ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y LA COMUNICACIÓN

ÁREA DE INGENIERÍA TELEMÁTICA

DESPLIEGUE DE UN ENTORNO CLOUD PRIVADO BASADO EN TECNOLOGÍAS DE CÓDIGO ABIERTO

D. BORES DÍAZ, Carlos
TUTOR: D. MELENDI PALACIO, David

FECHA: Febrero 2025

Resumen

En las últimas décadas las grandes infraestructuras cloud donde los recursos se distribuyen de manera escalable y flexible han ido sustituyendo a los antiguos sistemas centralizados y servidores autogestionados de las empresas. Antiguamente las corporaciones eran responsables de administrar y gestionar sus propios centros de datos lo que incurría en un alto coste de mantenimiento, escalabilidad limitada y complejidad en las operaciones, pero con la llegada de las redes de alta velocidad, así como el avance de los sistemas virtualizados el paradigma de los sistemas cloud se consolidan como solución eficiente. Los modelos como IaaS (Infrastructure as a Service) y PaaS (Platform as a Service) permitieron a las empresas externalizar sus necesidades de cómputo, almacenamiento y redes, eliminando la necesidad de infraestructuras locales y fomentando la adopción de entornos flexibles y altamente disponibles.

En el ecosistema cloud, OpenStack surge en el 2010 como una solución de nube open source impulsada por la NASA y Rackspace para ofrecer de forma gratuita una plataforma escalable y modular que permitiese a proveedores ofrecer servicios cloud sin alquilar espacios a grandes empresas propietarias. Con una arquitectura basada en componentes como Nova (computación), Neutron (redes) y Glance (almacenamiento de imágenes), OpenStack se ha consolidado como una de las alternativas más utilizadas en entornos empresariales y de telecomunicaciones. No obstante, tiene un problema, un sistema tan completo supone una complejidad alta en su despliegue y configuración, lo que ha llevado a lanzar alternativas más sencillas y ligeras.

En este contexto aparece MicroStack, una versión simplificada de OpenStack desarrollada por canonical y diseñada para una fácil implementación de entornos cloud privados pequeños y de desarrollo personal. Con una instalación rápida y sencilla basada en Snap es una alternativa asequible y atractiva para laboratorios o pequeñas empresas con presupuestos pequeños.

A pesar de que MicroStack tiene una documentación abundante y densa, y certificaciones para su administración, despliegue y manejo, MicroStack carece de una gran documentación oficial y se basa en comunidades, foros y publicaciones personales de valientes desarrolladores y administradores de sistemas y redes que dedican su tiempo a investigar.

Contenidos

Figuras

Tablas

1. Objetivos y alcance del trabajo

En primer lugar, se pretende introducir el concepto del cloud computing, desarrollando sobre sus características, ventajas y el impacto que ha supuesto en la transformación digital de las empresas. Se detallarán los diferentes modelos de servicios de nube, así como sus proveedores mayoritarios y la evolución del mercado cloud destacando empresas como AWS (Amazon Web Services), Azure (Microsoft) y Google Cloud y su importancia en la influencía del mercado cloud.

Como el trabajo trata principalmente sobre cloud open source, se introducirá OpenStack para dar paso a MicroStack, a su arquitectura, funcionalidades y componentes clave. Se explicarán los servicios que ofrece, como la gestión de instancias, redes y almacenamiento comparándolo con OpenStack. Además, se describirá como MicroStack simplifica la instalación mediante Snap.

Como paso natural, se llegará a la implementación de la infraestructura y por tanto a la confección de un manual detallado que permita la instalación y configuración de MicroStack de manera sencilla y estructurada. Este manual incluirá los requisitos previos del sistema, los pasos de instalación utilizando Snap y la configuración de los servicios básicos para desplegar una nube funcional. También se incorporarán buenas prácticas y soluciones a posibles errores comunes para asegurar una implementación estable y eficiente.

Uno de los aspectos críticos de cualquier infraestructura cloud es la seguridad. Por ello, se realizarán pruebas de intrusión sobre la implementación de MicroStack con el fin de evaluar su resistencia frente a amenazas comunes.

2. Tecnologías empleadas

2.1.- Cloud Computing

El cloud computing o computación en la nube es un modelo de tecnología informática distribuida que permite el acceso a grandes recursos computacionales a través de internet y de forma descentralizada evitando la dependencia de estructuras locales como servidores físicos de almacenamiento en un centro de datos propio. Los proveedores ofrecen acceso bajo demanda a sus recursos, una especie de arrendamiento de equipos, pero en este caso, de cómputo, redes y almacenamiento, pagando por tiempo de uso.

El cloud se basa en la virtualización y automatización de despliegues facilitando a empresas y usuarios individuales escalar sus sistemas sin el problema de espacio y organización que ello conlleva. Es una revolución de los sistemas informáticos dando lugar a grandes infraestructuras.

La nube es una arquitectura distribuida, no solo entre servidores físicos, si no entre grandes centros de datos distribuidos geográficamente lo que permite latencias más bajas mundialmente para una empresa local, por ejemplo. Se fundamenta en la virtualización, la compartición de un mismo hardware para distintos sistemas operativos y aplicaciones optimizando el uso de recursos. Por otro lado, está la capacidad de escalar dinámicamente, aumentando o reduciendo los recursos en tiempo real y sin intervención de terceros facilitando paneles de administración a clientes, lo que deriva en un modelo de autoservicio. Otro punto clave es la multitenencia donde varios clientes pueden usar el mismo hardware de forma aislada y segura a los demás usuarios. [2]

Características	Modelos tradicionales	Cloud Computing
Infraestructura	En local, propietaria	Proveedor externo distribuida
Escalabilidad	Limitada al espacio disponible	Escalable según demanda
Costes	Inversión inicial alta	Pago por uso
Mantenimiento	Personal de soporte 24/7	Mantenido por proveedor
Accesibilidad	Limitada a la red de la empresa	Desde cualquier sitio con internet
Seguridad	Control absoluto	Responsabilidad proveedor
Tiempo Despliegue	Lenta por instalación	Aprovisionamiento instantáneo

Tabla 2.1.- Diferencias modelos auto host y modelos cloud

Los modelos de servicio en la computación en la nube se categorizan en tres niveles principales:

- IaaS (Infraestructura como Servicio): Proporciona recursos virtualizados, como máquinas virtuales, almacenamiento y redes, permitiendo a las organizaciones desplegar y gestionar su propia infraestructura sin la necesidad de adquirir y mantener hardware físico.
- PaaS (Plataforma como Servicio): Ofrece un entorno de desarrollo preconfigurado, facilitando a los desarrolladores la creación, despliegue y gestión de aplicaciones sin preocuparse por la administración de la infraestructura subyacente.
- SaaS (Software como Servicio): Permite el acceso a aplicaciones completamente gestionadas por el proveedor, eliminando la necesidad de instalación, mantenimiento o administración por parte del usuario final. Este modelo es ideal para organizaciones que requieren soluciones listas para su uso con mínima personalización, como las ofrecidas por Google Workspace.[41]

Modelo	Ventajas	Desventajas	Productos
IaaS	Flexibilidad Control y personalización	Conocimientos avanzados Complejidad gestión Costes elevados	Amazon EC2 Google Engine MS Azure VM
PaaS	Despliegue rápido de aplicaciones No necesitas gestionar infraestructura Herramientas preconfiguradas	Menor control Dependencia proveedor para actualizaciones	Heroku Google App Engine MS App Services
SaaS	No requiere conocimientos avanzados Muy accesible	Falta personalización Dependencia total proveedor	MS 365 Google Workspace

Tabla 2.2.- Esquema modelos clouds.

Figura 2.1.- Esquema modelos cloud.[41]

2.2.- OpenStack

OpenStack es una plataforma open source diseñada para gestión de nube pública y privadas. Su arquitectura modular permite la administración de recursos de almacenamiento, redes y computación de forma escalable y flexible. Desde su lanzamiento se ha hecho un hueco en el mercado y se ha consolidado como alternativa gratuita de Amazon, Microsoft o Google.

Destaca por su código abierto y sin licencias, su capacidad de crecimiento dinámico según la demanda, la integración con contenedores y Kubernetes de forma nativa y un API abierta que facilita la automatización y el desarrollo de herramientas de gestión.

OpenStack se basa en una arquitectura modular, cada módulo o componente proporciona una funcionalidad específica. Nova para administración de instancias, Neutron para redes, almacenamiento a través de Swift (Object Storage) para almacenamiento de objetos distribuidos, ideal para datos no estructurados y copias de seguridad, Cinder (Block Storage) proporciona volúmenes de almacenamiento persistente a máquinas virtuales y Manila (Shared Filesystem) para soporte de almacenamiento en línea como NFS y CIFS, Glance para las imágenes de instalación, Veilometer para la monitorización y telemetría y Horizon como interfaz gráfica web.[48]

2.3.- MICROSTACK

OpenStack ha demostrado ser una solución robusta y escalable, pero en contra de su potencial, está la complejidad de despliegue y mantenimiento que ha sido una barrera para el acceso a este entorno. Por ello la empresa Canonical ha desarrollado MicroStack, una versión simplificada para laboratorios, entornos de prueba y pequeñas empresas. [1]

Microstack permite a los administradores y desarrolladores desplegar una nube privada en minutos a través de paquetes Snap, lo que evita la necesidad de dependencias externas.

En los siguientes apartados se darán a conocer los componentes o módulos de MicroStack, sus princiàñes características y diferencias con Openstack. Además de la aplicabilidad en entornos empresariales y de desarrollo.

2.3.1.- MICROSTACK: CLI

El primer componente para presentar es la CLI de MicroStack, una interfaz de línea de comandos basada en OpenStack que proporciona una capa de abstracción sobre los distintos componentes y sus respectivas APIs. Esto permite gestionar instancias, imágenes, redes y gestionar almacenamiento sin la necesidad de interactuar directamente con los endpoints, simplificando la administración y automatización de recursos. [3]

La configuración de la CLI se realiza mediante archivos en formato YAML o scripts en Shell, en los que se definen variables de entorno para establecer parámetros clave de operación. Esta metodología permite la reproducción de configuraciones, facilitando la gestión de despliegues y la integración con herramientas de automatización.

```
# Variables de autenticación para la CLI de MicroStack
export OS_USERNAME=admin
export OS_PASSWORD=admin
export OS_AUTH_URL=http://10.20.20.1:5000/v3
export OS_PROJECT_NAME=microstack
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_REGION_NAME=microstack
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
export OS_COMPUTE_API_VERSION=2

# Comprobar que las variables están definidas
echo "Variables de entorno de MicroStack cargadas."
```

Figura 2.2.- Variables entorno necesarias para la CLI

En la guía de instalación del siguiente apartado se verá cómo se configura y se usa la CLI.

2.3.2.- MICROSTACK: KEYSTONE

Keystone es el componente de MicroStack responsable de la gestión de identidad y autenticación. Su función principal es administrar los accesos y autorizaciones de usuarios y otros servicios de la nube, permitiendo la interacción segura entre los distintos componentes de MicroStack. [4]

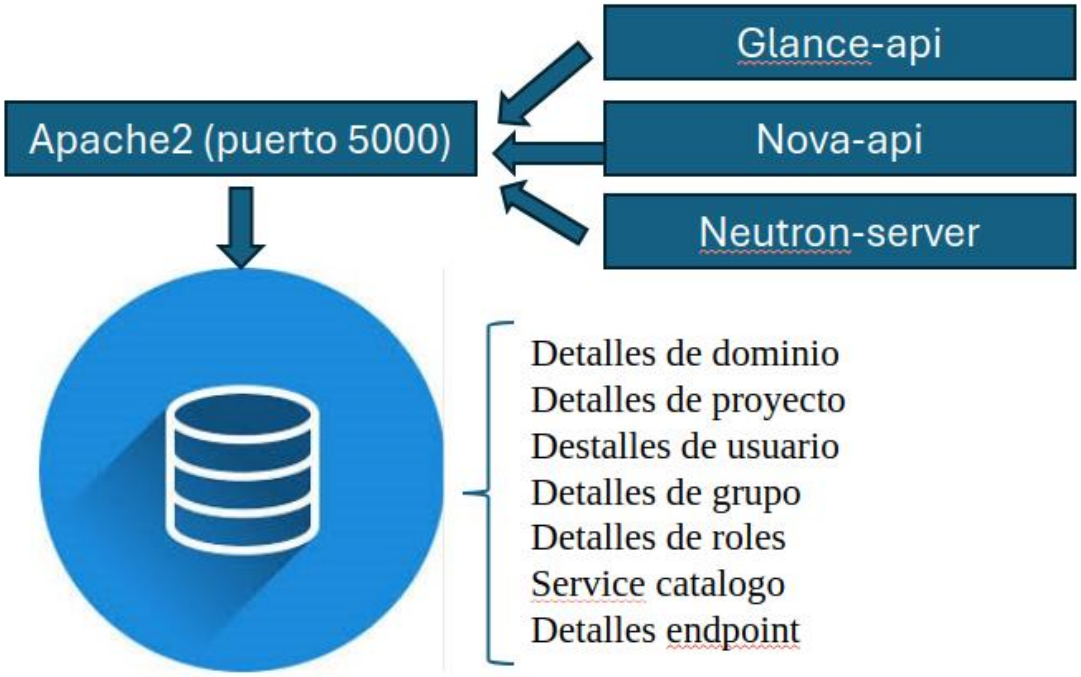

Figura 2.3- Componentes Keystone

2.3.2.1.- Dominios, Proyectos y Usuarios

Los dominios, son contenedores lógicos que agrupan usuarios, grupos y proyectos. En MicroStack, por defecto, solo existe el dominio default, sin posibilidad de crear otros adicionales. A modo de analogía, los dominios pueden entenderse como el espacio de trabajo de distintas empresas dentro de la infraestructura. [5][8]

Los proyectos (Tenants), son el entorno donde se despliegan y gestionan los recursos. En una plataforma cloud administrada, un dominio podría representar una empresa, y dentro de este se pueden crear múltiples proyectos según necesidades específicas. Por ejemplo, un proyecto puede contener la infraestructura de una plataforma web (servidores de bases de datos, backend y frontend), mientras que otro puede estar destinado a instancias para los empleados. [5]

Por último, los usuarios representan a los individuos o entidades que interactúan con MicroStack, almacenados por defecto en la base de datos de Keystone. Sin embargo, existen otras opciones de autenticación, como bases de datos externas. Un usuario se define dentro de un dominio y puede pertenecer a múltiples proyectos, con permisos diferenciados en cada uno.

