='0' port='2'/>
    </channel>
    <input type='mouse' bus='ps2'>
      <alias name='input0'/>
    </input>
    <input type='keyboard' bus='ps2'>
      <alias name='input1'/>
    </input>
    <graphics type='spice' port='5903' autoport='yes' listen='127.0.0.1'>
      <listen type='address' address='127.0.0.1'/>
      <image compression='off'/>
    </graphics>
    <sound model='ich9'>
      <alias name='sound0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1b' function='0x0'/>
    </sound>
```

```
    <video>
      <model type='virtio' heads='1' primary='yes'/>
      <alias name='video0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x0'/>
    </video>
    <redirdev bus='usb' type='spicevmc'>
      <alias name='redir0'/>
      <address type='usb' bus='0' port='1'/>
    </redirdev>
    <redirdev bus='usb' type='spicevmc'>
      <alias name='redir1'/>
      <address type='usb' bus='0' port='2'/>
    </redirdev>
    <memballoon model='virtio'>
      <alias name='balloon0'/>
      <address type='pci' domain='0x0000' bus='0x05' slot='0x00' function='0x0'/>
    </memballoon>
    <rng model='virtio'>
      <backend model='random'>/dev/urandom</backend>
      <alias name='rng0'/>
      <address type='pci' domain='0x0000' bus='0x06' slot='0x00' function='0x0'/>
    </rng>
  </devices>
  <seclabel type='dynamic' model='apparmor' relabel='yes'>
    <label>libvirt-7c7246f9-c46f-4427-adb5-d2d6a14d38a1</label>
    <imagelabel>libvirt-7c7246f9-c46f-4427-adb5-d2d6a14d38a1</imagelabel>
  </seclabel>
  <seclabel type='dynamic' model='dac' relabel='yes'>
    <label>+64055:+64055</label>
    <imagelabel>+64055:+64055</imagelabel>
  </seclabel>
</domain>
```

A.2. Nodo de cómputo-CU

En la consola A.2 se muestra el archivo de configuración XML del nodo de cómputo-CU.

Consola A.2: Archivo de configuración XML del nodo de cómputo-CU.

```
<domain type='kvm' id='17'>
  <name>10-IRVING-Compute-Antelope</name>
  <uuid>2d32a7ee-0965-4ad7-8118-972b617a9d4c</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://ubuntu.com/ubuntu/20.10"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit='KiB'>263781376</memory>
  <currentMemory unit='KiB'>263781376</currentMemory>
  <vcpu placement='static'>24</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
    <type arch='x86_64' machine='pc-q35-5.2'>hvm</type>
    <loader readonly='yes' secure='yes' type='pflash'>/usr/share/OVMF/OVMF_CODE_4M.secboot.fd</loader>
    <nvram>/var/lib/libvirt/qemu/nvram/10-IRVING-Compute-Antelope_VARS.fd</nvram>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <vmport state='off'/>
    <smm state='on'/>
  </features>
  <cpu mode='host-passthrough' check='none' migratable='on'/>
  <clock offset='utc'>
    <timer name='rtc' tickpolicy='catchup'/>
    <timer name='pit' tickpolicy='delay'/>
    <timer name='hpet' present='no'/>
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <pm>
    <suspend-to-mem enabled='no'/>
    <suspend-to-disk enabled='no'/>
  </pm>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='native'/>
      <source dev='/dev/vg0/IRVING-Compute-Antelope' index='1'/>
      <backingStore/>
      <target dev='vda' bus='virtio'/>
      <alias name='virtio-disk0'/>
      <address type='pci' domain='0x0000' bus='0x04' slot='0x00' function='0x0'/>
    </disk>
    <controller type='usb' index='0' model='ich9-ehci1'>
      <alias name='usb'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1d' function='0x7'/>
    </controller>
    <controller type='usb' index='0' model='ich9-uhci1'>
      <alias name='usb'/>
      <master startport='0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1d' function='0x0' multifunction='on'/>
    </controller>
    <controller type='usb' index='0' model='ich9-uhci2'>
      <alias name='usb'/>
      <master startport='2'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1d' function='0x1'/>
    </controller>
    <controller type='usb' index='0' model='ich9-uhci3'>
```

```
    <alias name='usb'/>
    <master startport='4'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x1d' function='0x2'/>
  </controller>
  <controller type='sata' index='0'>
    <alias name='ide'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x1f' function='0x2'/>
  </controller>
  <controller type='pci' index='0' model='pcie-root'>
    <alias name='pcie.0'/>
  </controller>
  <controller type='pci' index='1' model='pcie-root-port'>
    <model name='pcie-root-port'/>
    <target chassis='1' port='0x10'/>
    <alias name='pci.1'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0' multifunction='on'/>
  </controller>
  <controller type='pci' index='2' model='pcie-root-port'>
    <model name='pcie-root-port'/>
    <target chassis='2' port='0x11'/>
    <alias name='pci.2'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x1'/>
  </controller>
  <controller type='pci' index='3' model='pcie-root-port'>
    <model name='pcie-root-port'/>
    <target chassis='3' port='0x12'/>
    <alias name='pci.3'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x2'/>
  </controller>
  <controller type='pci' index='4' model='pcie-root-port'>
    <model name='pcie-root-port'/>
    <target chassis='4' port='0x13'/>
    <alias name='pci.4'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x3'/>
  </controller>
  <controller type='pci' index='5' model='pcie-root-port'>
    <model name='pcie-root-port'/>
    <target chassis='5' port='0x14'/>
    <alias name='pci.5'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x4'/>
  </controller>
  <controller type='pci' index='6' model='pcie-root-port'>
    <model name='pcie-root-port'/>
    <target chassis='6' port='0x15'/>
    <alias name='pci.6'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x5'/>
  </controller>
  <controller type='pci' index='7' model='pcie-root-port'>
    <model name='pcie-root-port'/>
    <target chassis='7' port='0x16'/>
    <alias name='pci.7'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x6'/>
  </controller>
  <controller type='virtio-serial' index='0'>
    <alias name='virtio-serial0'/>
    <address type='pci' domain='0x0000' bus='0x03' slot='0x00' function='0x0'/>
  </controller>
  <interface type='bridge'>
    <mac address='52:54:00:29:2c:22'/>
    <source bridge='br0'/>
    <target dev='vnet21'/>
    <model type='virtio'/>
    <alias name='net0'/>
    <address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0'/>
  </interface>
  <interface type='network'>
    <mac address='52:54:00:7b:60:21'/>
    <source network='management' portid='11020c8c-2e8a-4968-a68d-1db2fd4f363a' bridge='virbr2'/>
    <target dev='vnet22'/>
    <model type='virtio'/>
    <alias name='net1'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x00' function='0x0'/>
  </interface>
  <interface type='bridge'>
    <mac address='52:54:00:4e:62:55'/>
    <source bridge='br0'/>
    <target dev='vnet23'/>
    <model type='virtio'/>
    <alias name='net2'/>
    <address type='pci' domain='0x0000' bus='0x07' slot='0x00' function='0x0'/>
  </interface>
  <serial type='pty'>
    <source path='/dev/pts/2'/>
    <target type='isa-serial' port='0'>
      <model name='isa-serial'/>
    </target>
    <alias name='serial0'/>
  </serial>
  <console type='pty' tty='/dev/pts/2'>
    <source path='/dev/pts/2'/>
    <target type='serial' port='0'/>
    <alias name='serial0'/>
  </console>
  <channel type='unix'>
    <source mode='bind' path='/var/lib/libvirt/qemu/channel/target/domain-17-10-IRVING-Compute-An/org.qemu.guest_agent.0'/>
    <target type='virtio' name='org.qemu.guest_agent.0' state='disconnected'/>
    <alias name='channel0'/>
    <address type='virtio-serial' controller='0' bus='0' port='1'/>
  </channel>
  <channel type='spicevmc'>
    <target type='virtio' name='com.redhat.spice.0' state='disconnected'/>
    <alias name='channel1'/>
    <address type='virtio-serial' controller='0' bus='0' port='2'/>
  </channel>
  <input type='mouse' bus='ps2'>
    <alias name='input0'/>
  </input>
  <input type='keyboard' bus='ps2'>
    <alias name='input1'/>
  </input>
  <graphics type='spice' port='5902' autoport='yes' listen='127.0.0.1'>
```

```
      <listen type='address' address='127.0.0.1'/>
      <image compression='off'/>
    </graphics>
    <sound model='ich9'>
      <alias name='sound0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1b' function='0x0'/>
    </sound>
    <video>
      <model type='virtio' heads='1' primary='yes'/>
      <alias name='video0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x0'/>
    </video>
    <redirdev bus='usb' type='spicevmc'>
      <alias name='redir0'/>
      <address type='usb' bus='0' port='1'/>
    </redirdev>
    <redirdev bus='usb' type='spicevmc'>
      <alias name='redir1'/>
      <address type='usb' bus='0' port='2'/>
    </redirdev>
    <memballoon model='virtio'>
      <alias name='balloon0'/>
      <address type='pci' domain='0x0000' bus='0x05' slot='0x00' function='0x0'/>
    </memballoon>
    <rng model='virtio'>
      <backend model='random'>/dev/urandom</backend>
      <alias name='rng0'/>
      <address type='pci' domain='0x0000' bus='0x06' slot='0x00' function='0x0'/>
    </rng>
  </devices>
  <seclabel type='dynamic' model='apparmor' relabel='yes'>
    <label>libvirt-2d32a7ee-0965-4ad7-8118-972b617a9d4c</label>
    <imagelabel>libvirt-2d32a7ee-0965-4ad7-8118-972b617a9d4c</imagelabel>
  </seclabel>
  <seclabel type='dynamic' model='dac' relabel='yes'>
    <label>+64055:+64055</label>
    <imagelabel>+64055:+64055</imagelabel>
  </seclabel>
</domain>
```

A.3. Nodo de red-CU

En la consola A.3 se muestra el archivo de configuración XML del nodo de red-CU.

Consola A.3: Archivo de configuración XML del nodo de red-CU.

```
<domain type='kvm' id='20'>
  <name>10-IRVING-Net-Antelope</name>
  <uuid>3be82447-ee18-4ac8-a7ad-c9c2fd699879</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://ubuntu.com/ubuntu/20.10"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit='KiB'>167772160</memory>
  <currentMemory unit='KiB'>167772160</currentMemory>
  <vcpu placement='static'>12</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
    <type arch='x86_64' machine='pc-q35-5.2'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <vmport state='off'/>
  </features>
  <cpu mode='host-passthrough' check='partial' migratable='on'/>
  <clock offset='utc'>
    <timer name='rtc' tickpolicy='catchup'/>
    <timer name='pit' tickpolicy='delay'/>
    <timer name='hpet' present='no'/>
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <pm>
    <suspend-to-mem enabled='no'/>
    <suspend-to-disk enabled='no'/>
  </pm>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='native'/>
      <source dev='/dev/vg0/IRVING-Net-Antelope' index='2'/>
      <backingStore/>
      <target dev='vda' bus='virtio'/>
      <alias name='virtio-disk0'/>
      <address type='pci' domain='0x0000' bus='0x04' slot='0x00' function='0x0'/>
    </disk>
    <disk type='file' device='cdrom'>
      <driver name='qemu'/>
      <target dev='sda' bus='sata'/>
      <readonly/>
      <alias name='sata0-0-0'/>
      <address type='drive' controller='0' bus='0' target='0' unit='0'/>
    </disk>
    <controller type='usb' index='0' model='ich9-ehci1'>
      <alias name='usb'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1d' function='0x7'/>
    </controller>
```