2.3.2.2.- Roles y Grupos

Los roles son etiquetas asignadas a usuarios o grupos que determinan sus permisos dentro de un proyecto. Los roles son globales a nivel de dominio, pero su aplicación es específica para cada proyecto, lo que permite que un mismo usuario tenga diferentes permisos en distintos entornos.

Los grupos son una abstracción que facilita la gestión de permisos a múltiples usuarios. Un grupo puede contener varios usuarios y asignárseles un rol común dentro de un dominio o proyecto. [5][6][8]

2.3.2.3.- Catálogo de Servicios y Tokens de Autenticación

El catálogo de servicios funciona como un directorio de servicios de MicroStack, proporcionando un listado de endpoints accesibles según su tipo: [7]

- Públicos: Expuestos a internet.
- Internos: Comunicación entre componentes internos.
- Administrativos: Acceso exclusivo para tareas de gestión.

Los tokens son credenciales temporales que prueban la autenticidad y autorización de una solicitud. Un token es un hash con información sobre el usuario, el proyecto y los roles asignados. El flujo de autenticación sigue estos pasos:

- El usuario se autentica en Keystone.
- Keystone valida las credenciales y genera un token.
- El usuario utiliza este token para acceder a otros servicios de MicroStack.
- Cada servicio valida el token con Keystone antes de conceder acceso.

```
# Obtener TokenAuth
curl -X POST http://<IP_KEYSTONE>:5000/v3/auth/tokens \
     -H "Content-Type: application/json" \
     -d '{
           "auth": {
               "identity": {
                   "methods": ["password"],
                   "password": {
                       "user": {
                           "name": "admin",
                           "domain": { "id": "default" },
                           "password": "adminpassword"
                       }
                   }
               },
               "scope": {
                   "project": {
                       "name": "demo",
                       "domain": { "id": "default" }
                   }
               }
           }
       }'
```

Figura 2.4.- Petición HTTP a la API de Keystone para obtener token

2.3.2.4.- Principales Funcionalidades de Keystone [10]

- Autenticación de usuarios y servicios mediante credenciales, tokens o integración con terceros, como LDAP de Active Directory.
- Gestión de autorizaciones y roles mediante un modelo RBAC (Role-Based Access Control), con roles predefinidos como:
    - admin: Acceso total a la plataforma.
    - member: Permisos estándar de operación.
    - reader: Permiso exclusivo de lectura.
    - Los roles están aislados por proyecto, lo que permite granularidad en la administración de permisos.

- Integración con sistemas de autenticación externos, como SAML y OpenID Connect, permitiendo a los usuarios iniciar sesión con credenciales de Google, Microsoft u otros proveedores, si está configurado.

2.3.3.- MICROSTACK: GLANCE

Glance es el servicio de MicroStack encargado de la gestión y almacenamiento de imágenes de sistema operativo, permitiendo su uso como plantillas para el despliegue de máquinas virtuales. Sus funciones principales incluyen:

- Registro, descubrimiento y almacenamiento de imágenes de sistemas operativos.
- Exposición de una API REST para que otros servicios, como Nova, puedan acceder y utilizar las imágenes en la creación de instancias.
- Compatibilidad con múltiples sistemas de almacenamiento para la gestión de imágenes.

Glance se integra con otros componentes de MicroStack, como Nova y Keystone, para garantizar que únicamente usuarios autorizados puedan administrar las imágenes. [11]

Una imagen en Glance es un archivo utilizado como plantilla para la creación de instancias. Contiene el sistema operativo, las configuraciones y, opcionalmente, aplicaciones preinstaladas. Por ejemplo:

- Una imagen básica de Ubuntu Server 22.04 LTS sin configuraciones adicionales.
- Una imagen personalizada con Ubuntu Server 22.04 LTS + Apache2 preinstalado + reglas de firewall configuradas.

Las imágenes pueden contener metadatos adicionales utilizando la opción --property, incluyendo:

- architecture: Arquitectura del sistema operativo.

- min_disk: Tamaño mínimo de disco requerido.
- min_ram: Memoria RAM mínima en MiB.
- os_type: Tipo de sistema operativo.
- environment: Propósito de la imagen.
- application: Aplicaciones preinstaladas.

Estos metadatos pueden definirse al crear la imagen o modificarse posteriormente.

Glance admite varios formatos de imagen, cada uno con sus características específicas: [12]

- QCOW2 (QEMU Copy-On-Write v2): Formato optimizado para KVM, con soporte para compresión y snapshots.
- RAW: Imagen sin procesar, contiene datos binarios en bruto. Es el más rápido, ya que no requiere manipulación de datos, pero carece de funciones avanzadas como snapshots o compresión.
- VMDK (Virtual Machine Disk): Originalmente desarrollado por VMware, ahora es compatible con hipervisores como VirtualBox.
- VHD (Virtual Hard Disk): Formato utilizado por Microsoft Hyper-V y Azure.
- VDI (Virtual Disk Image): Formato nativo de Oracle VirtualBox.
- ISO: Imagen de disco óptico que contiene tanto el sistema de archivos como el sistema operativo.

La obtención de las imágenes puede ser:

```
# Descargar imagen ubuntu en formato qcow2
wget https://cloud-images.ubuntu.com/focal/current/focal-server-cloudimg-amd64.img -O ubuntu-20.04.qcow2
```

Figura 2.5.- A través de repositorios oficiales. [14]

```
# Convertir imagen RAW a qcow2, puede ser a la inversa
qemu-img convert -f raw -O qcow2 /ruta/disco_MV.raw imagen_destino.qcow2
```

Figura 2.6.- conversión de imagen de una instancia raw a qcow[14]

O realizando un snapshot de la máquina virtual en funcionamiento, esta opción almacena directamente la imagen en Glance

```
# Almacena el estado actual de la MV identificada por UUID
microstack.openstack server image create --name "Nombre_Snapshot" <UUID>
```

Figura 2.7.- Snapshot transformado a imagen [14]

```
# Registrar una imagen
microstack.openstack image create "Ubuntu-20.04" --file ubuntu-20.04.qcow2
--disk-format qcow2 --container-format bare --public --property architecture=x86_64
--property min_disk=20 --property environment=production

# Modificar los metadatos
microstack.openstack image set <IMAGE_ID> --property environment=development
--property application=nginx

# Listar imágenes disponibles
microstack.openstack image list

# Obtener detalles de una imagen
microstack.openstack image show <IMAGE_ID>

# Eliminar una imagen
microstack.openstack image delete <IMAGE_ID>
```

Figura 2.8.- Comandos básicos de Glance para usar en la CLI [14]

2.3.4.- MICROSTACK: NOVA

Nova es el componente encargado de la gestión del ciclo de vida de las instancias en MicroStack. Utiliza KVM (Kernel-based Virtual Machine) como hipervisor por defecto,

proporcionando virtualización a nivel de kernel. En una implementación más amplia de OpenStack, también es posible utilizar otros hipervisores como Xen, VMware ESXi o Microsoft Hyper-V. [15]

Nova administra los recursos de hardware del nodo anfitrión, incluyendo CPU, memoria RAM y redes, asignándolos a las instancias en ejecución. Además, se integra con otros servicios clave de MicroStack:

- Neutron, para la gestión de redes y conectividad.
- Glance, para el almacenamiento y aprovisionamiento de imágenes de sistema operativo.
- Keystone, para la autenticación y autorización de usuarios y servicios.

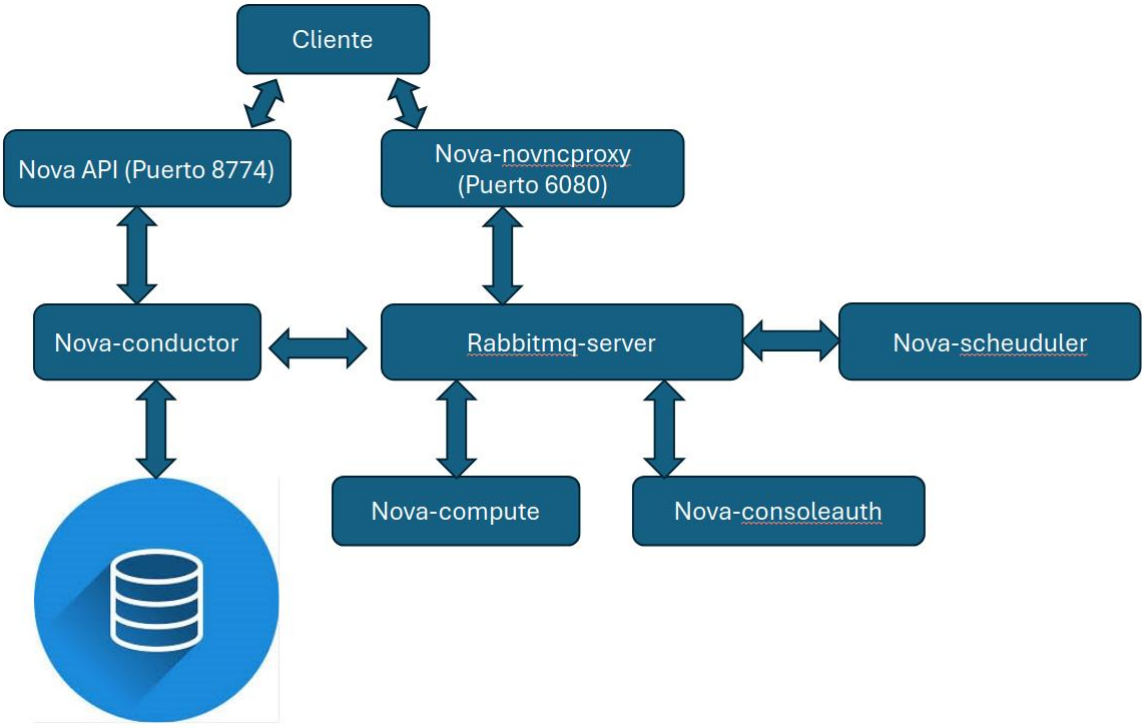

Figura 2.9.- Esquema servicios y comunicaciones del componente Nova

Nova está compuesto por varios subcomponentes con funciones especializadas: [16]

- Nova Scheduler: Se encarga de seleccionar el nodo de cómputo más adecuado para alojar nuevas instancias. Utiliza filtros de disponibilidad de recursos y

ponderadores (weighers), evaluando parámetros como la carga del nodo, afinidad de instancias dentro del mismo nodo o red, y la distribución eficiente de los recursos.

- Nova API: Es la interfaz RESTful principal, que actúa como intermediario entre las solicitudes de los clientes (HTTP) y la infraestructura de Nova. Orquesta las interacciones entre los distintos componentes de Nova y otros servicios de MicroStack.
- Nova Conductor: Gestiona las peticiones a la base de datos de Nova, reduciendo la exposición directa de los nodos de cómputo a la base de datos y disminuyendo el riesgo de corrupción de datos en operaciones críticas.
- Nova Database: Almacena información sobre el estado de las instancias, recursos asignados, nodos de cómputo y otras métricas clave para la administración de la nube.
- Nova Compute: Se encarga de la interacción directa con el hipervisor KVM, ejecutando operaciones sobre las instancias como creación, arranque, apagado, eliminación y migración.
- Nova Console Proxy: Proporciona acceso remoto a las instancias a través de protocolos como VNC (Virtual Network Computing) y RDP (Remote Desktop Protocol), permitiendo a los administradores y usuarios conectarse a sus máquinas virtuales de forma segura.