```
    <controller type='usb' index='0' model='ich9-uhci1'>
      <alias name='usb'/>
      <master startport='0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1d' function='0x0' multifunction='on'/>
    </controller>
    <controller type='usb' index='0' model='ich9-uhci2'>
      <alias name='usb'/>
      <master startport='2'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1d' function='0x1'/>
    </controller>
    <controller type='usb' index='0' model='ich9-uhci3'>
      <alias name='usb'/>
      <master startport='4'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1d' function='0x2'/>
    </controller>
    <controller type='sata' index='0'>
      <alias name='ide'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1f' function='0x2'/>
    </controller>
    <controller type='pci' index='0' model='pcie-root'>
      <alias name='pcie.0'/>
    </controller>
    <controller type='pci' index='1' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='1' port='0x10'/>
      <alias name='pci.1'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0' multifunction='on'/>
    </controller>
    <controller type='pci' index='2' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='2' port='0x11'/>
      <alias name='pci.2'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x1'/>
    </controller>
    <controller type='pci' index='3' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='3' port='0x12'/>
      <alias name='pci.3'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x2'/>
    </controller>
    <controller type='pci' index='4' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='4' port='0x13'/>
      <alias name='pci.4'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x3'/>
    </controller>
    <controller type='pci' index='5' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='5' port='0x14'/>
      <alias name='pci.5'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x4'/>
    </controller>
    <controller type='pci' index='6' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='6' port='0x15'/>
      <alias name='pci.6'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x5'/>
    </controller>
    <controller type='pci' index='7' model='pcie-root-port'>
      <model name='pcie-root-port'/>
      <target chassis='7' port='0x16'/>
      <alias name='pci.7'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x6'/>
    </controller>
    <controller type='virtio-serial' index='0'>
      <alias name='virtio-serial0'/>
      <address type='pci' domain='0x0000' bus='0x03' slot='0x00' function='0x0'/>
    </controller>
    <interface type='bridge'>
      <mac address='52:54:00:40:cd:7c'/>
      <source bridge='br0'/>
      <target dev='vnet30'/>
      <model type='virtio'/>
      <alias name='net0'/>
      <address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0'/>
    </interface>
    <interface type='network'>
      <mac address='52:54:00:2d:81:6a'/>
      <source network='management' portid='dcbd20f5-2153-44f4-b7d1-831699293b47' bridge='virbr2'/>
      <target dev='vnet31'/>
      <model type='virtio'/>
      <alias name='net1'/>
      <address type='pci' domain='0x0000' bus='0x02' slot='0x00' function='0x0'/>
    </interface>
    <interface type='bridge'>
      <mac address='52:54:00:43:7d:53'/>
      <source bridge='br0'/>
      <target dev='vnet32'/>
      <model type='virtio'/>
      <alias name='net2'/>
      <address type='pci' domain='0x0000' bus='0x07' slot='0x00' function='0x0'/>
    </interface>
    <serial type='pty'>
      <source path='/dev/pts/7'/>
      <target type='isa-serial' port='0'>
        <model name='isa-serial'/>
      </target>
      <alias name='serial0'/>
    </serial>
    <console type='pty' tty='/dev/pts/7'>
      <source path='/dev/pts/7'/>
      <target type='serial' port='0'/>
      <alias name='serial0'/>
    </console>
    <channel type='unix'>
      <source mode='bind' path='/var/lib/libvirt/qemu/channel/target/domain-20-10-IRVING-Net-Antelo/org.qemu.guest_agent.0'/>
      <target type='virtio' name='org.qemu.guest_agent.0' state='disconnected'/>
      <alias name='channel0'/>
      <address type='virtio-serial' controller='0' bus='0' port='1'/>
    </channel>
    <channel type='spicevmc'>
```

```
      <target type='virtio' name='com.redhat.spice.0' state='disconnected'/>
      <alias name='channel1'/>
      <address type='virtio-serial' controller='0' bus='0' port='2'/>
    </channel>
    <input type='mouse' bus='ps2'>
      <alias name='input0'/>
    </input>
    <input type='keyboard' bus='ps2'>
      <alias name='input1'/>
    </input>
    <graphics type='spice' port='5904' autoport='yes' listen='127.0.0.1'>
      <listen type='address' address='127.0.0.1'/>
      <image compression='off'/>
    </graphics>
    <sound model='ich9'>
      <alias name='sound0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1b' function='0x0'/>
    </sound>
    <video>
      <model type='virtio' heads='1' primary='yes'/>
      <alias name='video0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x0'/>
    </video>
    <redirdev bus='usb' type='spicevmc'>
      <alias name='redir0'/>
      <address type='usb' bus='0' port='1'/>
    </redirdev>
    <redirdev bus='usb' type='spicevmc'>
      <alias name='redir1'/>
      <address type='usb' bus='0' port='2'/>
    </redirdev>
    <memballoon model='virtio'>
      <alias name='balloon0'/>
      <address type='pci' domain='0x0000' bus='0x05' slot='0x00' function='0x0'/>
    </memballoon>
    <rng model='virtio'>
      <backend model='random'>/dev/urandom</backend>
      <alias name='rng0'/>
      <address type='pci' domain='0x0000' bus='0x06' slot='0x00' function='0x0'/>
    </rng>
  </devices>
  <seclabel type='dynamic' model='apparmor' relabel='yes'>
    <label>libvirt-3be82447-ee18-4ac8-a7ad-c9c2fd699879</label>
    <imagelabel>libvirt-3be82447-ee18-4ac8-a7ad-c9c2fd699879</imagelabel>
  </seclabel>
  <seclabel type='dynamic' model='dac' relabel='yes'>
    <label>+64055:+64055</label>
    <imagelabel>+64055:+64055</imagelabel>
  </seclabel>
</domain>
```

A.4. Nodo de control-HOME

En la consola A.4 se muestra el archivo de configuración XML del nodo de control-HOME.

Consola A.4: Archivo de configuración XML del nodo de control-HOME.

```
<domain type="kvm">
  <name>5-controller-home-v5</name>
  <uuid>cb4ae04e-d967-42cd-a00c-c27b0bce6c36</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://ubuntu.com/ubuntu/20.10"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit="KiB">12130304</memory>
  <currentMemory unit="KiB">12130304</currentMemory>
  <vcpu placement="static">4</vcpu>
  <os>
    <type arch="x86_64" machine="pc-q35-5.2">hvm</type>
    <boot dev="hd"/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <vmport state="off"/>
  </features>
  <cpu mode="host-passthrough" check="none" migratable="on"/>
  <clock offset="utc">
    <timer name="rtc" tickpolicy="catchup"/>
    <timer name="pit" tickpolicy="delay"/>
    <timer name="hpet" present="no"/>
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <pm>
    <suspend-to-mem enabled="no"/>
    <suspend-to-disk enabled="no"/>
  </pm>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type="file" device="disk">
      <driver name="qemu" type="qcow2"/>
      <source file="/mnt/openstack/controller.qcow2"/>
      <target dev="vda" bus="virtio"/>
      <address type="pci" domain="0x0000" bus="0x04" slot="0x00" function="0x0"/>
    </disk>
    <disk type="file" device="cdrom">
      <driver name="qemu" type="raw"/>
      <target dev="sda" bus="sata"/>
      <readonly/>
      <address type="drive" controller="0" bus="0" target="0" unit="0"/>
```

```
    </disk>
    <controller type="usb" index="0" model="ich9-ehci1">
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x7"/>
    </controller>
    <controller type="usb" index="0" model="ich9-uhci1">
      <master startport="0"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x0" multifunction="on"/>
    </controller>
    <controller type="usb" index="0" model="ich9-uhci2">
      <master startport="2"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x1"/>
    </controller>
    <controller type="usb" index="0" model="ich9-uhci3">
      <master startport="4"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x2"/>
    </controller>
    <controller type="sata" index="0">
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1f" function="0x2"/>
    </controller>
    <controller type="pci" index="0" model="pcie-root"/>
    <controller type="pci" index="1" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="1" port="0x10"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x0" multifunction="on"/>
    </controller>
    <controller type="pci" index="2" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="2" port="0x11"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x1"/>
    </controller>
    <controller type="pci" index="3" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="3" port="0x12"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x2"/>
    </controller>
    <controller type="pci" index="4" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="4" port="0x13"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x3"/>
    </controller>
    <controller type="pci" index="5" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="5" port="0x14"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x4"/>
    </controller>
    <controller type="pci" index="6" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="6" port="0x15"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x5"/>
    </controller>
    <controller type="pci" index="7" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="7" port="0x16"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x6"/>
    </controller>
    <controller type="virtio-serial" index="0">
      <address type="pci" domain="0x0000" bus="0x03" slot="0x00" function="0x0"/>
    </controller>
    <interface type="bridge">
      <mac address="52:54:00:62:52:3d"/>
      <source bridge="br0"/>
      <model type="virtio"/>
      <address type="pci" domain="0x0000" bus="0x01" slot="0x00" function="0x0"/>
    </interface>
    <interface type="network">
      <mac address="52:54:00:c4:b6:72"/>
      <source network="network-1"/>
      <model type="virtio"/>
      <address type="pci" domain="0x0000" bus="0x02" slot="0x00" function="0x0"/>
    </interface>
    <serial type="pty">
      <target type="isa-serial" port="0">
        <model name="isa-serial"/>
      </target>
    </serial>
    <console type="pty">
      <target type="serial" port="0"/>
    </console>
    <channel type="unix">
      <target type="virtio" name="org.qemu.guest_agent.0"/>
      <address type="virtio-serial" controller="0" bus="0" port="1"/>
    </channel>
    <channel type="spicevmc">
      <target type="virtio" name="com.redhat.spice.0"/>
      <address type="virtio-serial" controller="0" bus="0" port="2"/>
    </channel>
    <input type="mouse" bus="ps2"/>
    <input type="keyboard" bus="ps2"/>
    <graphics type="spice" autoport="yes">
      <listen type="address"/>
      <image compression="off"/>
    </graphics>
    <sound model="ich9">
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1b" function="0x0"/>
    </sound>
    <video>
      <model type="virtio" heads="1" primary="yes"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x01" function="0x0"/>
    </video>
    <redirdev bus="usb" type="spicevmc">
      <address type="usb" bus="0" port="1"/>
    </redirdev>
    <redirdev bus="usb" type="spicevmc">
      <address type="usb" bus="0" port="2"/>
    </redirdev>
    <memballoon model="virtio">
      <address type="pci" domain="0x0000" bus="0x05" slot="0x00" function="0x0"/>
    </memballoon>
    <rng model="virtio">
      <backend model="random">/dev/urandom</backend>
      <address type="pci" domain="0x0000" bus="0x06" slot="0x00" function="0x0"/>
```

```
    </rng>
  </devices>
</domain>
```

A.5. Nodo de cómputo-HOME

En la consola A.5 se muestra el archivo de configuración XML del nodo de cómputo-HOME.

Consola A.5: Archivo de configuración XML del nodo de cómputo-HOME.

```
<domain type="kvm">
  <name>5-compute-home-v5</name>
  <uuid>6eecbd87-a414-4f50-ae47-c7219aa83097</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://ubuntu.com/ubuntu/20.10"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit="KiB">12130304</memory>
  <currentMemory unit="KiB">12130304</currentMemory>
  <vcpu placement="static">4</vcpu>
  <os>
    <type arch="x86_64" machine="pc-q35-5.2">hvm</type>
    <boot dev="hd"/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <vmport state="off"/>
  </features>
  <cpu mode="host-passthrough" check="none" migratable="on"/>
  <clock offset="utc">
    <timer name="rtc" tickpolicy="catchup"/>
    <timer name="pit" tickpolicy="delay"/>
    <timer name="hpet" present="no"/>
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <pm>
    <suspend-to-mem enabled="no"/>
    <suspend-to-disk enabled="no"/>
  </pm>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type="file" device="disk">
      <driver name="qemu" type="qcow2"/>
      <source file="/mnt/openstack/compute.qcow2"/>
      <target dev="vda" bus="virtio"/>
      <address type="pci" domain="0x0000" bus="0x04" slot="0x00" function="0x0"/>
    </disk>
    <disk type="file" device="cdrom">
      <driver name="qemu" type="raw"/>
      <target dev="sda" bus="sata"/>
      <readonly/>
      <address type="drive" controller="0" bus="0" target="0" unit="0"/>
    </disk>
    <controller type="usb" index="0" model="ich9-ehci1">
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x7"/>
    </controller>
    <controller type="usb" index="0" model="ich9-uhci1">
      <master startport="0"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x0" multifunction="on"/>
    </controller>
    <controller type="usb" index="0" model="ich9-uhci2">
      <master startport="2"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x1"/>
    </controller>
    <controller type="usb" index="0" model="ich9-uhci3">
      <master startport="4"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x2"/>
    </controller>
    <controller type="sata" index="0">
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1f" function="0x2"/>
    </controller>
    <controller type="pci" index="0" model="pcie-root"/>
    <controller type="pci" index="1" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="1" port="0x10"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x0" multifunction="on"/>
    </controller>
    <controller type="pci" index="2" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="2" port="0x11"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x1"/>
    </controller>
    <controller type="pci" index="3" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="3" port="0x12"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x2"/>
    </controller>
    <controller type="pci" index="4" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="4" port="0x13"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x3"/>
    </controller>
    <controller type="pci" index="5" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="5" port="0x14"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x4"/>
    </controller>
    <controller type="pci" index="6" model="pcie-root-port">
      <model name="pcie-root-port"/>
```

```
      <target chassis="6" port="0x15"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x5"/>
    </controller>
    <controller type="pci" index="7" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="7" port="0x16"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x6"/>
    </controller>
    <controller type="virtio-serial" index="0">
      <address type="pci" domain="0x0000" bus="0x03" slot="0x00" function="0x0"/>
    </controller>
    <interface type="bridge">
      <mac address="52:54:00:e6:29:76"/>
      <source bridge="br0"/>
      <model type="virtio"/>
      <address type="pci" domain="0x0000" bus="0x01" slot="0x00" function="0x0"/>
    </interface>
    <interface type="network">
      <mac address="52:54:00:34:19:bf"/>
      <source network="network-1"/>
      <model type="virtio"/>
      <address type="pci" domain="0x0000" bus="0x02" slot="0x00" function="0x0"/>
    </interface>
    <serial type="pty">
      <target type="isa-serial" port="0">
        <model name="isa-serial"/>
      </target>
    </serial>
    <console type="pty">
      <target type="serial" port="0"/>
    </console>
    <channel type="unix">
      <target type="virtio" name="org.qemu.guest_agent.0"/>
      <address type="virtio-serial" controller="0" bus="0" port="1"/>
    </channel>
    <channel type="spicevmc">
      <target type="virtio" name="com.redhat.spice.0"/>
      <address type="virtio-serial" controller="0" bus="0" port="2"/>
    </channel>
    <input type="tablet" bus="usb">
      <address type="usb" bus="0" port="1"/>
    </input>
    <input type="mouse" bus="ps2"/>
    <input type="keyboard" bus="ps2"/>
    <graphics type="spice" autoport="yes">
      <listen type="address"/>
      <image compression="off"/>
    </graphics>
    <sound model="ich9">
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1b" function="0x0"/>
    </sound>
    <video>
      <model type="virtio" heads="1" primary="yes"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x01" function="0x0"/>
    </video>
    <redirdev bus="usb" type="spicevmc">
      <address type="usb" bus="0" port="2"/>
    </redirdev>
    <redirdev bus="usb" type="spicevmc">
      <address type="usb" bus="0" port="3"/>
    </redirdev>
    <memballoon model="virtio">
      <address type="pci" domain="0x0000" bus="0x05" slot="0x00" function="0x0"/>
    </memballoon>
    <rng model="virtio">
      <backend model="random">/dev/urandom</backend>
      <address type="pci" domain="0x0000" bus="0x06" slot="0x00" function="0x0"/>
    </rng>
  </devices>
</domain>
```

A.6. Nodo de red-HOME

En la consola A.6 se muestra el archivo de configuración XML del nodo de red-HOME.

Consola A.6: Archivo de configuración XML del nodo de red-HOME.

```
<domain type="kvm">
  <name>5-net-node-v5</name>
  <uuid>99c00577-ddaf-4d2f-83a8-39c2fb5fa81a</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://ubuntu.com/ubuntu/20.10"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit="KiB">12130304</memory>
  <currentMemory unit="KiB">12130304</currentMemory>
  <vcpu placement="static">4</vcpu>
  <os>
    <type arch="x86_64" machine="pc-q35-5.2">hvm</type>
    <boot dev="hd"/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <vmport state="off"/>
  </features>
  <cpu mode="host-passthrough" check="none" migratable="on"/>
  <clock offset="utc">
    <timer name="rtc" tickpolicy="catchup"/>
    <timer name="pit" tickpolicy="delay"/>
    <timer name="hpet" present="no"/>
  </clock>
```

```
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <pm>
    <suspend-to-mem enabled="no"/>
    <suspend-to-disk enabled="no"/>
  </pm>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type="file" device="disk">
      <driver name="qemu" type="qcow2"/>
      <source file="/mnt/openstack/net.qcow2"/>
      <target dev="vda" bus="virtio"/>
      <address type="pci" domain="0x0000" bus="0x04" slot="0x00" function="0x0"/>
    </disk>
    <disk type="file" device="cdrom">
      <driver name="qemu" type="raw"/>
      <target dev="sda" bus="sata"/>
      <readonly/>
      <address type="drive" controller="0" bus="0" target="0" unit="0"/>
    </disk>
    <controller type="usb" index="0" model="ich9-ehci1">
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x7"/>
    </controller>
    <controller type="usb" index="0" model="ich9-uhci1">
      <master startport="0"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x0" multifunction="on"/>
    </controller>
    <controller type="usb" index="0" model="ich9-uhci2">
      <master startport="2"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x1"/>
    </controller>
    <controller type="usb" index="0" model="ich9-uhci3">
      <master startport="4"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1d" function="0x2"/>
    </controller>
    <controller type="sata" index="0">
      <address type="pci" domain="0x0000" bus="0x00" slot="0x1f" function="0x2"/>
    </controller>
    <controller type="pci" index="0" model="pcie-root"/>
    <controller type="pci" index="1" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="1" port="0x10"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x0" multifunction="on"/>
    </controller>
    <controller type="pci" index="2" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="2" port="0x11"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x1"/>
    </controller>
    <controller type="pci" index="3" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="3" port="0x12"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x2"/>
    </controller>
    <controller type="pci" index="4" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="4" port="0x13"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x3"/>
    </controller>
    <controller type="pci" index="5" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="5" port="0x14"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x4"/>
    </controller>
    <controller type="pci" index="6" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="6" port="0x15"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x5"/>
    </controller>
    <controller type="pci" index="7" model="pcie-root-port">
      <model name="pcie-root-port"/>
      <target chassis="7" port="0x16"/>
      <address type="pci" domain="0x0000" bus="0x00" slot="0x02" function="0x6"/>
    </controller>
    <controller type="virtio-serial" index="0">
      <address type="pci" domain="0x0000" bus="0x03" slot="0x00" function="0x0"/>
    </controller>
    <interface type="bridge">
      <mac address="52:54:00:4e:b7:55"/>
      <source bridge="br0"/>
      <model type="virtio"/>
      <address type="pci" domain="0x0000" bus="0x01" slot="0x00" function="0x0"/>
    </interface>
    <interface type="network">
      <mac address="52:54:00:5d:70:2d"/>
      <source network="network-1"/>
      <model type="virtio"/>
      <address type="pci" domain="0x0000" bus="0x02" slot="0x00" function="0x0"/>
    </interface>
    <serial type="pty">
      <target type="isa-serial" port="0">
        <model name="isa-serial"/>
      </target>
    </serial>
    <console type="pty">
      <target type="serial" port="0"/>
    </console>
    <channel type="unix">
      <target type="virtio" name="org.qemu.guest_agent.0"/>
      <address type="virtio-serial" controller="0" bus="0" port="1"/>
    </channel>
    <channel type="spicevmc">
      <target type="virtio" name="com.redhat.spice.0"/>
      <address type="virtio-serial" controller="0" bus="0" port="2"/>
    </channel>
    <input type="tablet" bus="usb">
      <address type="usb" bus="0" port="1"/>
    </input>
    <input type="mouse" bus="ps2"/>
    <input type="keyboard" bus="ps2"/>
```

```
138     <graphics type="spice" autoport="yes">
139       <listen type="address"/>
140       <image compression="off"/>
141     </graphics>
142     <sound model="ich9">
143       <address type="pci" domain="0x0000" bus="0x00" slot="0x1b" function="0x0"/>
144     </sound>
145     <video>
146       <model type="virtio" heads="1" primary="yes"/>
147       <address type="pci" domain="0x0000" bus="0x00" slot="0x01" function="0x0"/>
148     </video>
149     <redirdev bus="usb" type="spicevmc">
150       <address type="usb" bus="0" port="2"/>
151     </redirdev>
152     <redirdev bus="usb" type="spicevmc">
153       <address type="usb" bus="0" port="3"/>
154     </redirdev>
155     <memballoon model="virtio">
156       <address type="pci" domain="0x0000" bus="0x05" slot="0x00" function="0x0"/>
157     </memballoon>
158     <rng model="virtio">
159       <backend model="random">/dev/urandom</backend>
160       <address type="pci" domain="0x0000" bus="0x06" slot="0x00" function="0x0"/>
161     </rng>
162   </devices>
163 </domain>
```

Apéndice B

Configuración base de los nodos para OpenStack

B.1. NTP

B.1.1. Nodo de control

En la consola B.1 se muestra el archivo completo `/etc/chrony/chrony.conf` del nodo de control-CU.

Consola B.1: Archivo /etc/chrony/chrony.conf del nodo de control-CU.

```
confdir /etc/chrony/conf.d
pool ntp.ubuntu.com        iburst maxsources 4
pool 0.ubuntu.pool.ntp.org iburst maxsources 1
pool 1.ubuntu.pool.ntp.org iburst maxsources 1
pool 2.ubuntu.pool.ntp.org iburst maxsources 2
sourcedir /run/chrony-dhcp
sourcedir /etc/chrony/sources.d
keyfile /etc/chrony/chrony.keys
driftfile /var/lib/chrony/chrony.drift
ntsdumpdir /var/lib/chrony
logdir /var/log/chrony
maxupdateskew 100.0
rtcsync
makestep 1 3
leapsectz right/UTC

server cronos.cenam.mx iburst
allow 192.168.10.0/24
```

B.1.2. Otros nodos

En la consola B.2 se muestra el archivo completo `/etc/chrony/chrony.conf` del nodos cómputo-CU y de nodo de red-CU.

Consola B.2: Archivo /etc/chrony/chrony.conf del nodo de cómputo-CU y del nodo de red-CU.

```
confdir /etc/chrony/conf.d
sourcedir /run/chrony-dhcp
sourcedir /etc/chrony/sources.d
keyfile /etc/chrony/chrony.keys
driftfile /var/lib/chrony/chrony.drift
```

```
ntsdumpdir /var/lib/chrony
logdir /var/log/chrony
maxupdateskew 100.0
rtcsync
makestep 1 3
leapsectz right/UTC

server 192.168.10.200 iburst
```

B.2. SQL database

En la consola B.3 se muestra la salida del comando `mysql_secure_installation`.

Consola B.3: Salida del comando mysql_secure_installation.

```
# mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we -ll need the current
password for the root user. If you ve just installed MariaDB, and
haven-t set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n]
Enabled successfully!
Reloading privilege tables..
 ... Success!


You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n]
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n]
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.
```

```
45
46 Disallow root login remotely? [Y/n]
47  ... Success!
48
49 By default, MariaDB comes with a database named 'test' that anyone can
50 access.  This is also intended only for testing, and should be removed
51 before moving into a production environment.
52
53 Remove test database and access to it? [Y/n]
54  - Dropping test database...
55  ... Success!
56  - Removing privileges on test database...
57  ... Success!
58
59 Reloading the privilege tables will ensure that all changes made so far
60 will take effect immediately.
61
62 Reload privilege tables now? [Y/n]
63  ... Success!
64
65 Cleaning up...
66
67 All done!  If you've completed all of the above steps, your MariaDB
68 installation should now be secure.
69
70 Thanks for using MariaDB!
```

Apéndice C

Openstack

C.1. Nodo de control

C.1.1. Scripts para usuario admin de OpenStack

1. Tal como se muestra en la consola C.1, se crea un archivo llamado `admin-openrc` y se modifican los permisos para que solo el propietario del documento tenga acceso al archivo.

Consola C.1: Comando para crear archivo admin-openrc.

```
root@controllerCU:~# touch admin-openrc && chmod 700 admin-openrc
```

Nota: es fundamental tener en cuenta de que en le archivo `admin-openrc` se van a almacenar credenciales confidenciales para el cliente admin por lo que es importante colocarlo en un lugar seguro y apropiado para la implementación.

2. Dentro del archivo `admin-openrc`(ver consola C.2) se escribe el *script* para el usuario `admin`.

Consola C.2: Archivo admin-openrc.

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=316ac4
export OS_AUTH_URL=http://192.168.10.200:5000/v3
export OS_IDENTITY_API_VERSION=3
```

3. Para cargar las variables de `admin-openrc` se debe estar en el directorio donde está guardado dicho archivo y ejecutar el comando de la consola C.3.

Consola C.3: Comando para adquirir credenciales de admin.

```
root@controllerCU:~# . admin-openrc
```

4. Para verificar que contamos con las variables de entorno, propias del proyecto y usuario admin, se debe de solicitar un token (ver consola C.4).

Consola C.4: Comando para solicitar un token.

```
root@controllerCU:~# openstack token issue
```

C.1.2. Scripts para el usuario home-user de OpenStack

1. Tal como se muestra en la consola C.5, se crea un archivo llamado `home-openrc` y se modifican los permisos para que solo el propietario del documento tenga acceso al archivo.

Consola C.5: Comando para crear archivo home-openrc.

```
root@controllerHOME:~# touch home-openrc && chmod 700 home-openrc
```

NOTA: es fundamental tener en cuenta de que en le archivo `home-openrc` se van a almacenar credenciales confidenciales para el usuario `home-user`.