Los distintos componentes de Nova se comunican mediante una combinación de protocolos para garantizar una interacción eficiente y resiliente: [17][15]

- HTTP/REST: Utilizado para la comunicación entre clientes y servicios web. Nova expone su API mediante peticiones HTTP, permitiendo la interacción con los recursos a través de endpoints RESTful.
- AMQP (Advanced Message Queuing Protocol): Protocolo diseñado para sistemas distribuidos basado en un modelo publicación/suscripción, que permite una comunicación asíncrona y resiliente mediante el uso de colas de mensajes.

- RabbitMQ: Implementación de AMQP, que gestiona la transmisión de mensajes entre los distintos servicios de Nova, asegurando la entrega de tareas en sistemas distribuidos.
- RPC (Remote Procedure Call): Permite la ejecución remota de procedimientos en diferentes componentes de Nova, simulando una interacción local y facilitando la comunicación entre los servicios.

Componente	Protocolo Entrante (De quién)	Protocolo Saliente (Con quién)
Nova API	HTTP/REST (Clientes, Horizon)	AMQP (Nova Scheduler), HTTP/REST (Keystone, Glance, Neutron, Placement)
Nova Scheduler	AMQP (Nova API)	AMQP (Nova Compute), HTTP/REST (Placement)
Nova Compute	AMQP (Nova Scheduler, Conductor)	RabbitMQ (Nova Scheduler), HTTP/REST (Glance, Neutron), libvirt (KVM)
Nova Conductor	RPC/AMQP (Nova Compute, Nova API)	Database Queries (Nova Database)
Nova Console Proxy	HTTP/VNC/SPICE (Clientes, Horizon)	Directo (Máquinas virtuales)
Nova Database	RPC/AMQP (Nova Conductor)	-

Tabla 2.2.- Servicios Nova y sus endpoints

En MicroStack las instancias se crean a partir de los flavor, una plantilla para estandarizar los recursos que tendrá a disposición una instancia, se crean a nivel de dominio y son compartidos entre proyectos. Se compone de:

- vCPUs: Número de CPUs virtuales asignadas.
- RAM: Cantidad de memoria asignad en MB.
- Disco: Espacio de almacenamiento asignado en GiB.

```
# Creación flavor
microstack.openstack flavor create <nombre_flavor> --vcpus <num_cpu>
--ram <ram_MB> --disk <disco_GB>
```

Figura 2.10.- Creación flavor mediante CLI

Para la creación de las maquinas podemos optar dos opciones, la interfaz web Horizon (en la guía de instalación se verá cómo) y a través de la línea de comandos (CLI) que por detrás realiza peticiones HTTP a la API de Nova.

Para el caso de CLI es necesario tener acceso a la terminal y cargar las variables de entorno un usuario con rol member o admin (en el apartado MicroStack: CLI se aclara como iniciar la terminal) en el proyecto que queramos crear la instancia. [19]

```
# Creación máquina virtual
microstack.openstack server create --name vm_ejemplo --image
ubuntu-20.04 --flavor f_ejemplo --network demo
```

Figura 2.11.- Creación máquina virtual desde CLI

Es obligatorio especificar el nombre, la imagen, el flavor y, por último, la red (ver Neutron). Por otro lado, se puede indicar de forma opcional la clave ssh para acceder a la instancia --key-name <nombre_clave> (previamente creada MicroStack.OpenStack keypair create --public-key <ruta_destino> <nombre_clave>) o --user-data <archivo> para cargar un script que ejecutar al momento de creación de la instancia (útil para configuración Cloud Init como usuarios y contraseñas entre otros).

Cuando ejecutamos el comando de creación de la instancia desde terminal se hace, automáticamente, una petición a Keystone para obtener el token de autenticación y este es usado para enviar la petición a Nova API de creación de la máquina. Una vez obtenido ese token podremos lanzar la petición a la API:

```
# Creación instancia
curl -X POST http://<IP_NOVA>:8774/v2.1/servers \
     -H "Content-Type: application/json" \
     -H "X-Auth-Token: <token_obtenido>" \
     -d '{
           "server": {
               "name": "vm_ejemplo",
               "imageRef": "ubuntu-20.04",
               "flavorRef": "f_ejemplo",
               "networks": [{"uuid": "demo"}]
           }
       }'
```

Figura 2.12.- Creación de una instancia mediante la petición al endpoint de Nova API

Después de esto Nova API reenvía la solicitud a Nova Scheduler el cual escoge un nodo donde crear la máquina y Nova Compute ejecuta la orden de crear la máquina virtual a través de libvirt, el cual traduce el archivo XML generado por Nova a comandos KVM.

2.3.5.- MICROSTACK: NEUTRON

Toda esta sección es una expresión de lo comprendido en la documentación de Neutron, páginas de internet, youtube, udemy y la asignatura de la carrea de redes de computadores. [20][21][22][23][24][25]

Es el componente de MicroStack que permite administrar las redes para las máquinas virtuales, es clave a la hora de proporcionar conectividad entre las instancias, de dentro hacía afuera y viceversa. Ofrece redes como servicio (NaaS) de forma escalable y dinámica. Las funciones de Neutron son las propias de un administrador de red:

- Gestiona la definición por parte de los usuarios de redes aisladas y subredes en cada proyecto (Multitenant).

- Tiene funciones de enrutamiento y NAT (SNAT y DNAT), proporciona una puerta de enlace para la conexión entre redes públicas y privadas.
- Permite configurar reglas para los paquetes entrantes y salientes (grupos de seguridad).

Para este apartado, que sin duda es el más complejo de todos, voy a empezar explicando la topología de una red convencional para luego introducir la MicroStack.

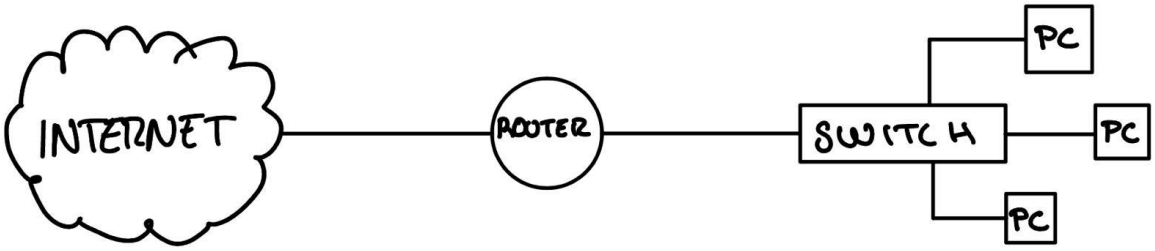

Figura 2.13.- Topología de red convencional

En una topología estándar como la de la imagen nos encontramos con la capa 3 del modelo OSI, la capa red (IP) y con la capa 2, capa enlace de datos (MAC). Los paquetes llegan de internet a la IP pública del router, el cual traduce mediante NAT (Network Address Translation) y PAT (Port Address Translation) a las IP privadas de la red interna, envía un paquete al switch el cual deshace para obtener la trama e identificar la dirección MAC. Esa dirección MAC estará asociada a una de sus NIC físicas, por donde envía el paquete. De igual forma el PC puede enviar información al exterior identificando en su tabla ARP la dirección MAC del Gateway (el router en este caso), encapsula la información, se la manda al switch indicando la MAC de destino (la de la Gateway) y este envía por la NIC del router la información, el router la recibe usa NAT/PAT a la inversa y cambia la IP privada por la suya pública y manda el mensaje al siguiente punto de internet.

MicroStack en sus versiones recientes y en la que está fundamentado este trabajo tiene los siguientes componentes

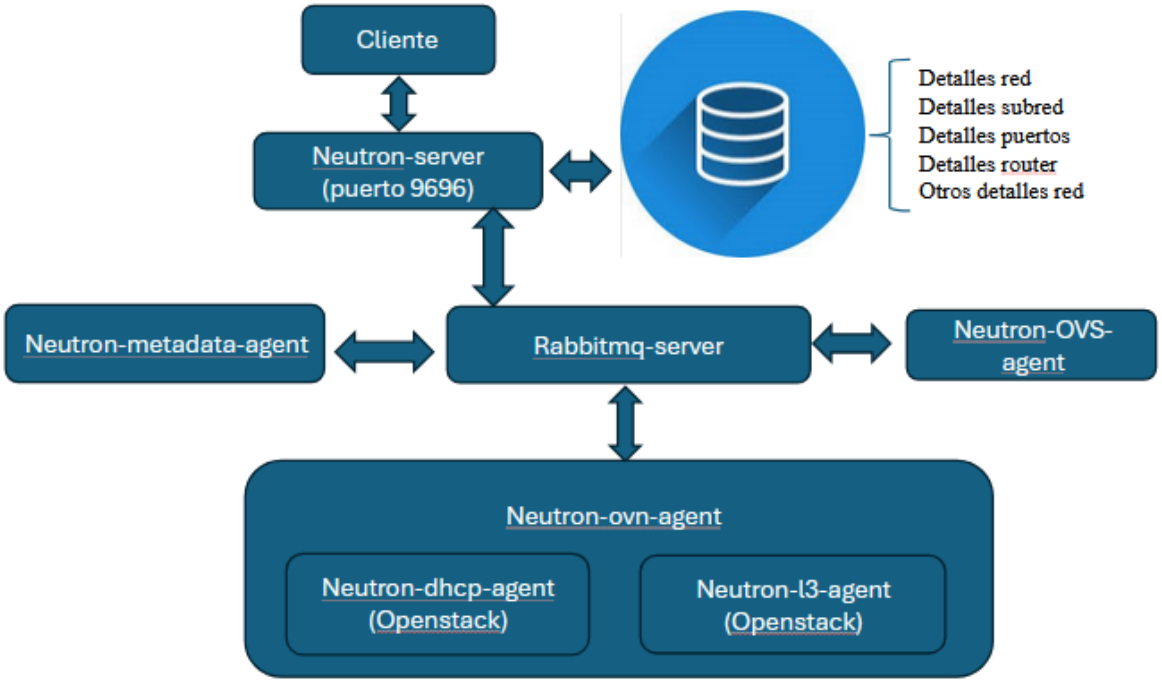

Figura 2.14.- Arquitectura Neutron

2.3.5.1.- Componentes Neutron

En OpenStack tradicional, Neutron emplea agentes como Neutron-DHCP-Agent y Neutron-L3-Agent para proporcionar DHCP y funciones de enrutamiento. En muchas infraestructuras, estos agentes siguen siendo el estándar. Sin embargo, las últimas versiones de MicroStack utilizan OVN (Open Virtual Network), que integra enrutamiento, NAT y DHCP en un solo componente, suprimiendo la necesidad de los agentes clásicos.

Open vSwitch es un switch virtual programable que implementa redes definidas por software. Gestiona el tráfico de capa 2 entre instancias virtualizadas, tanto dentro de un mismo host como entre hosts distintos. En MicroStack, OVS proporciona dos puentes principales:

- br-int (interno), switch interno que conmute el tráfico de las redes lógicas de capa 2. Todos los puertos virtuales de las instancias se asocian a br-int mediante interfaces tap.

- br-ex (externo), switch externo que conecta la red externa virtual de MicroStack con la NIC física del nodo. A su vez, los routers virtuales (gestionados por OVN) se conectan aquí para salir a la red física.

Las conexiones entre br-int y br-ex se hacen mediante un **puerto patch**. Este canal de capa 2 permite el tráfico entre redes internas y externas, pero el NAT y enrutamiento lo hace el router virtual de OVN.

Aunque OVS está presente dentro de MicroStack, no siempre se ve como un paquete instalado directamente en el sistema operativo del host, sino que se incluye en la propia implementación de MicroStack.

OVN es una extensión de OVS que ofrece enrutamiento de capa 2 y 3, NAT (SNAT y DNAT), gestión de IP flotantes y servidor DHCP. OVN se integra directamente con OVS y le indica cómo procesar paquetes a nivel de capa 2 y 3.

- SNAT: reemplaza la IP privada de origen por la IP de la interfaz de router o la IP flotante de la instancia.
- DNAT: reemplaza la IP flotante de destino por la IP privada de la instancia.

OVN se encarga así de todas las funciones de red (L2, L3, DHCP, NAT) que en OpenStack clásico recaían en agentes separados.

Type	Name
OVN Controller Gateway agent	ovn-controller
OVN Metadata agent	networking-ovn-metadata-agent

Figura 2.15.- Procesos OVN visto desde Horizon

2.3.5.2.- Redes lógicas y subredes

Las redes lógicas en MicroStack son una abstracción de una red de capa 2, se representa como un switch virtual y no tiene IP por sí misma, es solo un puente entre máquinas. Pueden llevar asociada una subred.

Las subredes de MicroStack, en cambio, definen una red IP con Gateway y rango de red, de ese modo los routers pueden conectarse mediante una interfaz y las máquinas mediante un puerto obteniendo una IP por DHCP (OVN) si está activo o creandolo para asociar a esa interfaz con una IP fija. Por lo general la primera IP disponible se reserva para el router como Gateway y la segunda para el servidor DHCP si está activo.