2. Dentro del archivo `home-openrc` (ver consola C.6) se escribe el script para el usuario `home-user`.

Consola C.6: Archivo home-openrc.

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=home-workspace
export OS_USERNAME=home-user
export OS_PASSWORD=316ac4
export OS_AUTH_URL=http://192.168.0.200:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

3. Para cargar las variables entorno del usuario `home-user` se tiene que estar en el directorio donde esta guardado dicho archivo y ejecutar el comando de la consola C.7.

Consola C.7: Comando para adquirir credenciales del usuario home-user.

```
root@controllerHOME:~# . home-openrc
```

C.1.3. Scripts para el usuario cu-user de OpenStack

1. Tal como se muestra en la consola C.8, se crea un archivo llamado `cu-openrc` y se modifican los permisos para que solo el propietario del documento tenga acceso al archivo.

Consola C.8: Comando para crear archivo cu-openrc

```
root@controllerCU:~# touch cu-openrc && chmod 700 cu-openrc
```

NOTA: es fundamental tener en cuenta de que en le archivo `cu-openrc` se van a almacenar credenciales confidenciales para el usuario `cu-user`.

2. Dentro del archivo `cu-openrc` (ver consola C.9) se escribe el script para el usuario `cu-user`.

Consola C.9: Archivo cu-openrc.

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=cu-workspace
export OS_USERNAME=cu-user
export OS_PASSWORD=316ac4
export OS_AUTH_URL=http://192.168.10.200:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

3. Para cargar las variables entorno del usuario `cu-user` se tiene que estar en el directorio donde esta guardado dicho archivo y ejecutar el comando de la consola C.10.

Consola C.10: Comando para adquirir credenciales del usuario cu-user

```
root@controllerCU:~# . cu-openrc
```

C.1.4. /etc/keystone/keystone.conf

En la consola C.11 se muestra el archivo `keystone.conf` del nodo de control-CU.

Consola C.11: Archivo /etc/keystone/keystone.conf.

```
[DEFAULT]
log_dir = /var/log/keystone
[application_credential]
[assignment]
[auth]
[cache]
[catalog]
[cors]
[credential]
[database]
connection = mysql+pymysql://keystone:316ac4@192.168.10.200/keystone
[domain_config]
[endpoint_filter]
[endpoint_policy]
[eventlet_server]
[extra_headers]
Distribution = Ubuntu
[federation]
[fernet_receipts]
[fernet_tokens]
[healthcheck]
[identity]
[identity_mapping]
[jwt_tokens]
[ldap]
[memcache]
[oauth1]
[oauth2]
[oslo_messaging_amqp]
[oslo_messaging_kafka]
[oslo_messaging_notifications]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[policy]
[profiler]
[receipt]
[resource]
[revoke]
[role]
[saml]
[security_compliance]
[shadow_users]
[token]
```

```
provider = fernet
[tokenless_auth]
[totp]
[trust]
[unified_limit]
[wsgi]
```

C.1.5. /etc/glance/glance-api.conf

En la consola C.12 se muestra el archivo `glance-api.conf` del nodo de control-CU.

Consola C.12: Archivo /etc/glance/glance-api.conf.

```
[DEFAULT]
[barbican]
[barbican_service_user]
[cinder]
[cors]
[database]
connection = mysql+pymysql://glance:316ac4@192.168.10.200/glance
backend = sqlalchemy
[file]
[glance.store.http.store]
[glance.store.rbd.store]
[glance.store.s3.store]
[glance.store.swift.store]
[glance.store.vmware_datastore.store]
[glance_store]
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
[healthcheck]
[image_format]
disk_formats = ami,ari,aki,vhd,vhdx,vmdk,raw,qcow2,vdi,iso,ploop.root-tar
[key_manager]
[keystone_authtoken]
www_authenticate_uri = http://192.168.10.200:5000
auth_url = http://192.168.10.200:5000
memcached_servers = 192.168.10.200:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = 316ac4
[os_brick]
[oslo_concurrency]
[oslo_limit]
[oslo_messaging_amqp]
[oslo_messaging_kafka]
[oslo_messaging_notifications]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[oslo_reports]
[paste_deploy]
```

```
flavor = keystone
[profiler]
[store_type_location_strategy]
[task]
[taskflow_executor]
[vault]
[wsgi]
```

C.1.6. /etc/placement/placement.conf

En la consola C.13 se muestra el archivo `placement.conf` del nodo de control-CU.

Consola C.13: Archivo /etc/placement/placement.conf.

```
[DEFAULT]
[api]
auth_strategy = keystone
[cors]
[keystone_authtoken]
auth_url = http://192.168.10.200:5000/v3
memcached_servers = 192.168.10.200:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = placement
password = 316ac4
[oslo_middleware]
[oslo_policy]
[placement]
[placement_database]
connection = mysql+pymysql://placement:316ac4@192.168.10.200/placement
[profiler]
```

C.1.7. /etc/neutron/neutron.conf del nodo de control

En la consola C.14 se muestra archivo completo `neutron.conf` del nodo de control-CU.

Consola C.14: Archivo /etc/neutron/neutron.conf del nodo de control.

```
[DEFAULT]
core_plugin = ml2
service_plugins = router
transport_url = rabbit://openstack:316ac4@192.168.10.200
auth_strategy = keystone
dhcp_agents_per_network = 2
notify_nova_on_port_status_changes = true
notify_nova_on_port_data_changes = true
[agent]
root_helper = "sudo␣/usr/bin/neutron-rootwrap␣/etc/neutron/rootwrap.conf"
[cache]
[cors]
[database]
connection = mysql+pymysql://neutron:316ac4@192.168.10.200/neutron
[designate]
[experimental]
```

```
[healthcheck]
[ironic]
[keystone_authtoken]
www_authenticate_uri = http://192.168.10.200:5000
auth_url = http://192.168.10.200:5000
memcached_servers = 192.168.10.200:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = 316ac4
[nova]
auth_url = http://192.168.10.200:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = nova
password = 316ac4
[oslo_concurrency]
lock_path = /var/lib/neutron/tmp
[oslo_messaging_amqp]
[oslo_messaging_kafka]
[oslo_messaging_notifications]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[oslo_reports]
[placement]
[privsep]
[profiler]
[quotas]
```

C.1.8. /etc/neutron/plugins/ml2/ml2_conf.ini del nodo de control

En la consola C.15 se muestra archivo completo `ml2_conf.ini` del nodo de control-CU.

Consola C.15: Archivo /etc/neutron/plugins/ml2/ml2_conf.ini del nodo de control.

```
[DEFAULT]
[ml2]
type_drivers = flat,vlan,vxlan
tenant_network_types = vxlan
mechanism_drivers = openvswitch,l2population
extension_drivers = port_security
[ml2_type_flat]
flat_networks = provider
[ml2_type_geneve]
[ml2_type_gre]
[ml2_type_vlan]
network_vlan_ranges = provider
[ml2_type_vxlan]
vni_ranges = 1:1000
[ovs_driver]
```

```
[securitygroup]
[sriov_driver]
```

C.1.9. /etc/nova/nova.conf del nodo de control

En la consola C.16 se muestra archivo completo `nova.conf` del nodo de control-CU.

Consola C.16: Archivo /etc/nova/nova.conf del nodo de control.

```
[DEFAULT]
lock_path = /var/lock/nova
state_path = /var/lib/nova
transport_url = rabbit://openstack:316ac4@192.168.10.200:5672/
my_ip = 192.168.10.200
vnc_enabled = False
novnc_enabled = False
ssl_only = true
cert = /etc/apache2/SSL/fullchain.pem
key = /etc/apache2/SSL/privkey.pem
[api]
auth_strategy = keystone
[api_database]
connection = mysql+pymysql://nova:316ac4@192.168.10.200/nova_api
[barbican]
[barbican_service_user]
[cache]
[cinder]
[compute]
[conductor]
[console]
[consoleauth]
[cors]
[cyborg]
[database]
connection = mysql+pymysql://nova:316ac4@192.168.10.200/nova
[devices]
[ephemeral_storage_encryption]
[filter_scheduler]
[glance]
api_servers = http://192.168.10.200:9292
[guestfs]
[healthcheck]
[hyperv]
[image_cache]
[ironic]
[key_manager]
[keystone]
[keystone_authtoken]
www_authenticate_uri = http://192.168.10.200:5000/
auth_url = http://192.168.10.200:5000/
memcached_servers = 192.168.10.200:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
```

```
password = 316ac4
[libvirt]
[metrics]
[mks]
[neutron]
auth_url = http://192.168.10.200:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = 316ac4
service_metadata_proxy = true
metadata_proxy_shared_secret = 316ac4
[notifications]
[oslo_concurrency]
lock_path = /var/lib/nova/tmp
[oslo_messaging_amqp]
[oslo_messaging_kafka]
[oslo_messaging_notifications]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[oslo_reports]
[pci]
[placement]
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://192.168.10.200:5000/v3
username = placement
password = 316ac4
[powervm]
[privsep]
[profiler]
[quota]
[rdp]
[remote_debug]
[scheduler]
[serial_console]
[service_user]
[spice]
enabled = True
html5proxy_host = 0.0.0.0
html5proxy_port = 6082
keymap = en-us
[upgrade_levels]
[vault]
[vendordata_dynamic_auth]
[vmware]
[vnc]
enabled = False
[workarounds]
```

```
[wsgi]
[zvm]
[cells]
enable = False
[os_region_name]
openstack =
```

C.1.10. /etc/openstack-dashboard/local_settings.py

En la consola C.17 se muestra archivo `local_settings.py` del nodo de control-CU.

Consola C.17: Archivo /etc/openstack-dashboard/local_settings.py.

```
import os
from django.utils.translation import gettext_lazy as _
from horizon.utils import secret_key
from openstack_dashboard.settings import HORIZON_CONFIG
USE_SSL = True
CSRF_COOKIE_SECURE = True
SESSION_COOKIE_SECURE = True
SESSION_COOKIE_HTTPONLY = True
DEBUG = False
LOCAL_PATH = os.path.dirname(os.path.abspath(__file__))
SECRET_KEY = secret_key.generate_or_read_from_file('/var/lib/openstack-dashboard/secret_key'\
    )
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
OPENSTACK_HOST = "192.168.10.200"
ALLOWED_HOSTS = ['*']
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
    'default': {
         'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
         'LOCATION': '192.168.10.200:11211',
    }
}
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
    "volume": 3,
}
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "admin"
OPENSTACK_NEUTRON_NETWORK = {
    'enable_router': True,
    'enable_quotas': True,
    'enable_ipv6': False,
    'enable_distributed_router': True,
    'enable_ha_router': True,
    'enable_fip_topology_check': True,
}
TIME_ZONE = 'America/Mexico_City'
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'console': {
            'format': '%(levelname)s␣%(name)s␣%(message)s'
        },
        'operation': {
            'format': '%(message)s'
        },
    },
    'handlers': {
        'null': {
            'level': 'DEBUG',
            'class': 'logging.NullHandler',
        },
```

```
        'console': {
            'level': 'DEBUG' if DEBUG else 'INFO',
            'class': 'logging.StreamHandler',
            'formatter': 'console',
        },
        'operation': {
            'level': 'INFO',
            'class': 'logging.StreamHandler',
            'formatter': 'operation',
        },
    },
    'loggers': {
        'horizon': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'horizon.operation_log': {
            'handlers': ['operation'],
            'level': 'INFO',
            'propagate': False,
        },
        'openstack_dashboard': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'novaclient': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'cinderclient': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'keystoneauth': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'keystoneclient': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'glanceclient': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'neutronclient': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'swiftclient': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'oslo_policy': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'openstack_auth': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
```

```
126         },
127         'django': {
128             'handlers': ['console'],
129             'level': 'DEBUG',
130             'propagate': False,
131         },
132         'django.template': {
133             'handlers': ['console'],
134             'level': 'INFO',
135             'propagate': False,
136         },
137         'django.db.backends': {
138             'handlers': ['null'],
139             'propagate': False,
140         },
141         'requests': {
142             'handlers': ['null'],
143             'propagate': False,
144         },
145         'urllib3': {
146             'handlers': ['null'],
147             'propagate': False,
148         },
149         'chardet.charsetprober': {
150             'handlers': ['null'],
151             'propagate': False,
152         },
153         'iso8601': {
154             'handlers': ['null'],
155             'propagate': False,
156         },
157         'scss': {
158             'handlers': ['null'],
159             'propagate': False,
160         },
161     },
162 }
163 SECURITY_GROUP_RULES = {
164     'all_tcp': {
165         'name': _('All␣TCP'),
166         'ip_protocol': 'tcp',
167         'from_port': '1',
168         'to_port': '65535',
169     },
170     'all_udp': {
171         'name': _('All␣UDP'),
172         'ip_protocol': 'udp',
173         'from_port': '1',
174         'to_port': '65535',
175     },
176     'all_icmp': {
177         'name': _('All␣ICMP'),
178         'ip_protocol': 'icmp',
179         'from_port': '-1',
180         'to_port': '-1',
181     },
182     'ssh': {
183         'name': 'SSH',
184         'ip_protocol': 'tcp',
185         'from_port': '22',
186         'to_port': '22',
187     },
188     'smtp': {
189         'name': 'SMTP',
190         'ip_protocol': 'tcp',
191         'from_port': '25',
192         'to_port': '25',
193     },
194     'dns': {
195         'name': 'DNS',
196         'ip_protocol': 'tcp',
```

```
        'from_port': '53',
        'to_port': '53',
    },
    'http': {
        'name': 'HTTP',
        'ip_protocol': 'tcp',
        'from_port': '80',
        'to_port': '80',
    },
    'pop3': {
        'name': 'POP3',
        'ip_protocol': 'tcp',
        'from_port': '110',
        'to_port': '110',
    },
    'imap': {
        'name': 'IMAP',
        'ip_protocol': 'tcp',
        'from_port': '143',
        'to_port': '143',
    },
    'ldap': {
        'name': 'LDAP',
        'ip_protocol': 'tcp',
        'from_port': '389',
        'to_port': '389',
    },
    'https': {
        'name': 'HTTPS',
        'ip_protocol': 'tcp',
        'from_port': '443',
        'to_port': '443',
    },
    'smtps': {
        'name': 'SMTPS',
        'ip_protocol': 'tcp',
        'from_port': '465',
        'to_port': '465',
    },
    'imaps': {
        'name': 'IMAPS',
        'ip_protocol': 'tcp',
        'from_port': '993',
        'to_port': '993',
    },
    'pop3s': {
        'name': 'POP3S',
        'ip_protocol': 'tcp',
        'from_port': '995',
        'to_port': '995',
    },
    'ms_sql': {
        'name': 'MS_SQL',
        'ip_protocol': 'tcp',
        'from_port': '1433',
        'to_port': '1433',
    },
    'mysql': {
        'name': 'MYSQL',
        'ip_protocol': 'tcp',
        'from_port': '3306',
        'to_port': '3306',
    },
    'rdp': {
        'name': 'RDP',
        'ip_protocol': 'tcp',
        'from_port': '3389',
        'to_port': '3389',
    },
}
DEFAULT_THEME = 'ubuntu'
```

```
268 WEBROOT='/horizon/'
269 COMPRESS_OFFLINE = True
```

C.1.11. /etc/apache2/conf-available/openstack-dashboard.conf

En la consola C.18 se muestra el archivo `openstack-dashboard.conf` del nodo de control-CU.

Consola C.18: Archivo /etc/apache2/conf-available/openstack-dashboard.conf.

```
1  <VirtualHost *:80>
2    ServerName 192.168.10.200
3    <IfModule mod_rewrite.c>
4      RewriteEngine On
5      RewriteCond %{HTTPS} off
6      RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
7    </IfModule>
8    <IfModule !mod_rewrite.c>
9      RedirectPermanent / https://192.168.10.200
10   </IfModule>
11 </VirtualHost>
12
13 <VirtualHost *:443>
14   ServerName 192.168.10.200
15   SSLEngine On
16   SSLCertificateFile /etc/apache2/SSL/fullchain.pem
17   SSLCACertificateFile /etc/apache2/SSL/fullchain.pem
18   SSLCertificateKeyFile /etc/apache2/SSL/privkey.pem
19   SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown
20   WSGIScriptAlias /horizon /usr/share/openstack-dashboard/openstack_dashboard/wsgi.py \
         process-group=horizon
21   WSGIDaemonProcess horizon user=horizon group=horizon processes=3 threads=10 display-name\
         =%{GROUP}
22   WSGIProcessGroup horizon
23   WSGIApplicationGroup %{GLOBAL}
24   Alias /static /usr/share/openstack-dashboard/openstack_dashboard/static/
25   Alias /static /var/lib/openstack-dashboard/static/
26   Alias /horizon/static /var/lib/openstack-dashboard/static/
27
28   <Directory /usr/share/openstack-dashboard/openstack_dashboard>
29     Require all granted
30   </Directory>
31
32   <Directory /var/lib/openstack-dashboard/static>
33     Require all granted
34   </Directory>
35 </VirtualHost>
```

Donde:

- Se direcciona las solicitudes del puerto 80 al puerto 443.

```
<IfModule !mod_rewrite.c>
RedirectPermanent / https://192.168.10.200
</IfModule>
```

- Se habilita el canal cifrado y se determina la dirección del certificado wildcard y de la llave privada:

```
SSLEngine On
SSLCertificateFile /etc/apache2/SSL/fullchain.pem
SSLCACertificateFile /etc/apache2/SSL/fullchain.pem
SSLCertificateKeyFile /etc/apache2/SSL/privkey.pem
```

C.1.12. Salidas

En la consola C.19 se muestra un fragmento de la lista de recursos disponibles para `Placement`.

Consola C.19: Fragmento de la salida del comando para ver recursos disponibles de Placement.

```
# openstack --os-placement-api-version 1.6 trait list --sort-column name
+---------------------------------------+
| name                                  |
+---------------------------------------+
| COMPUTE_ACCELERATORS                  |
| COMPUTE_ADDRESS_SPACE_EMULATED        |
| COMPUTE_ADDRESS_SPACE_PASSTHROUGH     |
| COMPUTE_ARCH_AARCH64                  |
| COMPUTE_ARCH_MIPSEL                   |
| HW_GPU_MAX_DISPLAY_HEADS_2            |
| HW_GPU_MAX_DISPLAY_HEADS_4            |
| HW_GPU_MAX_DISPLAY_HEADS_6            |
| HW_GPU_MAX_DISPLAY_HEADS_8            |
| HW_GPU_RESOLUTION_W1024H600           |
| HW_GPU_RESOLUTION_W1024H768           |
| HW_GPU_RESOLUTION_W1152H864           |
| HW_GPU_RESOLUTION_W1280H1024          |
| HW_GPU_RESOLUTION_W1280H720           |
| HW_GPU_RESOLUTION_W1280H768           |
| HW_GPU_RESOLUTION_W1280H800           |
| HW_GPU_RESOLUTION_W1360H768           |
| HW_GPU_RESOLUTION_W1366H768           |
| HW_GPU_RESOLUTION_W1440H900           |
| HW_GPU_RESOLUTION_W1600H1200          |
| HW_GPU_RESOLUTION_W1600H900           |
| HW_GPU_RESOLUTION_W1680H1050          |
| HW_GPU_RESOLUTION_W1920H1080          |
| HW_GPU_RESOLUTION_W1920H1200          |
| HW_GPU_RESOLUTION_W2560H1440          |
| HW_GPU_RESOLUTION_W2560H1600          |
| HW_GPU_RESOLUTION_W320H240            |
| HW_GPU_RESOLUTION_W3840H2160          |
| HW_GPU_RESOLUTION_W640H480            |
| HW_GPU_RESOLUTION_W7680H4320          |
| HW_GPU_RESOLUTION_W800H600            |
| HW_NIC_ACCEL_DEFLATE                  |
| HW_NIC_ACCEL_DIFFIEH                  |
| HW_NIC_ACCEL_ECC                      |
| HW_NIC_ACCEL_IPSEC                    |
| HW_NIC_ACCEL_LZS                      |
| HW_NIC_ACCEL_RSA                      |
| HW_NIC_ACCEL_SSL                      |
| HW_NIC_ACCEL_TLS                      |
| HW_NIC_DCB_ETS                        |
| HW_NIC_DCB_PFC                        |
| HW_NIC_DCB_QCN                        |
| HW_NIC_MULTIQUEUE                     |
| HW_NIC_OFFLOAD_FDF                    |
| HW_NIC_OFFLOAD_GENEVE                 |
| HW_NIC_OFFLOAD_GRE                    |
| HW_NIC_OFFLOAD_GRO                    |
| HW_NIC_OFFLOAD_GSO                    |
| HW_NIC_OFFLOAD_L2CRC                  |
| HW_NIC_OFFLOAD_LRO                    |
| HW_NIC_OFFLOAD_LSO                    |
| HW_NIC_OFFLOAD_QINQ                   |
| HW_NIC_OFFLOAD_RDMA                   |
| HW_NIC_OFFLOAD_RX                     |
| HW_NIC_OFFLOAD_RXHASH                 |
| HW_NIC_OFFLOAD_RXVLAN                 |
| HW_NIC_OFFLOAD_SCS                    |
| HW_NIC_OFFLOAD_SG                     |
| HW_NIC_OFFLOAD_SWITCHDEV              |
| HW_NIC_OFFLOAD_TCS                    |
| HW_NIC_OFFLOAD_TSO                    |
| HW_NIC_OFFLOAD_TX                     |
| HW_NIC_OFFLOAD_TXUDP                  |
| HW_NIC_OFFLOAD_TXVLAN                 |
| HW_NIC_OFFLOAD_UCS                    |
| HW_NIC_OFFLOAD_UFO                    |
| HW_NIC_OFFLOAD_VXLAN                  |
| HW_NIC_PROGRAMMABLE_PIPELINE          |
| HW_NIC_SRIOV                          |
| HW_NIC_SRIOV_MULTIQUEUE               |
| HW_NIC_SRIOV_QOS_RX                   |
| HW_NIC_SRIOV_QOS_TX                   |
| HW_NIC_SRIOV_TRUSTED                  |
| HW_NIC_VMDQ                           |
| HW_NUMA_ROOT                          |
| MISC_SHARES_VIA_AGGREGATE             |
| OWNER_CYBORG                          |
| OWNER_NOVA                            |
| STORAGE_DISK_HDD                      |
| STORAGE_DISK_SSD                      |
+---------------------------------------+
```

En la consola C.20 se muestra la salida del comando de inicialización de la base de datos de `Nova`.

Consola C.20: Salida del comando de inicialización de la base de datos de Nova.

```
root@controllerCU:~#  nova-manage db sync
2023-07-31 21:20:00.540 9948 INFO alembic.runtime.migration [None req-12eec5f3-477c-4929-9dea-57c5b41c540e - - - - - -] Context impl MySQLImpl.
2023-07-31 21:20:00.541 9948 INFO alembic.runtime.migration [None req-12eec5f3-477c-4929-9dea-57c5b41c540e - - - - - -] Will assumenon-transactional \
       DDL.
2023-07-31 21:20:00.569 9948 INFO alembic.runtime.migration [None req-12eec5f3-477c-4929-9dea-57c5b41c540e - - - - - -] Running upgrade  -> 8\
       f2f1571d55b, Initial version
```

```
2023-07-31 21:20:07.084 9948 INFO alembic.runtime.migration [None req-12eec5f3-477c-4929-9dea-57c5b41c540e - - - - - -] Running upgrade 8f2f1571d55b \
      -> 16f1fbcab42b, Resolve shadow table diffs
2023-07-31 21:20:07.206 9948 INFO alembic.runtime.migration [None req-12eec5f3-477c-4929-9dea-57c5b41c540e - - - - - -] Running upgrade 16f1fbcab42b \
      -> ccb0fa1a2252, Add encryption fields to BlockDeviceMapping
2023-07-31 21:20:07.456 9948 INFO alembic.runtime.migration [None req-12eec5f3-477c-4929-9dea-57c5b41c540e - - - - - -] Running upgrade ccb0fa1a2252 \
      -> 960aac0e09ea, de-duplicate_indexes_in_instances__console_auth_tokens
2023-07-31 21:20:07.523 9948 INFO alembic.runtime.migration [None req-12eec5f3-477c-4929-9dea-57c5b41c540e - - - - - -] Context impl MySQLImpl.
2023-07-31 21:20:07.524 9948 INFO alembic.runtime.migration [None req-12eec5f3-477c-4929-9dea-57c5b41c540e - - - - - -] Will assumenon-transactional \
      DDL.
2023-07-31 21:20:07.547 9948 INFO alembic.runtime.migration [None req-12eec5f3-477c-4929-9dea-57c5b41c540e - - - - - -] Running upgrade  -> 8\
      f2f1571d55b, Initial version
2023-07-31 21:20:13.511 9948 INFO alembic.runtime.migration [None req-12eec5f3-477c-4929-9dea-57c5b41c540e - - - - - -] Running upgrade 8f2f1571d55b \
      -> 16f1fbcab42b, Resolve shadow table diffs
2023-07-31 21:20:13.605 9948 INFO alembic.runtime.migration [None req-12eec5f3-477c-4929-9dea-57c5b41c540e - - - - - -] Running upgrade 16f1fbcab42b \
      -> ccb0fa1a2252, Add encryption fields to BlockDeviceMapping
2023-07-31 21:20:13.837 9948 INFO alembic.runtime.migration [None req-12eec5f3-477c-4929-9dea-57c5b41c540e - - - - - -] Running upgrade ccb0fa1a2252 \
      -> 960aac0e09ea
```

En la consola C.21 se muestra la salida del comando de inicialización de la base de datos de `Neutron`.