Los puertos son interfaces para asignar a instancias y controlar que IP va a recibir, se definen a nivel de red y pueden cambiarse entre máquinas. Las IP flotante se asocian a puertos existentes y es posible asociarles también un grupo de seguridad a modo de firewall.

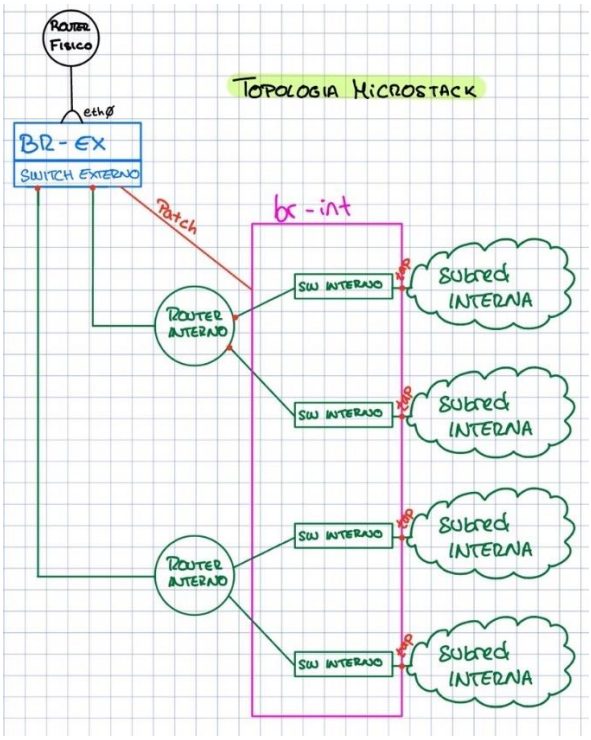

Figura 2.16.- Red MicroStack

Como capa de seguridad extra MicroStack posibilita crear grupos de seguridad y reglas para actuar a modo de firewall (IPtables o nftables), funciona en modo *whitelist* lo que quiere decir que por defecto todo esta denegado y solo permite lo que nosotros indiquemos. Los grupos de seguridad se conforman de reglas de seguridad para el tráfico entrante y saliente, para cada proyecto está el grupo de seguridad por defecto *default* que permite todo el tráfico hacia fuera, pero bloquea el entrante salvo ICMP (ping) y ssh. Las reglas pueden granular por protocolo, puertos, dirección del tráfico e IPs de origen y de destino. Más de un grupo de seguridad puede ser asignado a un puerto.

Si no asignas puerto ni grupo se asigna automáticamente el puerto por defecto y al puerto por defecto se le asigna el grupo por defecto.

Cuando creamos una instancia en MicroStack, le podemos asociar a una red interna mediante puerto preconfigurado o dejar que se le asigne uno. Esta máquina tiene conexión con las demás de su subred interna mediante br-int, y si hay un router conectado a la red, puede comunicarse con otras redes internas. Si envía un paquete a internet lo envía con su ip privada, pasa por br-int, quien comprueba con reglas Openflow si tiene una IP flotante asociada, si no la tiene llega al router virtual y aplica SNAT cambiando la IP de la instancia por la de su interfaz conectada a la red externa mediante br-ex, atraviesa br-ex y sale por la NIC física hacía el router físico y a internet. Hay un doble NAT.

Si queremos que la máquina pueda salir con su propia IP real, es necesario configurar las IP flotantes, estas IP son una reserva previa en el router físico e indicada en la configuración de la subred externa. El paquete seguiría el camino anterior, pero en el SNAT se cambiaría su IP privada por la IP flotante y al recibir respuesta se haría DNAT cambiando la IP flotante por la IP privada. En este método, la instancia no es consciente de que tiene una IP flotante asignada, las IP flotantes se asocian a puertos ya existentes para hacer reglas Openflow que relacionen IP flotante con IP privada. Pueden desasociarse de un puerto para asociarse a otro de otra máquina en otro proyecto.

El último caso es si conectamos la instancia directamente a la red externa, es decir a br-ex. El puerto conectado recibirá directamente una IP física y la instancia si tendrá esa interfaz definida en su SO.

En el apartado siguiente de despliegue, se verán estos casos de forma práctica y facilitará la comprensión.

```
root@nodo01:/home/carlos# ovs-vsctl show
d4ef4959-8be2-47f6-8ff0-e38fa70b366f
    Bridge br-int
        fail_mode: secure
        datapath_type: system
        Port tap57aea3b7-a2
            Interface tap57aea3b7-a2
        Port patch-br-int-to-provnet-39c4cae4-f10c-4428-ac34-91a8505c3b39
            Interface patch-br-int-to-provnet-39c4cae4-f10c-4428-ac34-91a8505c3b39
                type: patch
                options: {peer=patch-provnet-39c4cae4-f10c-4428-ac34-91a8505c3b39-to-br-int}
        Port tapc95fcc05-e3
            Interface tapc95fcc05-e3
        Port tapd32a7bc5-a9
            Interface tapd32a7bc5-a9
        Port tap78f43010-ff
            Interface tap78f43010-ff
        Port tap9799665d-5b
            Interface tap9799665d-5b
        Port tap1c7ec428-f0
            Interface tap1c7ec428-f0
        Port tap507f289a-d0
            Interface tap507f289a-d0
        Port br-int
            Interface br-int
                type: internal
        Port tapc90d1545-c4
            Interface tapc90d1545-c4
        Port tape7d44e7d-fd
            Interface tape7d44e7d-fd
    Bridge br-ex
        datapath_type: system
        Port br-ex
            Interface br-ex
                type: internal
        Port patch-provnet-39c4cae4-f10c-4428-ac34-91a8505c3b39-to-br-int
            Interface patch-provnet-39c4cae4-f10c-4428-ac34-91a8505c3b39-to-br-int
                type: patch
                options: {peer=patch-br-int-to-provnet-39c4cae4-f10c-4428-ac34-91a8505c3b39}
        Port enp5s0f1
            Interface enp5s0f1
    ovs_version: "2.14.0"
```

Figura 2.17.- Interfaces tap en br-int e interfaz patch entre br-int y br-ex

2.3.6.- HORIZON

Es el componente más visual de Micrsotack, es su interfaz web. Facilita un punto gráfico y centralizado para la administración de los servicios y componentes vistos antes. Es posible usarlo desde la parte de administrador o de usuario final, permitiendo interactuar de forma sencilla e intuitiva con el sistema sin necesidad de terminal o peticiones a APIs. [26]

Se puede utilizar para ejecutar la creación, modificación o eliminación de usuarios, asignar roles y crear proyectos. También incluye una visión gráfica del estado de los recursos disponibles (instancias, almacenamiento, memoria y redes).

Dispone de una interfaz amigable para el uso de usuarios no técnicos, con menús claros y organizados. Es posible facilitar acceso a cualquier usuario, pero adecuándose la interfaz a su rol de forma que un administrador pueda administrar y un lector pueda observar, por ejemplo. Es posible modificar la apariencia de la web para que sea más accesible para el usuario.

2.3.7.- MICROSTACK: EPHEMERAL STORAGE

Ephemeral Storage en MicroStack es el tipo de almacenamiento que se asigna automáticamente a las instancias en función del flavor elegido y que desaparece al eliminar la máquina virtual. A diferencia de los volúmenes persistentes de Cinder, este almacenamiento está vinculado directamente a la vida útil de la instancia y se almacena localmente en el nodo de cómputo, generalmente bajo el directorio /var/snap/microstack/common/lib/libvirt/images/. Su uso es ideal para cargas de trabajo temporales donde la persistencia de datos no es un requisito, como entornos de prueba, procesamiento temporal de datos o despliegues efímeros que pueden ser reemplazados sin necesidad de retener información entre reinicios.

Cuando lanzamos una instancia en MicroStack con un flavor, se le asigna automáticamente un espacio en disco igual al del flavor, el sistema operativo de la instancia se instala sobre este disco y en caso de reinicio no pasa nada, pero si es eliminada desaparecerán.

Por defecto estos discos se relacionan con el formato de imagen QCOW2 lo que proporciona compresión y capacidades avanzadas como snapshots, además de poder gestionar el almacenamiento en discos dedicados a este fin como SSD o configuraciones RAID. Se puede hibridar con almacenamiento persistente para datos que necesitemos almacenar de manera persistente.

Para muchos casos este tipo de almacenamiento es suficiente, pero es importante considerar sus limitaciones y pensar si nuestras necesidades exigen almacenamiento persistente. Para entornos de producción se recomienda complementar las instancias con volúmenes en Cinder. [53][54]

2.3.8.- MICROSTACK: CINDER

Cinder es el servicio de almacenamiento en bloque de OpenStack, y en MicroStack se encarga de proporcionar almacenamiento persistente para las instancias. A diferencia del almacenamiento ephemeral, que se elimina al destruir la máquina virtual, los volúmenes creados con Cinder permanecen en el sistema y pueden ser reutilizados, adjuntados a otras instancias e incluso respaldados o replicados. Su funcionamiento es similar al de un disco duro virtual, ya que permite asignar espacio de almacenamiento independiente del ciclo de vida de las instancias, facilitando la persistencia de datos y la administración flexible del almacenamiento en la nube.

En MicroStack Cinder permite adjuntar nuevos volúmenes a máquinas y desacoplarlos. Una de sus principales ventajas es la capacidad de desacoplar el almacenamiento del cómputo, lo que significa que una instancia puede ser eliminada sin

afectar los datos almacenados en un volumen. Además, los volúmenes pueden migrarse entre instancias, realizar snapshots para respaldos y soportar funcionalidades avanzadas como la replicación de datos.

El backend por defecto de Cinder en MicroStack suele estar basado en archivos almacenados localmente en el nodo de cómputo, pero en entornos más avanzados puede integrarse con sistemas de almacenamiento distribuidos como Ceph, LVM o NFS. Cada volumen creado en MicroStack se almacena en una ruta específica dentro del sistema de archivos y se expone a las instancias como si fuera un disco físico. Cuando un volumen se adjunta a una instancia, el sistema operativo dentro de la máquina virtual lo reconoce como un nuevo dispositivo de almacenamiento que puede ser formateado, montado y utilizado según sea necesario.

El uso de Cinder en MicroStack también permite realizar operaciones avanzadas como la creación de snapshots, que son copias en un punto determinado en el tiempo de un volumen. Estos snapshots pueden utilizarse para restaurar datos en caso de fallos, desplegar nuevas instancias con configuraciones predefinidas o replicar entornos de prueba sin necesidad de recrear la información desde cero. Además, los volúmenes pueden ser redimensionados dinámicamente, lo que permite aumentar su capacidad sin necesidad de recrear el almacenamiento o migrar los datos manualmente.

En entornos de producción, es común combinarlo con Ceph para obtener alta disponibilidad y redundancia, evitando así la dependencia de un único nodo de almacenamiento. En implementaciones más simples, el backend por defecto de MicroStack es suficiente para gestionar volúmenes locales sin requerir configuraciones adicionales.

En términos prácticos, el uso de Cinder dentro de MicroStack es fundamental para cualquier despliegue que requiera almacenamiento persistente. Su flexibilidad lo convierte en una solución eficiente para bases de datos, almacenamiento de aplicaciones críticas y entornos donde la información debe permanecer disponible incluso después de

que las instancias sean eliminadas. Aunque el almacenamiento ephemeral puede ser útil en escenarios donde la persistencia no es un requisito, Cinder proporciona una solución robusta para garantizar la integridad y disponibilidad de los datos en una infraestructura basada en MicroStack.[55][56]

3. Implementación de la infraestructura

Toda esta guía está basada en el autoaprendizaje del error, es decir, a partir de iniciar sesión en la web y con el conocimiento teórico fui investigando como crear, modificar y destruir los objetos. [29]

Para el despliegue de la infraestructura se tienen dos servidores idénticos con la siguiente configuración individual:

- RAM: 130 GiB
- CPU: AMD Opteron(tm) Processor 6172 x 2 sockets
- Almacenamiento: 300 GiB en RAID1 por hardware.

Primero se instalará el sistema operativo Ubuntu Server 22.04 LTS en ambos nodos y luego se instalará MicroStack a través de Snap. Para el primer nodo que lo configuraremos como el de control:

```
# Instalacion
root@nodo01:/home/carlos# snap install microstack --beta

# Iniciar el nodo como nodo de control
root@nodo01:/home/carlos# microstack init --auto --control
```

Figura 3.1.- Instalación y despliegue de MicroStack en el nodo de control

He realizado este proceso varias veces y en todas me daba un error, la falta de documentación de MicroStack me hizo imposible encontrar una solución en internet. Mi solución recomendada es volver a ejecutar el comando de iniciación y esta vez sí se instalará correctamente. En el nodo02 instalamos MicroStack con Snap de la misma forma, pero este lo inicializaremos como cómputo, en el nodo01 solicitamos un token para unir un nodo de cómputo.

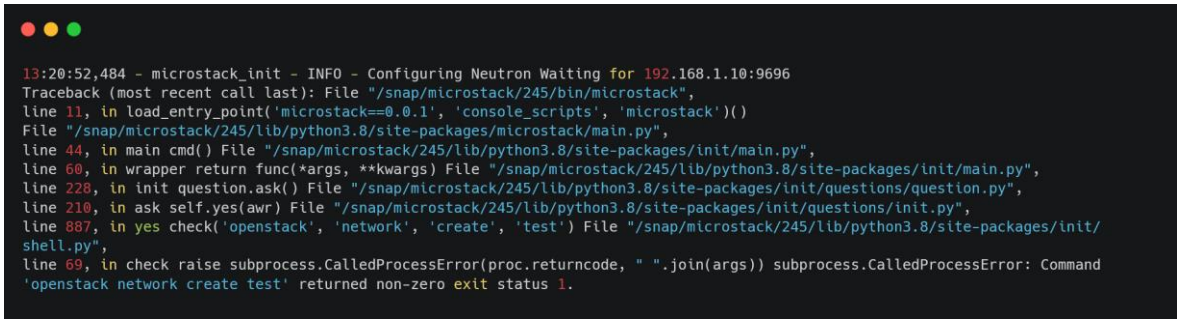

Figura 3.2.- Error común durante la instalación

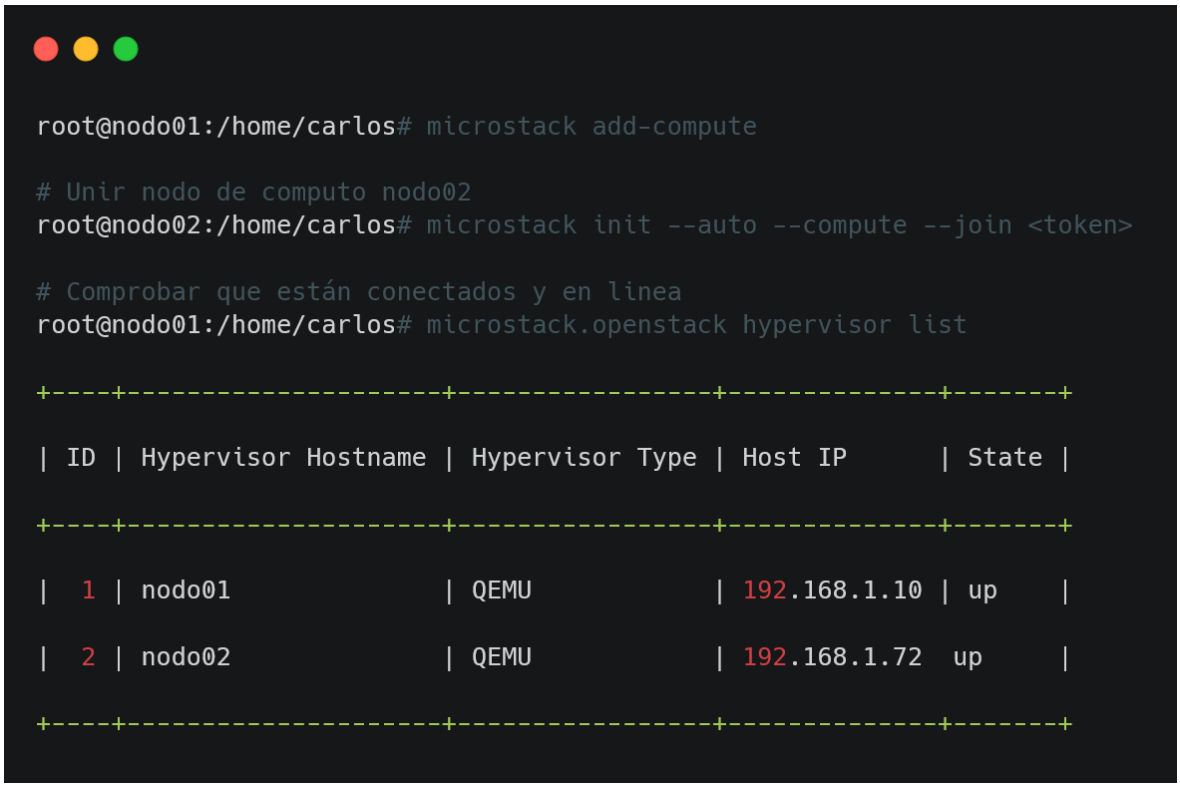

Figura 3.3.- Tabla con los nodos de cómputo disponibles

A partir de aquí realizaré los pasos a través de Horizon mayormente y lo documentaré con imágenes, para algunos pasos usaré la CLI que también explicaré como configurar. Antes de nada, he añadido unos alias a la terminal del nodo para hacer más simple los comandos.

```
sudo snap alias microstack.ovs-vsctl ovs-vsctl
sudo snap alias microstack.openstack openstack
```

Figura 3.4 .- Creación de alias

El primer paso es acceder a la IP del nodo de control a través del navegador y nos saldrá una alerta de que el certificado usado es auto firmado y puede ser inseguro, para nuestro entorno es indiferente, pero en producción es recomendable obtener un certificado legítimo de una CA legítima, se puede usar *certbot*. En la pantalla de login usaremos el usuario admin y para la contraseña ejecutaremos *root@nodo01:/home/carlos# sudo snap get MicroStack config.credentials.keystone-password* que nos devolverá un hash que usaremos como contraseña y que posteriormente puede ser cambiada.

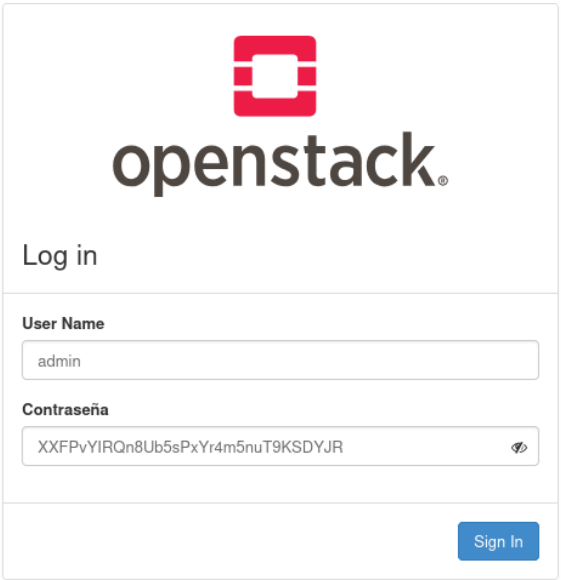

Figura 3.5.- Pantalla Login MicroStack

El proyecto que aparece es *admin* y existe otro creado por defecto que se llama *service.* Es muy importante no borrar ninguno de los dos para el correcto funcionamiento de la plataforma. Mas adelante crearemos el proyecto TFG para crear la infraestructura.

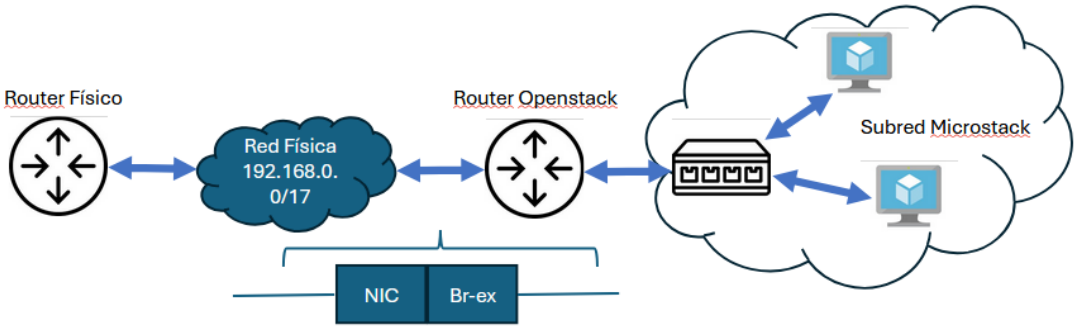

Figura 3.6.- Topología de red MicroStack

La imagen muestra la topología con la que trabajaremos en esta demo, para llegar a ella hay que, asociar la NIC física al bridge y a acoplarle la IP del nodo01 a dicho bridge. Además, en mi caso y lo que he podido contrastar en varios documentos de internet, MicroStack asigna un rango de IP 10.20.20.0/24 a br lo que en mi escenario no es adecuado ya que difiere de la red física. ¿Es posible mantener ese rango y hacerlo funcional? Si, pero tendríamos que añadir una regla de NAT en el nodo para que traduzca las IP de la red 10.20.20.0/24 a la red 192.168.0.0/17 o añadir una ruta estática en el router que indique que para alcanzar esa red el siguiente salto es la IP del nodo. Es una opción igualmente válida, yo he optado por la primera ya que me parece más limpia y no es necesario configurar nada fuera del nodo y de MicroStack.

Desde Horizon en la barra de la izquierda nos dirigimos *a Admin -> Network -> Router* y borramos el router existente, en *Admin -> Network -> Networks* borramos las redes que aparecen. Puede aparecer un error si hay instancias conectadas, pero no debería ser nuestro caso ya que no hemos creado instancias, en ese caso, borraríamos las interfaces que conectan las instancias a la red.

Ahora que está borrado recrearemos estas redes, empezando por la red externa. En el mismo menú donde las hemos borrado, hacemos click sobre *Create Network*, yo le he asignado los siguientes parámetros:

- Name: public
- Project: admin
- Provider Network type: Flat
  - Flat: Toda la red comparte el mismo rango sin segmentación.
  - VLAN: Segmentación por etiquetas VLAN (requiere hardware compatible).
  - VXLAN: Redes superpuestas usando encapsulación para aislamiento (necesita configuración avanzada).
  - GRE: Similar a VXLAN, pero con encapsulación GRE para túneles punto a punto (menos usado hoy en día).
  - Geneve: Más flexible que VXLAN, diseñado para redes escalables con encapsulación avanzada.
- Physical Network: 192.168.0.0/17
- Check en *Enable Admin State* (está creado y activado), *External Network* (indica que es una red externa, está conectada a br-ex) y *Create Subnet* (crear una subred asociada)
- Adicionalmente y es recomendable, seleccionar *shared*. De este modo la red será compartida entre proyectos.

Al marcar la casilla de crear subred el siguiente paso es definirla, le daremos un nombre, una dirección y rango de red (debe coincidir con la real) y la Gateway que debe ser la del router físico. Debemos desmarcar la opción de DHCP ya que el router físico tiene DHCP, además que generalmente se asignaran IP del pool mediante IP flotante.

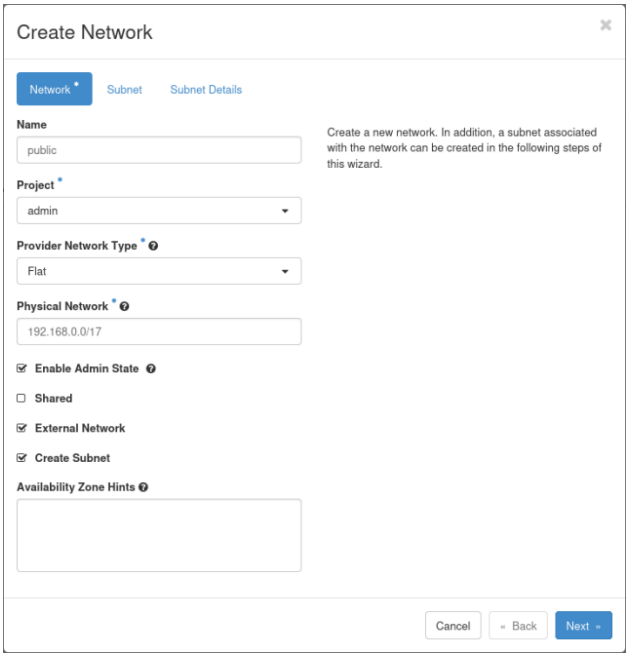

Figura 3.7.- Formulario creación Red Externa

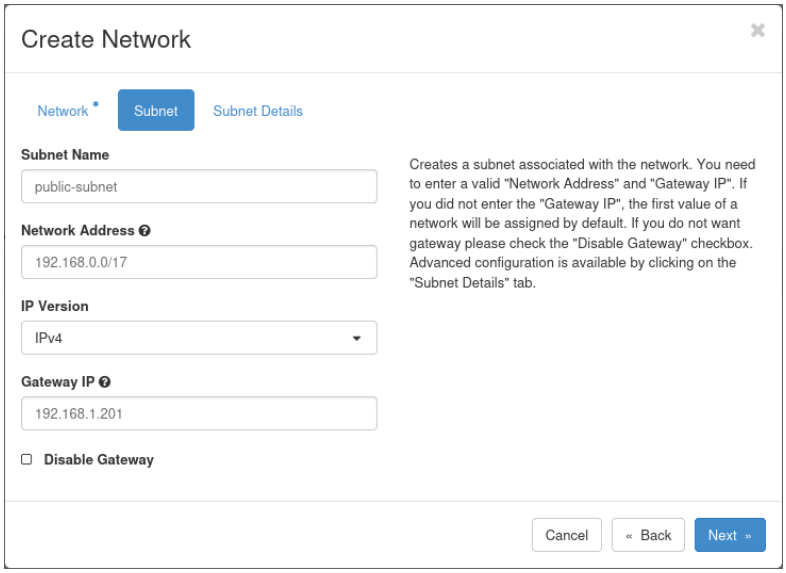

Figura 3.8.- Formulario Creación subnet

En este momento está definida la red que permitiría conectar una instancia directamente a la red externa. Igual que para esta red, definiré una red lógica con su subred asociada. Pero en este caso no marcaré la casilla de red externa ni de compartida, quiero que la red sea solo del proyecto, le asocio una subred, pero no indicamos puerta de enlace (será la interfaz del router y se asigna automáticamente como la primera) y si

activaremos el servidor DHCP, opcionalmente podemos indicar el rango de IPs que repartirá el servidor por si queremos reservar algunas para asignar mediante puertos.