Consola C.21: Salida del comando de inicialización de datos de Neutron.

```
--config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head
INFO  [alembic.runtime.migration] Context impl MySQLImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
  Running upgrade for neutron ...
INFO  [alembic.runtime.migration] Context impl MySQLImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade  -> kilo
INFO  [alembic.runtime.migration] Running upgrade kilo -> 354db87e3225
INFO  [alembic.runtime.migration] Running upgrade 354db87e3225 -> 599c6a226151
INFO  [alembic.runtime.migration] Running upgrade 599c6a226151 -> 52c5312f6baf
INFO  [alembic.runtime.migration] Running upgrade 52c5312f6baf -> 313373c0ffee
INFO  [alembic.runtime.migration] Running upgrade 313373c0ffee -> 8675309a5c4f
INFO  [alembic.runtime.migration] Running upgrade 8675309a5c4f -> 45f955889773
INFO  [alembic.runtime.migration] Running upgrade 45f955889773 -> 26c371498592
INFO  [alembic.runtime.migration] Running upgrade 26c371498592 -> 1c844d1677f7
INFO  [alembic.runtime.migration] Running upgrade 1c844d1677f7 -> 1b4c6e320f79
INFO  [alembic.runtime.migration] Running upgrade 1b4c6e320f79 -> 48153cb5f051
INFO  [alembic.runtime.migration] Running upgrade 48153cb5f051 -> 9859ac9c136
INFO  [alembic.runtime.migration] Running upgrade 9859ac9c136 -> 34af2b5c5a59
INFO  [alembic.runtime.migration] Running upgrade 34af2b5c5a59 -> 59cb5b6cf4d
INFO  [alembic.runtime.migration] Running upgrade 59cb5b6cf4d -> 13cfb89f881a
INFO  [alembic.runtime.migration] Running upgrade 13cfb89f881a -> 32e5974ada25
INFO  [alembic.runtime.migration] Running upgrade 32e5974ada25 -> ec7fcfbf72ee
INFO  [alembic.runtime.migration] Running upgrade ec7fcfbf72ee -> dce3ec7a25c9
INFO  [alembic.runtime.migration] Running upgrade dce3ec7a25c9 -> c3a73f615e4
INFO  [alembic.runtime.migration] Running upgrade c3a73f615e4 -> 659bf3d90664
INFO  [alembic.runtime.migration] Running upgrade 659bf3d90664 -> 1df244e556f5
INFO  [alembic.runtime.migration] Running upgrade 1df244e556f5 -> 19f26505c74f
INFO  [alembic.runtime.migration] Running upgrade 19f26505c74f -> 15be73214821
INFO  [alembic.runtime.migration] Running upgrade 15be73214821 -> b4caf27aae4
INFO  [alembic.runtime.migration] Running upgrade b4caf27aae4 -> 15e43b934f81
INFO  [alembic.runtime.migration] Running upgrade 15e43b934f81 -> 31ed664953e6
INFO  [alembic.runtime.migration] Running upgrade 31ed664953e6 -> 2f9e956e7532
INFO  [alembic.runtime.migration] Running upgrade 2f9e956e7532 -> 3894bccad37f
INFO  [alembic.runtime.migration] Running upgrade 3894bccad37f -> 0e66c5227a8a
INFO  [alembic.runtime.migration] Running upgrade 0e66c5227a8a -> 45f8dd33480b
INFO  [alembic.runtime.migration] Running upgrade 45f8dd33480b -> 5abc0278ca73
INFO  [alembic.runtime.migration] Running upgrade 5abc0278ca73 -> d3435b514502
INFO  [alembic.runtime.migration] Running upgrade d3435b514502 -> 30107ab6a3ee
INFO  [alembic.runtime.migration] Running upgrade 30107ab6a3ee -> c415aab1c048
INFO  [alembic.runtime.migration] Running upgrade c415aab1c048 -> a963b38d82f4
INFO  [alembic.runtime.migration] Running upgrade kilo -> 30018084ec99
INFO  [alembic.runtime.migration] Running upgrade 30018084ec99 -> 4ffceebfada
INFO  [alembic.runtime.migration] Running upgrade 4ffceebfada -> 5498d17be016
INFO  [alembic.runtime.migration] Running upgrade 5498d17be016 -> 2a16083502f3
INFO  [alembic.runtime.migration] Running upgrade 2a16083502f3 -> 2e5352a0ad4d
INFO  [alembic.runtime.migration] Running upgrade 2e5352a0ad4d -> 11926bcfe72d
INFO  [alembic.runtime.migration] Running upgrade 11926bcfe72d -> 4af11ca47297
INFO  [alembic.runtime.migration] Running upgrade 4af11ca47297 -> 1b294093239c
INFO  [alembic.runtime.migration] Running upgrade 1b294093239c -> 8a6d8bdae39
INFO  [alembic.runtime.migration] Running upgrade 8a6d8bdae39 -> 2b4c2465d44b
INFO  [alembic.runtime.migration] Running upgrade 2b4c2465d44b -> e3278ee65050
INFO  [alembic.runtime.migration] Running upgrade e3278ee65050 -> c6c112992c9
INFO  [alembic.runtime.migration] Running upgrade c6c112992c9 -> 5ffceebfada
INFO  [alembic.runtime.migration] Running upgrade 5ffceebfada -> 4ffceebfcdc
INFO  [alembic.runtime.migration] Running upgrade 4ffceebfcdc -> 7bbb25278f53
INFO  [alembic.runtime.migration] Running upgrade 7bbb25278f53 -> 89ab9a816d70
INFO  [alembic.runtime.migration] Running upgrade 89ab9a816d70 -> c879c5e1ee90
INFO  [alembic.runtime.migration] Running upgrade c879c5e1ee90 -> 8fd3918ef6f4
INFO  [alembic.runtime.migration] Running upgrade 8fd3918ef6f4 -> 4bcd4df1f426
INFO  [alembic.runtime.migration] Running upgrade 4bcd4df1f426 -> b67e765a3524
INFO  [alembic.runtime.migration] Running upgrade a963b38d82f4 -> 3d0e74aa7d37
INFO  [alembic.runtime.migration] Running upgrade 3d0e74aa7d37 -> 030a959ceafa
INFO  [alembic.runtime.migration] Running upgrade 030a959ceafa -> a5648cfeeadf
INFO  [alembic.runtime.migration] Running upgrade a5648cfeeadf -> 0f5bef0f87d4
INFO  [alembic.runtime.migration] Running upgrade 0f5bef0f87d4 -> 67daae611b6e
INFO  [alembic.runtime.migration] Running upgrade b67e765a3524 -> a84ccf28f06a
INFO  [alembic.runtime.migration] Running upgrade a84ccf28f06a -> 7d9d8eeec6ad
INFO  [alembic.runtime.migration] Running upgrade 67daae611b6e -> 6b461a21bcfc
INFO  [alembic.runtime.migration] Running upgrade 6b461a21bcfc -> 5cd92597d11d
INFO  [alembic.runtime.migration] Running upgrade 5cd92597d11d -> 929c968efe70
INFO  [alembic.runtime.migration] Running upgrade 929c968efe70 -> a9c43481023c
INFO  [alembic.runtime.migration] Running upgrade a9c43481023c -> 804a3c76314c
INFO  [alembic.runtime.migration] Running upgrade 804a3c76314c -> 2b42d90729da
INFO  [alembic.runtime.migration] Running upgrade 2b42d90729da -> 62c781cb6192
INFO  [alembic.runtime.migration] Running upgrade 62c781cb6192 -> c8c222d42aa9
INFO  [alembic.runtime.migration] Running upgrade c8c222d42aa9 -> 349b6fd605a6
INFO  [alembic.runtime.migration] Running upgrade 349b6fd605a6 -> 7d32f979895f
INFO  [alembic.runtime.migration] Running upgrade 7d32f979895f -> 594422d373ee
INFO  [alembic.runtime.migration] Running upgrade 594422d373ee -> 61663558142c
INFO  [alembic.runtime.migration] Running upgrade 61663558142c -> 867d39095bf4, port forwarding
INFO  [alembic.runtime.migration] Running upgrade 867d39095bf4 -> d72db3e25539, modify uniq port forwarding
INFO  [alembic.runtime.migration] Running upgrade d72db3e25539 -> cada2437bf41
INFO  [alembic.runtime.migration] Running upgrade cada2437bf41 -> 195176fb410d, router gateway IP QoS
INFO  [alembic.runtime.migration] Running upgrade 195176fb410d -> fb0167bd9639
```

```
INFO  [alembic.runtime.migration] Running upgrade fb0167bd9639 -> 0ff9e3881597
INFO  [alembic.runtime.migration] Running upgrade 0ff9e3881597 -> 9bfad3f1e780
INFO  [alembic.runtime.migration] Running upgrade 9bfad3f1e780 -> 63fd95af7dcd
INFO  [alembic.runtime.migration] Running upgrade 63fd95af7dcd -> c613d0b82681
INFO  [alembic.runtime.migration] Running upgrade c613d0b82681 -> c3e9d13c4367
INFO  [alembic.runtime.migration] Running upgrade c3e9d13c4367 -> 86274d77933e
INFO  [alembic.runtime.migration] Running upgrade 86274d77933e -> f4b9654dd40c
INFO  [alembic.runtime.migration] Running upgrade f4b9654dd40c -> a010322604bc
INFO  [alembic.runtime.migration] Running upgrade a010322604bc -> 263d454a9655
INFO  [alembic.runtime.migration] Running upgrade 263d454a9655 -> Ibac91d24da2
INFO  [alembic.runtime.migration] Running upgrade Ibac91d24da2 -> 2217c4222de6
INFO  [alembic.runtime.migration] Running upgrade 2217c4222de6 -> 18a7e90ae768
INFO  [alembic.runtime.migration] Running upgrade 18a7e90ae768 -> e4e236b0e1ff
INFO  [alembic.runtime.migration] Running upgrade e4e236b0e1ff -> e88badaa9591
INFO  [alembic.runtime.migration] Running upgrade e88badaa9591 -> d8bdf05313f4
INFO  [alembic.runtime.migration] Running upgrade d8bdf05313f4 -> dfe425060830
INFO  [alembic.runtime.migration] Running upgrade dfe425060830 -> fd6107509ccd
INFO  [alembic.runtime.migration] Running upgrade fd6107509ccd -> 1ea5dab0897a
INFO  [alembic.runtime.migration] Running upgrade 1ea5dab0897a -> 49d8622c5221
INFO  [alembic.runtime.migration] Running upgrade 49d8622c5221 -> I38991de2b4
INFO  [alembic.runtime.migration] Running upgrade I38991de2b4 -> 532aa95457e2
INFO  [alembic.runtime.migration] Running upgrade 532aa95457e2 -> f010820fc498
INFO  [alembic.runtime.migration] Running upgrade f010820fc498 -> a964d94b4677
INFO  [alembic.runtime.migration] Running upgrade a964d94b4677 -> 26d1e9f5c766
INFO  [alembic.runtime.migration] Running upgrade 26d1e9f5c766 -> 1e0744e4ffea
INFO  [alembic.runtime.migration] Running upgrade 1e0744e4ffea -> 6135a7bd4425
INFO  [alembic.runtime.migration] Running upgrade 6135a7bd4425 -> 8df53b0d2c0e
INFO  [alembic.runtime.migration] Running upgrade 8df53b0d2c0e -> 1bb3393de75d, add qos policy rule Packet Rate Limit
INFO  [alembic.runtime.migration] Running upgrade 1bb3393de75d -> c181bb1d89e4
INFO  [alembic.runtime.migration] Running upgrade c181bb1d89e4 -> ba859d649675
INFO  [alembic.runtime.migration] Running upgrade ba859d649675 -> e981acd076d3
INFO  [alembic.runtime.migration] Running upgrade e981acd076d3 -> 76df7844a8c6, add Local IP tables
INFO  [alembic.runtime.migration] Running upgrade 76df7844a8c6 -> 1ffef8d6f371, migrate RBAC registers from "target_tenant" to "target_project"
INFO  [alembic.runtime.migration] Running upgrade 1ffef8d6f371 -> 8160f7a9cebb, drop portbindingportstable
INFO  [alembic.runtime.migration] Running upgrade 8160f7a9cebb -> cd9ef14ccf87
INFO  [alembic.runtime.migration] Running upgrade cd9ef14ccf87 -> 34cf8b009713
INFO  [alembic.runtime.migration] Running upgrade 34cf8b009713 -> I43e0b669096
INFO  [alembic.runtime.migration] Running upgrade I43e0b669096 -> 4e6e655746f6
INFO  [alembic.runtime.migration] Running upgrade 4e6e655746f6 -> 659cbedf30a1
INFO  [alembic.runtime.migration] Running upgrade 659cbedf30a1 -> 21ff98fabab1
INFO  [alembic.runtime.migration] Running upgrade 21ff98fabab1 -> 5881373af7f5
INFO  [alembic.runtime.migration] Running upgrade 5881373af7f5 -> fc153938cdc1
INFO  [alembic.runtime.migration] Running upgrade 7d9d8eeec6ad -> a8b517cff8ab
INFO  [alembic.runtime.migration] Running upgrade a8b517cff8ab -> 3b935b28e7a0
INFO  [alembic.runtime.migration] Running upgrade 3b935b28e7a0 -> b12a3ef66e62
INFO  [alembic.runtime.migration] Running upgrade b12a3ef66e62 -> 97c25b0d2353
INFO  [alembic.runtime.migration] Running upgrade 97c25b0d2353 -> 2e0d7a8a1586
INFO  [alembic.runtime.migration] Running upgrade 2e0d7a8a1586 -> 5c85685d616d
  OK
```

C.2. Nodo de cómputo

C.2.1. /etc/nova/nova.conf

En la consola C.22 se muestra el archivo `nova.conf` del nodo de cómputo-CU.