Nos dirigimos a la casilla de *Routers* debajo de donde estamos para crear el primer router MicroStack y el que conectará las redes interna y externa que acabamos de definir. Ponemos un nombre, indicamos, en caso de estar conectado a una red externa (nuestro caso), a cuál y habilitamos SNAT. Ahora en la pestaña de *Netowork Topology* podemos ver las dos redes y el router conectado a la externa, si nos situamos sobre él se abre una ventana donde nos da información y además nos permite añadir más interfaces, creamos una para conectar el router a la red interna. Nos permiten indicar una IP si queremos, pero lo dejamos en blanco de manera que el router escoja la de la Gateway.

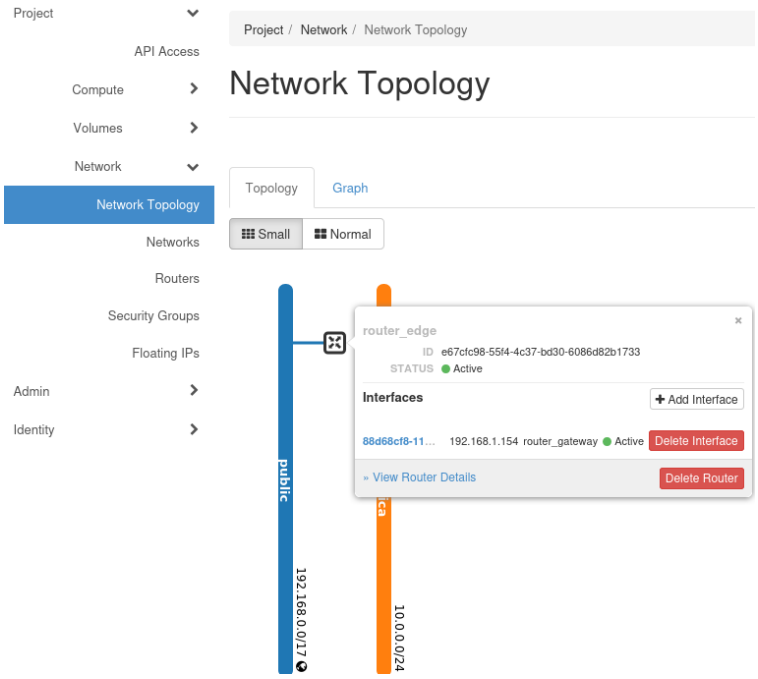

Figura 3.9.- Topología actual MicroStack

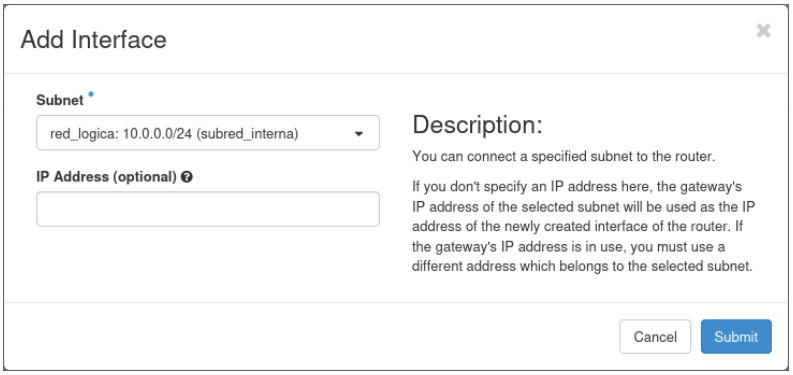

Figura 3.10.- Formulario añadir interfaz

Hasta ahora todo normal, pero falta la parte más delicada, asociar la NIC al bridge, debemos ir a la terminal del nodo01 que es donde se ejecuta Neutron y ejecutar lo siguiente, en mi caso es la 192.168.1.10/17 porque es la IP del nodo, si no se pudiera ejecutar desde la terminal física y estuvieses en remoto, se debe ejecutar todo en una línea para no perder la conexión. Es un error asignar a un bridge una NIC física sin eliminar la IP de la NIC y reasignarla al puerto. [27]

```
ovs-vsctl add-port br-ex ens0f5
ip addr flush dev enp3s4
ip addr add 192.168.1.10/17 dev br-ex
ip link set br-ex up
```

Figura 3.11.- Configuración br-ex host

Si vamos la ventana de *Project -> Compute -> Instance* podemos hacer click en *Launch Instance* y crear nuestra primera máquina para probar la config, por defecto hay una imagen *cirros* para usar. Asignamos nombre, elegimos la imagen cirros, escogemos un flavor de los preconfigurados por MicroStack, seleccionamos la red a la que conectar, security group dejamos el de por defecto y podemos crear un par de claves si queremos ese método de autenticación, en mi caso la creo y copio en un archivo en local la clave privada generada. Mientras se crea la máquina podemos asociarle una IP flotante

previamente creada o creada en el momento de asociar, las IP flotantes se asocian a puertos, en nuestro caso, el puerto de conectar cirros con la red interna.

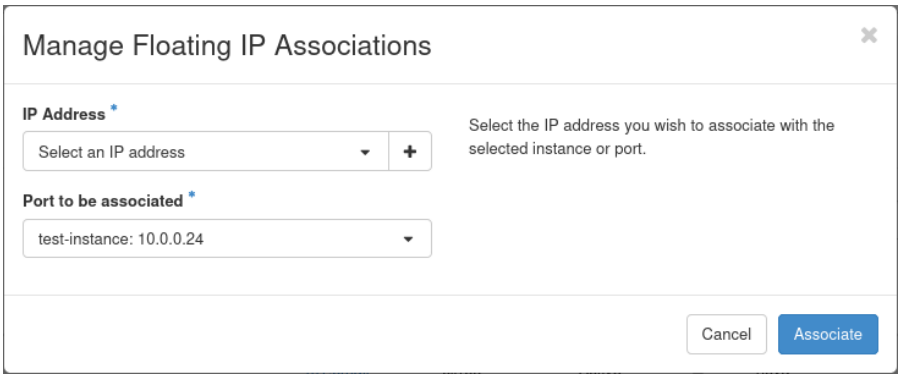

Figura 3.12.- Formulario añadir IP flotante

Cuando finaliza la creación se arranca sola y desde la terminal local de un equipo en nuestra red podremos conectarnos y comprobar la IP asignada de la red interna verificando que la red está en funcionamiento.

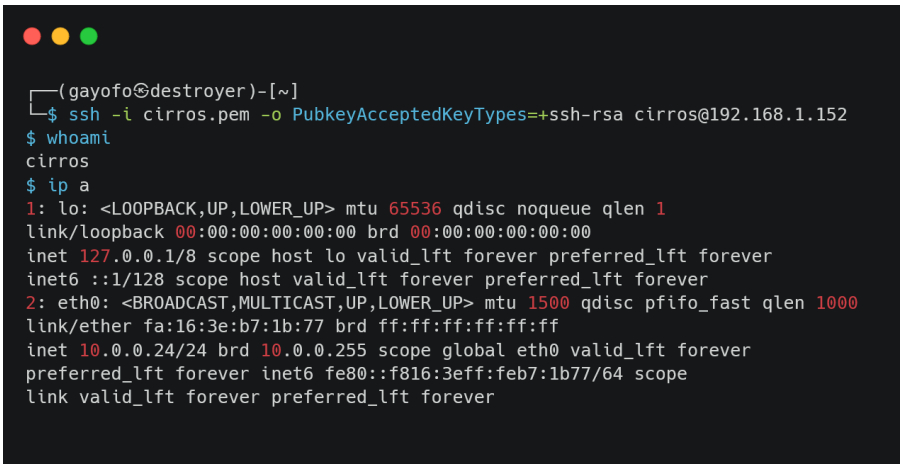

Figura 3.13.- Conexión máquina Cirros

Llegado este punto, voy a crear un proyecto y un usuario para simular un cliente al que ofrecemos nuestro servicio de hostear su infraestructura (literalmente IaaS, Infraestructura como servicio). En *Identity -> Project - > Create Project*.

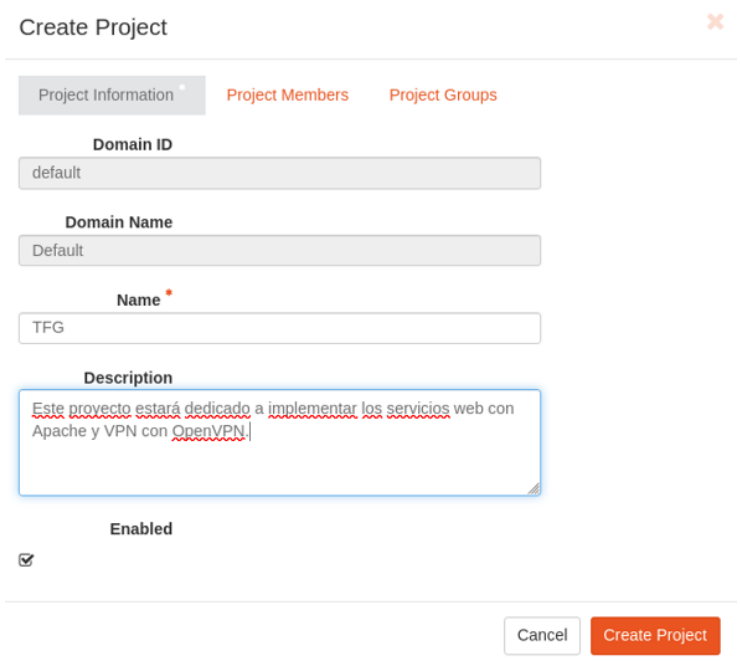

Figura 3.14.- Formulario crear proyecto

Como comenté en el apartado de Keystone, aunque Opensatck puede implementar distintos dominios para separar áreas de trabajo a más alto nivel que los proyectos, en MicroStack solo está disponible el dominio *default*. Lo demás es simplemente darle un nombre y una descripción de forma opcional (OJO que a mí por poner tildes me dio problemas y no supe solucionar por lo que decidí quitarlas). En la parte de añadir usuarios o grupos nos aparecerán los usuarios y grupos disponibles para añadir. En mi caso crearé el usuario a continuación y lo añadiré al proyecto en la creación.

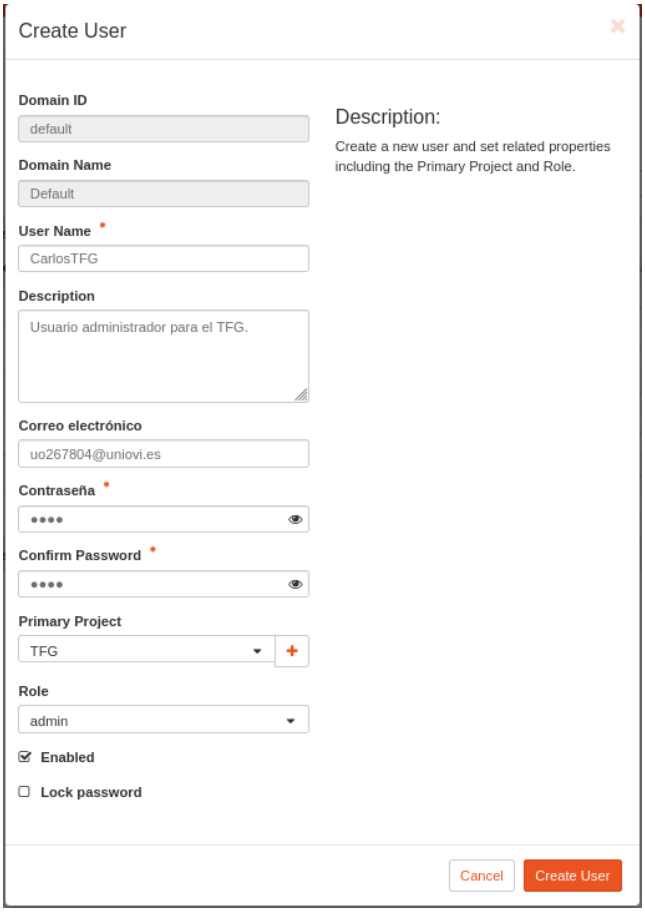

Figura 3.15.- Formulario crear usuario

Si queremos que el usuario actúe en otros proyectos podremos ir a la pestaña de proyectos, seleccionar un proyecto y en *Manage Members* podremos añadir usuarios seleccionando su rol.

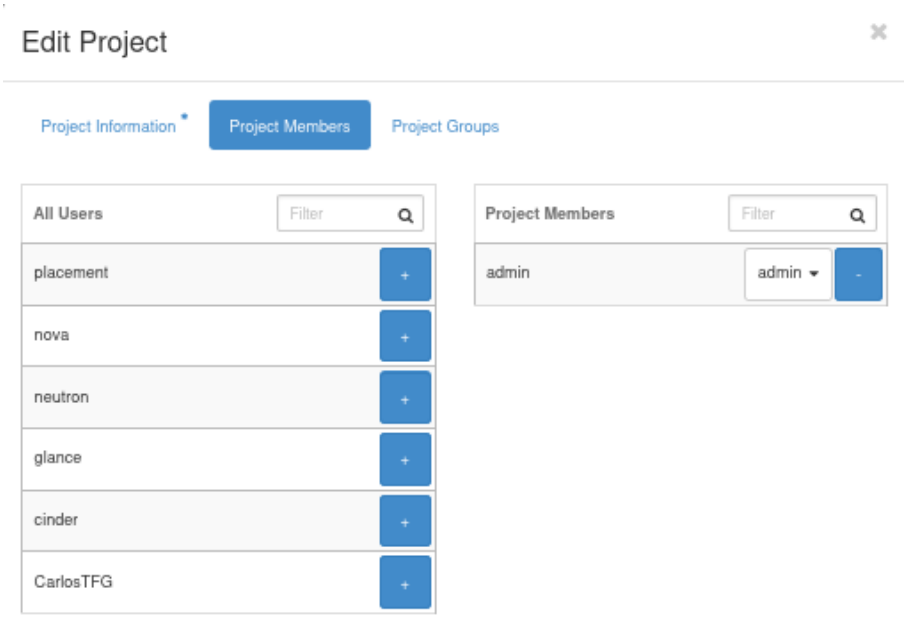

Figura 3.16.- Formulario añadir miembros proyecto

En la pestaña *Identity -> Projects* podemos localizar nuestro proyecto y desplegar un menú en el que aparecerá la opción *Edit Quotas*, este diálogo nos permitirá editar los recursos asignados a ese proyecto (RAM, vCPUs, elementos de red, espacio almacenamiento...).

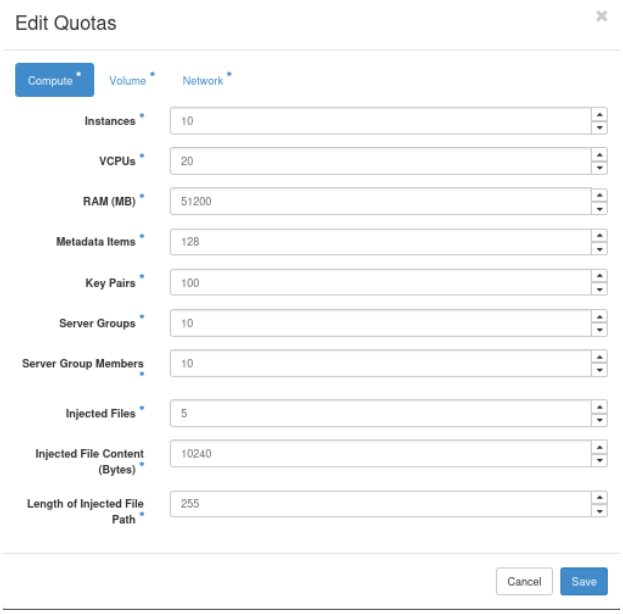

Figura 3.17.- Formulario editar cuotas

Por último, vamos a crear un flavor para indicar los recursos asignados a las máquinas que creemos. Tendremos el m.ubuntu para la máquina de propósito general y m.apache para el servidore web. En la pestaña *Admin -> Compute -> Flavor* clicando en el botón *Create Flavor* tendremos el dialogo de creación.

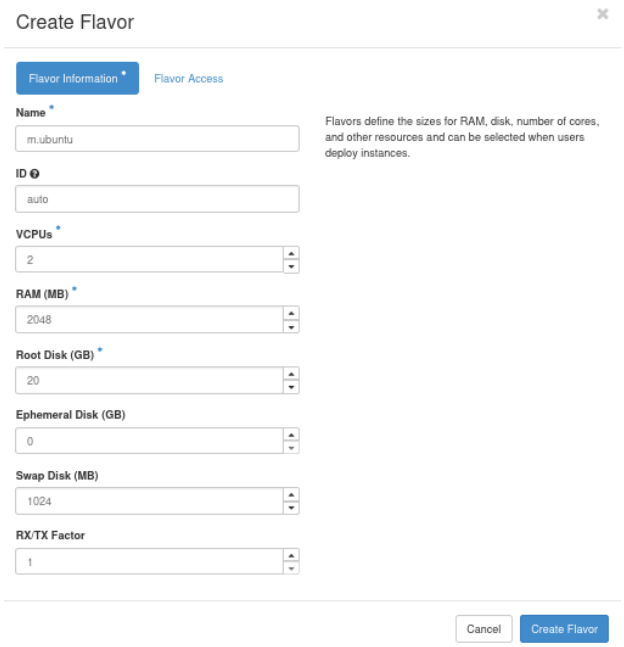

Figura 3.18.- Formulario crear flavor

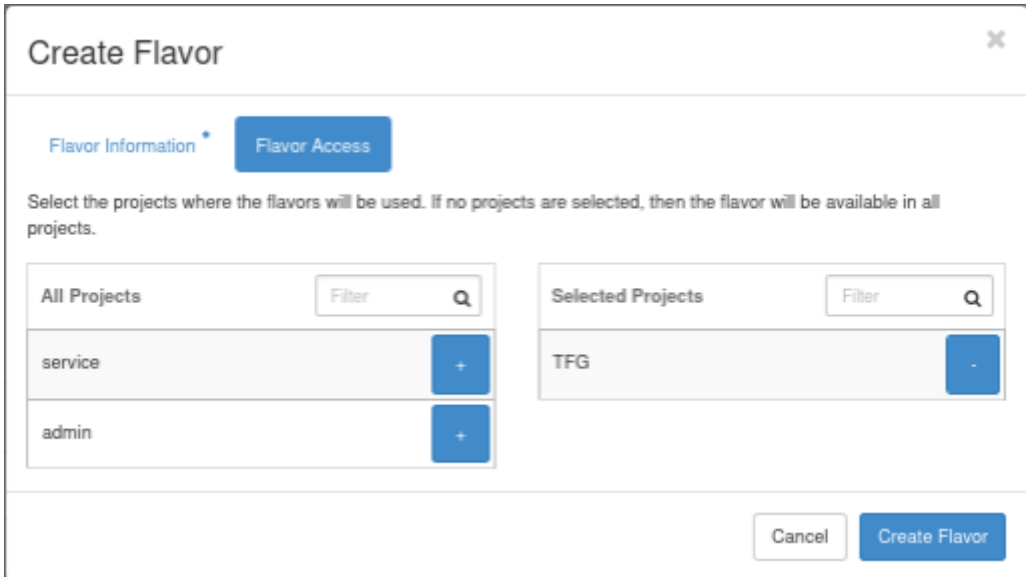

Figura 3.19.- Formulario añadir flavor a proyecto

En este punto ya tenemos un usuario CarlosTFG con un rol administrador en el proyecto TFG, los flavors necesarios para las máquinas están creados y hemos modificado los recursos. Ahora entraremos a Horizon con el usuario CarlosTFG para seguir con los pasos de cargar una imagen, crear instancias y proporcionar conexión.

Vamos a configurar una red interna y un router ya que los creados desde el usuario admin están en su proyecto y no los hemos compartido, veremos que el soporte multi-tenant permite la creación de redes y routers con la misma topología en diferentes proyectos. Replico lo ya hecho anteriormente para las redes internas, la externa y su subred si son compartidas.

Crearé un grupo de seguridad para el acceso a la máquina de Apache ya que va a estar expuesta al exterior es recomendable añadir esta capa extra de seguridad, aunque por encima tengamos firewalls. En *Network -> Security Group* seleccionamos en *Create Security Group*

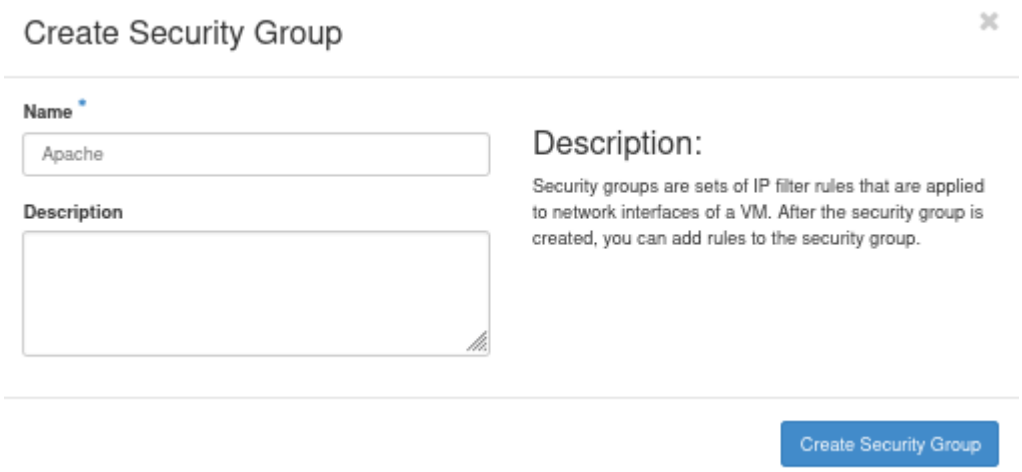

Figura 3.20.- Formulario crear grupo de seguridad

Ahora seleccionamos en *Manage Rules* y empezamos a crearlas haciendo clic en *Add Rule*. Yo crearé para permitir ssh (TCP) y ping (ICMP) desde la red interna (10.0.1.0/24) y externa (192.168.0.0/17) además de permitir acceder a puerto 80 y 443 desde cualquier IP. En sentido hacía afuera permito todo TCP y todo ICMP.

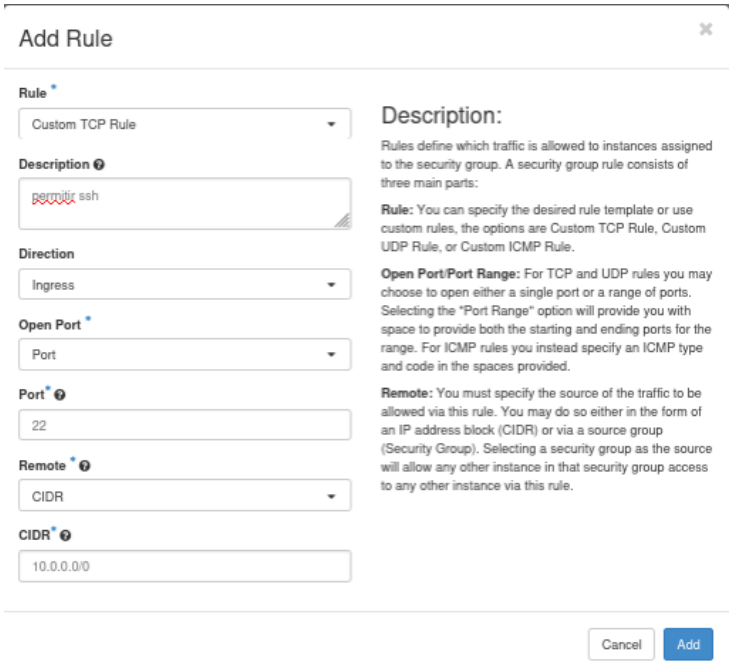

Figura 3.21.- Formulario añadir regla a grupo de seguridad

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix
Egress	IPv4	Any	Any	0.0.0.0/0
Egress	IPv4	ICMP	Any	0.0.0.0/0
Egress	IPv4	TCP	Any	0.0.0.0/0
Egress	IPv6	Any	Any	::/0
Ingress	IPv4	ICMP	Any	10.0.1.0/0
Ingress	IPv4	ICMP	Any	192.168.0.0/17
Ingress	IPv4	TCP	22 (SSH)	10.0.1.0/24
Ingress	IPv4	TCP	22 (SSH)	192.168.0.0/17
Ingress	IPv4	TCP	80 (HTTP)	0.0.0.0/0
Ingress	IPv4	TCP	443 (HTTPS)	0.0.0.0/0

Figura 3.22.- Lista de reglas grupo de seguridad Apache

De la misma forma he creado otro grupo para la máquina genérica que permite todo TCP e ICMP hacia afuera pero solo permite ssh desde las máquinas de la red interna y física de los nodos.