Consola C.22: Archivo /etc/nova/nova.conf del nodo de cómputo.

```
[DEFAULT]
log_dir = /var/log/nova
lock_path = /var/lock/nova
state_path = /var/lib/nova
my_ip = 192.168.10.205
transport_url = rabbit://openstack:316ac4@192.168.10.200
vnc_enabled = false
[api]
auth_strategy = keystone
[api_database]
connection = sqlite:////var/lib/nova/nova_api.sqlite
[barbican]
[barbican_service_user]
[cache]
[cinder]
[compute]
[conductor]
[console]
[consoleauth]
[cors]
[cyborg]
[database]
```

```
connection = sqlite:////var/lib/nova/nova.sqlite
[devices]
[ephemeral_storage_encryption]
[filter_scheduler]
[glance]
api_servers = http://192.168.10.200:9292
[guestfs]
[healthcheck]
[hyperv]
[image_cache]
[ironic]
[key_manager]
[keystone]
[keystone_authtoken]
www_authenticate_uri = http://192.168.10.200:5000/
auth_url = http://192.168.10.200:5000/
memcached_servers = 192.168.10.200:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = 316ac4
[libvirt]
[metrics]
[mks]
[neutron]
auth_url = http://192.168.10.200:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = 316ac4
[notifications]
[oslo_concurrency]
lock_path = /var/lib/nova/tmp
[oslo_messaging_amqp]
[oslo_messaging_kafka]
[oslo_messaging_notifications]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[oslo_reports]
[pci]
[placement]
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://192.168.10.200:5000/v3
username = placement
password = 316ac4
[powervm]
```

```
[privsep]
[profiler]
[quota]
[rdp]
[remote_debug]
[scheduler]
[serial_console]
[service_user]
[spice]
enabled = True
html5proxy_base_url = https://irvingyohanan.com:6082/spice_auto.html
keymap = en-us
server_listen = 0.0.0.0
server_proxyclient_address = $my_ip
[upgrade_levels]
[vault]
[vendordata_dynamic_auth]
[vmware]
[vnc]
enabled = False
[workarounds]
[wsgi]
[zvm]
[cells]
enable = False
[os_region_name]
openstack =
```

C.2.2. /etc/neutron/neutron.conf

En la consola C.23 se muestra el archivo `neutron.conf` del nodo de cómputo-CU.

Consola C.23: Archivo /etc/neutron/neutron.conf del nodo de cómputo.

```
[DEFAULT]
core_plugin = ml2
transport_url = rabbit://openstack:316ac4@192.168.10.200
auth_strategy = keystone
[agent]
root_helper = "sudo␣/usr/bin/neutron-rootwrap␣/etc/neutron/rootwrap.conf"
[cache]
[cors]
[database]
[designate]
[experimental]
[healthcheck]
[ironic]
[keystone_authtoken]
www_authenticate_uri = http://192.168.10.200:5000
auth_url = http://192.168.10.200:5000
memcached_servers = 192.168.10.200:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
```

```
password = 316ac4
[nova]
[oslo_concurrency]
lock_path = /var/lib/neutron/tmp
[oslo_messaging_amqp]
[oslo_messaging_kafka]
[oslo_messaging_notifications]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[oslo_reports]
[placement]
[privsep]
[profiler]
[quotas]
[ssl]
```

C.2.3. /etc/neutron/plugins/ml2/openvswitch_agent.ini

En la consola C.24 se muestra el archivo `openvswitch_agent.ini` del nodo de cómputo-CU.

Consola C.24: Archivo /etc/neutron/plugins/ml2/openvswitch_agent.ini del nodo de cómputo.

```
[DEFAULT]
[agent]
tunnel_types = vxlan
l2_population = True
[dhcp]
[network_log]
[ovs]
bridge_mappings = provider:sw-provider
local_ip = 192.168.10.206
[securitygroup]
firewall_driver = iptables_hybrid
```

C.2.4. /etc/neutron/dhcp_agent.ini

En la consola C.25 se muestra el archivo `dhcp_agent.ini` del nodo de cómputo-CU.

Consola C.25: Archivo /etc/neutron/dhcp_agent.ini del nodo de cómputo.

```
[DEFAULT]
interface_driver = openvswitch
enable_isolated_metadata = True
force_metadata = True
[agent]
[ovs]
```

C.2.5. /etc/neutron/metadata_agent.ini

En la consola C.26 se muestra el archivo `metadata_agent.ini` del nodo de cómputo-CU.

Consola C.26: Archivo /etc/neutron/metadata_agent.ini del nodo de cómputo.

```
[DEFAULT]
nova_metadata_host = 192.168.10.200
```

```
metadata_proxy_shared_secret = 316ac4
[agent]
[cache]
```

C.3. Nodo de red

C.3.1. /etc/neutron/neutron.conf del nodo de red

En la consola C.27 se muestra el archivo `neutron.conf` del nodo de red-CU.

Consola C.27: Archivo /etc/neutron/neutron.conf del nodo de red.

```
[DEFAULT]
core_plugin = ml2
transport_url = rabbit://openstack:316ac4@192.168.10.200
auth_strategy = keystone
[agent]
root_helper = "sudo␣/usr/bin/neutron-rootwrap␣/etc/neutron/rootwrap.conf"
[cache]
[cors]
[database]
[designate]
[experimental]
[healthcheck]
[ironic]
[keystone_authtoken]
www_authenticate_uri = http://192.168.10.200:5000
auth_url = http://192.168.10.200:5000
memcached_servers = 192.168.10.200:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = 316ac4
[nova]
[oslo_concurrency]
lock_path = /var/lib/neutron/tmp
[oslo_messaging_amqp]
[oslo_messaging_kafka]
[oslo_messaging_notifications]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[oslo_reports]
[placement]
[privsep]
[profiler]
[quotas]
[ssl]
```

C.3.2. /etc/neutron/plugins/ml2/openvswitch_agent.ini del nodo de red

En la consola C.28 se muestra el archivo `openvswitch_agent.ini` del nodo de red-CU.

Consola C.28: Archivo /etc/neutron/plugins/ml2/openvswitch_agent.ini del nodo de red.

```
[DEFAULT]
[agent]
tunnel_types = vxlan
l2_population = True
[dhcp]
[network_log]
[ovs]
bridge_mappings = provider:sw-provider
local_ip = 192.168.10.211
[securitygroup]
firewall_driver = iptables_hybrid
```

C.3.3. /etc/neutron/l3_agent.ini

En la consola C.29 se muestra el archivo `l3_agent.ini` del nodo de red-CU.

Consola C.29: Archivo /etc/neutron/l3_agent.ini del nodo de red.

```
[DEFAULT]
interface_driver = openvswitch
[agent]
[network_log]
[ovs]
```

C.4. Habilitación de puente de red para firewall de OpenStack

1. Para habilitar la compatibilidad del puente de red, se cargar el módulo del *kernel* `br_netfilter` modificando al archivo `/etc/modules`, como se ilustra en la consola C.30.

Consola C.30: Archivo /etc/modules.

```
br_netfilter
```

2. Se reinicia la maquina para cargar el módulo con el comando de la consola C.31.

Consola C.31: Comando para reiniciar servidor.

```
# shutdown -r now
```

3. Utilizando el comando que se muestra en la consola C.32, se verifica que el módulo haya sido cargado correctamente.

Consola C.32: Comando para verifica que el módulo br_netfilter haya sido cargado correctamente.

```
# lsmod | grep br_netfilter
Module                  Size  Used by
br_netfilter           32768  0
bridge                307200  1 br_netfilter
```

4. Para garantizar el funcionamiento del firewall, se crea un archivo llamando `/etc/sysctl.d/br-openstack.conf` empleando el comando que se muestra en la consola C.33.

Consola C.33: Comando para crear el archivo /etc/sysctl.d/br-openstack.conf.

```
# emacs /etc/sysctl.d/br-openstack.conf
```

5. Utilizando el comando que se muestra en la consola C.34, se permite el uso de los filtros de puente de red.

Consola C.34: Archivo /etc/sysctl.d/br-openstack.conf.

```
1 net.bridge.bridge-nf-call-ip6tables = 1
2 net.bridge.bridge-nf-call-iptables = 1
```

6. Se ejecuta el comando de la consola C.35 para que los cambios hechos al archivo `br-openstack.conf` surtan efecto.

Consola C.35: Comando sysctl para br-openstack.conf.

```
# sysctl -p /etc/sysctl.d/br-openstack.conf
```

7. Se verifica que los valores de `net.bridge` estén establecidos en `1` utilizando el comando proporcionado en la consola C.36.

Consola C.36: Comando para verificar estado de net.bridge.

```
# sudo sysctl net.bridge.bridge-nf-call-iptables
sudo sysctl net.bridge.bridge-nf-call-ip6tables
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
```

C.5. Creación de switch con OpenvSwitch

A continuación se describe el proceso para crear un switch con `OpenvSwitch` para proveer de internet a las instancias creadas por `OpenStack`.

1. Se crea un *switch* y se le nombra `sw-provider` utilizando el comando mostrado en la consola C.37.

Consola C.37: Comando para crear un switch con OpenvSwitch.

```
root@controllerCU:~# ovs-vsctl add-br sw-provider
```

2. Como se muestra en la consola C.38, se añade un puerto al *switch* que actuará como la interfaz utilizada por `OpenStack` para proveer de internet a las instancias.

Consola C.38: Comando para agregar puerto sw-provider a switch de OpenvSwitch.

```
root@controllerCU:~# ovs-vsctl add-port sw-provider enp2s0
```

Ninguna interfaz que forme parte de un *switch* de `OpenvSwitch` debe de tener IP, así que para solucionarlo se crea un *script* que se ocupa cada vez que de reinicia el nodo de computo y el de red.

3. Se crea un archivo llamado `/bin/redup` (ver consola C.39), que tiene como función restaurar la funcionalidad de la red al remover la *IP* que tenia la interfaz y asignarle una al *switch* recién creado.

Consola C.39: Archivo /bin/redup.

```
#! /bin/bash

ip addr flush dev enp7s0
ip addr add 192.168.10.206/24 dev sw-provider
ip link set sw-provider up
```

4. Para ejecutar el *script*, se modifican los permisos del archivo `/bin/redup` utilizando el comando de la consola C.40.

Consola C.40: Comando para cambiar permisos al archivo /bin/redup.

```
root@controllerCU:~# chmod +x /bin/redup
```

5. Con el comando mostrado en al consola C.41, se ejecuta el *script* para que los comandos tenga efectos.

Consola C.41: Comando ejecutar script redup.

```
root@controllerCU:~# redup
```

6. Se visualiza el estado de red como se muestra en la consola C.42.

Consola C.42: Comando ip address.

```
root@computeCU:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group \
    default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP\
     group default qlen 1000
    link/ether 52:54:00:29:2c:22 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.205/24 brd 192.168.10.255 scope global enp1s0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe29:2c22/64 scope link
       valid_lft forever preferred_lft forever
3: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP\
     group default qlen 1000
    link/ether 52:54:00:7b:60:21 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.231/24 metric 100 brd 10.0.0.255 scope global dynamic enp2s0
       valid_lft 2751sec preferred_lft 2751sec
    inet6 fe80::5054:ff:fe7b:6021/64 scope link
       valid_lft forever preferred_lft forever
4: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master \
    ovs-system state UP group default qlen 1000
    link/ether 52:54:00:4e:62:55 brd ff:ff:ff:ff:ff:ff
5: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group \
    default qlen 1000
    link/ether 86:df:0f:0c:c6:7a brd ff:ff:ff:ff:ff:ff
6: sw-provider: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue \
    state UNKNOWN group default qlen 1000
    link/ether 06:5a:d8:04:ae:40 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.206/24 scope global sw-provider
       valid_lft forever preferred_lft forever
    inet6 fe80::45a:d8ff:fe04:ae40/64 scope link
```

Apéndice D

Certbot

D.1. README de Certbot

En la consola D.1 se muestra archivo `README` de `certbot`.

Consola D.1: Archivo README de certbot.

```
This directory contains your keys and certificates.

`privkey.pem`  : the private key for your certificate.
`fullchain.pem`: the certificate file used in most server software.
`chain.pem`    : used for OCSP stapling in Nginx >=1.3.7.
`cert.pem`     : will break many server configurations, and should not be used
                 without reading further documentation (see link below).

WARNING: DO NOT MOVE OR RENAME THESE FILES!
         Certbot expects these files to remain in this location in order
         to function properly!

We recommend not moving these files. For more information, see the Certbot
User Guide at https://certbot.eff.org/docs/using.html#where-are-my-certificates.
```

D.2. Salida completa de la solicitud de retos de certbot

En la consola D.2 se muestra la salida completa de la solicitud de retos de `certbot`.

Consola D.2: Salida completa de la solitud de desafios de certbot.

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Requesting a certificate for *.yohanpena.com and yohanpena.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please deploy a DNS TXT record under the name:

_acme-challenge.yohanpena.com.

with the following value:

CeMw_0r5sfwXvVjzOoStLBPSKSFI9Wqf6FKYsTEpPOA

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Press Enter to Continue

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please deploy a DNS TXT record under the name:
```

```
18
19 _acme-challenge.yohanpena.com.
20
21 with the following value:
22
23 SMF_s2olgLVBRy5uQXGmX3IbEBXrPwZfqCeFyPQ6Ufk
24
25 (This must be set up in addition to the previous challenges; do not remove,
26 replace, or undo the previous challenge tasks yet. Note that you might be
27 asked to create multiple distinct TXT records with the same name. This is
28 permitted by DNS standards.)
29
30 Before continuing, verify the TXT record has been deployed. Depending on the DNS
31 provider, this may take some time, from a few seconds to multiple minutes. You can
32 check if it has finished deploying with aid of online tools, such as the Google
33 Admin Toolbox: https://toolbox.googleapps.com/apps/dig/#TXT/_acme-challenge.\
       yohanpena.com.
34 Look for one or more bolded line(s) below the line ';ANSWER'. It should show the
35 value(s) you've just added.
36
37 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
38 Press Enter to Continue
39
40 Successfully received certificate.
41 Certificate is saved at: /etc/letsencrypt/live/yohanpena.com/fullchain.pem
42 Key is saved at:         /etc/letsencrypt/live/yohanpena.com/privkey.pem
43 This certificate expires on 2024-04-25.
44 These files will be updated when the certificate renews.
45
46 NEXT STEPS:
47 - This certificate will not be renewed automatically. Autorenewal of --manual \
       certificates requires the use of an authentication hook script (--manual-auth-\
       hook) but one was not provided. To renew this certificate, repeat this same \
       certbot command before the certificate's expiry date.
```

Bibliografía

[1] Cloudflare. (s. f.). *What is a WAN?*. https://www.cloudflare.com/learning/network-layer/what-is-a-wan/

[2] Palo Alto Networks. (s. f.). *What is SD-WAN?*. https://www.paloaltonetworks.com/cyberpedia/what-is-sd-wan

[3] Red Hat. (s. f.). *What is SD-WAN?*. https://www.redhat.com/en/topics/edge-computing/what-is-sd-wan

[4] Fortinet. (2023, 27 de septiembre). *2023 Gartner Magic QuadrantTM for SD-WAN - Fortinet named a Leader*. https://www.fortinet.com/solutions/gartner-wan-edge

[5] Certbot. (s. f.). *About certbot*. https://certbot.eff.org/pages/about

[6] Let's Encrypt. (s. f.). *Acerca de Let's encrypt*. https://letsencrypt.org/es/about/

[7] Certbot (s. f.). *CertBot Glossary*. https://certbot.eff.org/glossary

[8] Certbot (s. f.). *Frequently asked questions*. https://certbot.eff.org/faq#does-let-s-encrypt-issue-wildcard-certificates

[9] Maurya, H. (2021, 2 de diciembre). *How to install and configure KVM on Debian 11 Bullseye Linux*. Linux Shout. https://linux.how2shout.com/how-to-install-and-configure-kvm-on-debian-11-bullseye-linux/

[10] Canonical Ubuntu. (s. f.). *What is OpenStack?*. Linux Shout. https://ubuntu.com/openstack/what-is-openstack

[11] Google Cloud. (s. f.). *What is IaaS (Infrastructure as a Service)?* . https://cloud.google.com/learn/what-is-iaas

[12] OpenStack. (s. f.). *Get started with OpenStack*. https://docs.openstack.org/install-guide/get-started-with-openstack.html

[13] Amazon Web Services. (s. f.). *¿Qué es la virtualización?*. https://aws.amazon.com/es/what-is/virtualization/

[14] Red Hat. (s. f.). *What is KVM?*. https://www.redhat.com/en/topics/virtualization/what-is-KVM

[15] IBM. (s. f). *¿Qué es la virtualización?* . https://www.ibm.com/mx-es/topics/virtualization

[16] IBM. (s. f). *What are hypervisors?*. https://www.ibm.com/topics/hypervisors

[17] Open vSwitch (s. f). *What Is Open vSwitch? — Open vSwitch 3.3.90 documentation*. https://docs.openvswitch.org/en/latest/intro/what-is-ovs/

[18] OpenMetal. (2024, 6 de marzo). *What is OpenStack Glance?*. OpenMetal IaaS. https://openmetal.io/docs/glossary/what-is-openstack-glance/

[19] OpenStack. (s. f.). *Placement*. hhttps://docs.openstack.org/placement/latest/

[20] OpenMetal. (2023, 26 de julio). *What is OpenStack Keystone?*. OpenMetal IaaS.. https://openmetal.io/docs/glossary/what-is-openstack-keystone/

[21] University of Cambridge. (s. f.). *Computing service: OpenStack Nova*. RCC Documentation. https://docs.hpc.cam.ac.uk/cloud/userguide/03-nova.html

[22] OpenStack. (2023, 30 de agosto). *OpenStack Compute (nova)*. https://docs.openstack.org/nova/2023.2/index.html

[23] GeeksforGeeks. (2022, 13 de noviembre). *What is Openstack Nova Service?* GeeksforGeeks. https://www.geeksforgeeks.org/what-is-openstack-nova-service/

[24] University of Cambridge. (s. f.). *Networking service: OpenStack Neutron*. RCC Documentation. https://docs.hpc.cam.ac.uk/cloud/userguide/02-neutron.html

[25] OpenMetal. (2023, 29 de Agosto). *What is OpenStack Neutron?*. OpenMetal IaaS. https://openmetal.io/docs/glossary/what-is-openstack-neutron/

[26] OpenStack. (s. f.). *Networking architecture — Security* https://docs.openstack.org/security-guide/networking/architecture.html

[27] Red Hat. (s. f.). *Chapter 2. OpenStack Networking Concepts Red Hat OpenStack Platform 11*. https://access.redhat.com/documentation/th-th/red_hat_openstack_platform/11/html/networking_guide/openstack_networking_concepts#doc-wrapper

[28] OpenMetal. (2023, 26 de Julio). *What is OpenStack Horizon?*.OpenMetal IaaS. https://openmetal.io/docs/glossary/what-is-openstack-horizon/

[29] Fedora Documentation Team. (2023, 02 de febrero). *How to enable nested virtualization in KVM*. Fedora Docs. https://docs.fedoraproject.org/en-US/quick-docs/using-nested-virtualization-in-kvm/

[30] OpenMetal. (2023, 26 de julio). *What is OpenStack Horizon?*. https://openmetal.io/docs/glossary/what-is-openstack-horizon/

[31] OpenStack. (s. f.). *SQL database for Ubuntu — Installation Guide documentation.*. https://docs.openstack.org/install-guide/environment-sql-database-ubuntu.html

[32] OpenStack. (s. f.). *Message queue for Ubuntu — Installation Guide documentation.*. https://docs.openstack.org/install-guide/environment-messaging-ubuntu.html

[33] OpenStack. (s. f.). *Memcached for Ubuntu — Installation Guide.*. https://docs.openstack.org/install-guide/environment-memcached-ubuntu.html

[34] Red Hat. (s. f.). *What is etcd?*. https://www.redhat.com/en/topics/containers/what-is-etcd

[35] Equipo editorial de IONOS. (2020, 12 de octubre). *WireGuard VPN: conceptos básicos*.https://www.ionos.es/digitalguide/servidores/herramientas/wireguard-vpn-conceptos-basicos/

[36] Ubuntu. (s. f.). *WireGuard VPN - Introduction*.https://ubuntu.com/server/docs/wireguard-vpn-introduction

[37] Donenfeld, J. A. (s. f.). *Protocol Cryptography - WireGuard*. https://www.wireguard.com/protocol/

[38] Donenfeld, J. A. (s. f.). *WireGuard: fast, modern, secure VPN tunnel*. https://www.wireguard.com/

[39] Donenfeld, J. A. (s. f.). *Known Limitations - WireGuard*. https://www.wireguard.com/known-limitations/

[40] Equipo editorial de IONOS. (2020, 12 de octubre). *WireGuard VPN: conceptos básicos*. https://www.ionos.es/digitalguide/servidores/herramientas/wireguard-vpn-conceptos-basicos/

[41] Red Hat. (s. f.). *Chapter 8. Setting up a WireGuard VPN Red Hat Enterprise Linux 9*. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/configuring_and_managing_networking/assembly_setting-up-a-wireguard-vpn_configuring-and-managing-networking

[42] Li, Z. y. C. (2024, 23 de enero). *What Is VXLAN? How Does It Differ from VLAN? - Huawei*. Huawei. *WireGuard VPN: conceptos básicos*. https://info.support.huawei.com/info-finder/encyclopedia/en/VXLAN.html

[43] Donenfeld, J. A. (2020, 1 de junio). *WireGuard:Next Generation Kernel Network Tunnel*. WireGuard Documentación. *WireGuard VPN: conceptos básicos*. https://www.wireguard.com/papers/wireguard.pdf

[44] Higgins, M. (2024, 11 de marzo). *VPN WireGuard: What is it and how does it work? NordVPN*.NordVPN. https://nordvpn.com/es/blog/vpn-wireguard/

[45] Tanenbaum, A.S., *Redes de computadoras*, Editorial Alhambra S. A. (SP) 2023, https://books.google.com.mx/books?id=WWD-4oF9hjEC

[46] Palo Alto Networks.(s.f).*What is multiprotocol label switching*.https://www.paloaltonetworks.com/cyberpedia/mpls-what-is-multiprotocol-label-switching

[47] Kaspersky.(s.f).*How to improve game performance*.https://latam.kaspersky.com/resource-center/preemptive-safety/how-to-improve-game-performanceafety/how-to-improve-game-performance

[48] Amazon Web Services. (s. f). *¿Qué es una VPN?*.https://aws.amazon.com/es/what-is/vpn/

[49] Openstack. (s. f). *Installation-Placement*. https://docs.openstack.org/placement/2023.1/install/

[50] Openstack. (s. f). *Image service overview*. https://docs.openstack.org/glance/2023.1/install/get-started.html

[51] Openstack. (s. f). *Security Groups*. https://docs.openstack.org/nova/latest/user/security-groups.html

[52] Openstack. (s. f). *Flavors*. https://docs.openstack.org/nova/latest/user/flavors.html

[53] Openstack. (s. f). *Keystone-Create a domain, projects, users, and roles*. https://docs.openstack.org/keystone/2023.1/install/keystone-users-obs.html

[54] Openstack. (s. f). *Create OpenStack client environment scripts — keystone*. https://docs.openstack.org/keystone/2023.1/install/keystone-openrc-ubuntu.html