Para crear un puerto que usaremos con la máquina Ubuntu_base de la subred_interna, en el menú de *Networks -> Red interna-> Ports -> Create Port*. Como

resultado, un error, después de investigar no he encontrado nada por lo que es una buena oportunidad para enseñar como usar la CLI y probar desde ahí.

[28]Para usar la CLI es necesario un archivo en el que se está la configuración para poder usar las APIs, en la parte izquierda de Horizon está la pestaña API Access, si accedemos veremos esta ventana

Displaying 7 items

Service	Service Endpoint
Compute	https://192.168.1.10:8774/v2.1
Identity	https://192.168.1.10:5000/v3/
Image	https://192.168.1.10:9292
Network	https://192.168.1.10:9696
Placement	https://192.168.1.10:8778
Volumev2	https://192.168.1.10:8776/v2/99fee0511a254b589a6267242562c04d
Volumev3	https://192.168.1.10:8776/v3/99fee0511a254b589a6267242562c04d

Displaying 7 items

Figura 3.23.- Endpoints servicios MicroStack

Esto de aquí son los endpoints de las diferentes APIs, y justo encima podremos ver un botón *Download OpenStack RC File* que ofrece el archivo en formato yaml (usado en aplicaciones de administración de servidores como Ansible) o sh, que será el que nosotros elijamos para el cliente de MicroStack. Una vez descargado podemos examinarlo para ver que se indican variables para saber el usuario, el proyecto, el ID del proyecto y demás información de ubicación.

Yo estoy usando Linux por lo que la instalación de la CLI mediante terminal, hay guías de un minuto para instalar en MacOS y Windows. Una vez instalado, actualizamos las variables de entorno de nuestro SO para incluir las de MicroStack. Probamos a

ejecutar un comando, pero nos da un error, ese error es a causa del certificado mencionado al iniciar el portal de Horizon, hay dos opciones, la más sencilla es incluir el *flag* --insecure para ejecutar comandos. La segunda opción y yo creo que más adecuada es la de incluir el certificado en nuestro SO, podemos descargarlo del nodo e incluirlo en nuestro sistema, cada uno es de una forma y yo enseño como seria para un entorno Linux basado en Debian.

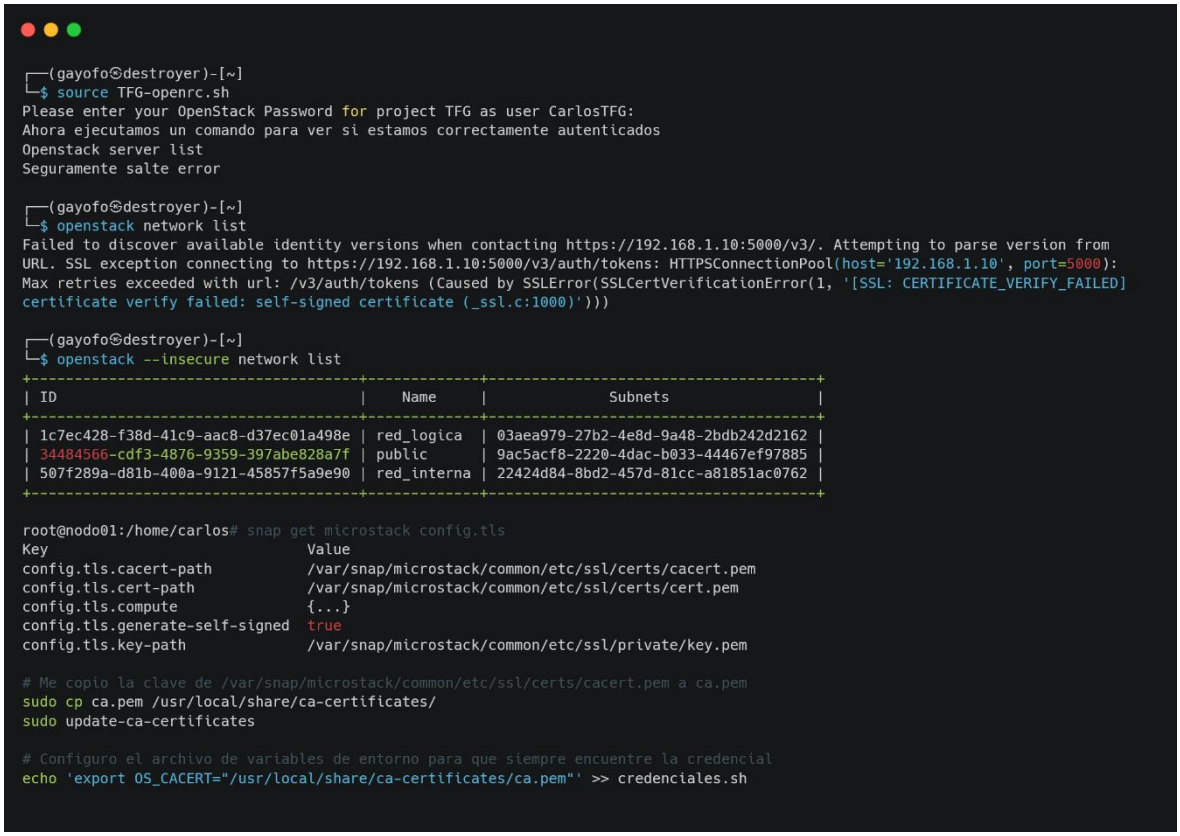

Figura 3.24.- Configurar la CLI

Ahora desde Horizon podemos verificar la creación y añadir grupos de seguridad al puerto creado.

Name	Fixed IPs	MAC Address	Attached Device
(5f15e69b-6c0a)	• 10.0.1.1	fa:16:3e:fe:81:4e	network:router_interface
puerto_ubuntu_web	• 10.0.1.20	fa:16:3e:94:58:15	Detached
(a816c117-b199)	• 10.0.1.2	fa:16:3e:c4:c8:15	network:dhcp
puerto_ubuntu_base	• 10.0.1.10	fa:16:3e:40:ca:38	Detached

Figura 3.25.- Interfaces de la subred interna

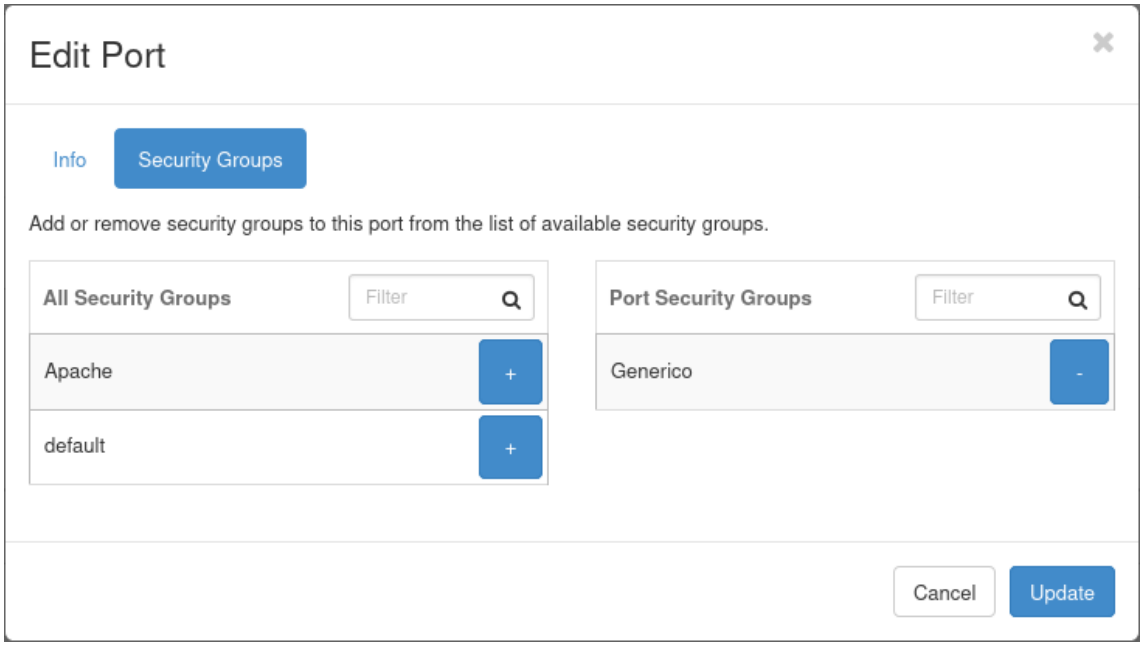

Figura 3.26.- Formulario para añadir grupos de seguridad a puertos

Bien, pues ya está todo, ahora crearemos la máquina Ubuntu_web como ya hemos visto en la introducción de este apartado y la máquina Ubuntu_base a través de la CLI. En la creación desde Horizon y desde CLI, es posible pasarle un archivo para que ejecute en la creación de la máquina, es conocido como *Cloud-init* y es muy útil en despliegues en grandes compañías porque permite que en el momento de creación de la máquina esta esté configurada. En Horizon a la hora de crear la instancia debemos ir a la parte de *Configuration* y ahí subir el archivo o escribirlo a mano, desde la CLI se pasa como objeto a uno de los flag.

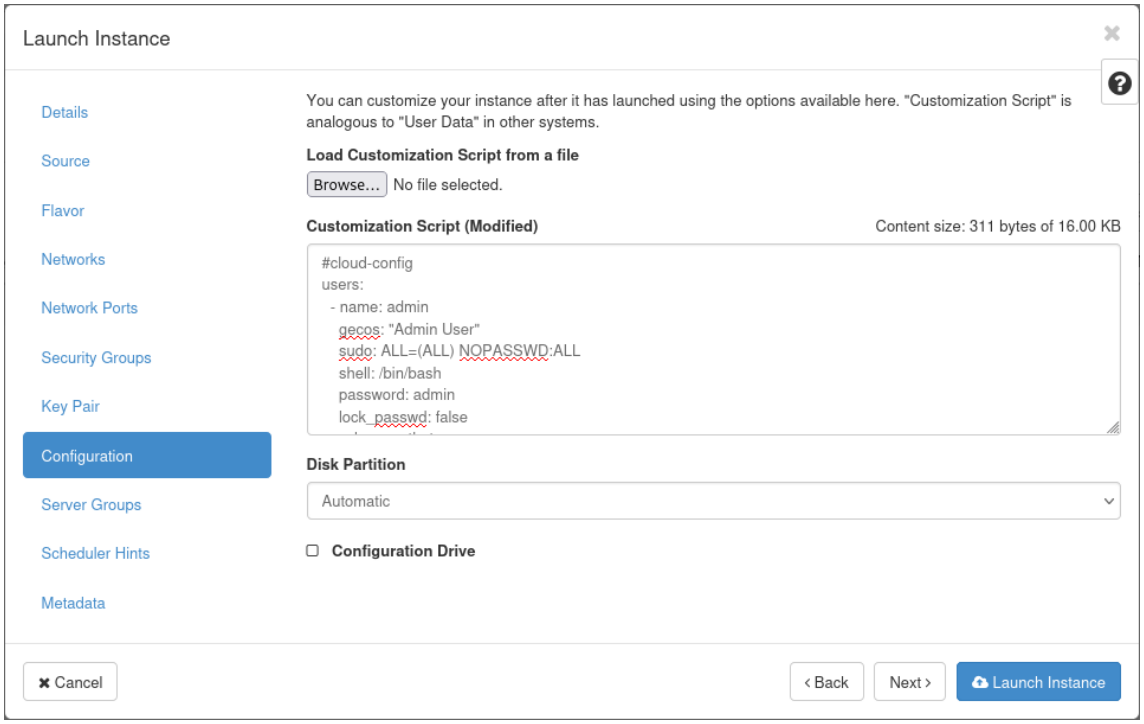

Figura 3.27.- Introducir código Cloud-init para configurar instancia en despliegue

Al usar la CLI para crear la máquina es posible indicar todas las configuraciones que son posibles desde la web Horizon.

```
openstack server create \
  --image "Ubuntu 22.04" \  # Imagen a usar (ver "openstack image list")
  --flavor m1.small \       # Tipo de máquina (ver "openstack flavor list")
  --network red_privada \   # Red a la que conectarla (si no usas un puerto específico)
  --port puerto_creado \    # Usar un puerto específico (ver "openstack port list")
  --security-group default \ # Grupo de seguridad (ver "openstack security group list")
  --key-name llave_ssh \    # Clave SSH para acceso (ver "openstack keypair list")
  --user-data cloud-init.yaml \  # Archivo de Cloud-Init
  --availability-zone nova \  # Zona de disponibilidad (ver "openstack availability zone list")
  --volume boot-volumen \     # Boot desde un volumen (ver "openstack volume list")
  --block-device-mapping vdb=extra-volumen \  # Adjuntar otro volumen extra
  --nic net-id=ID_RED,v4-fixed-ip=192.168.1.100 \  # Red + IP fija
  --config-drive true \  # Activa config-drive (útil para cloud-init)
  --property custom-tag=value \  # Añadir metadatos personalizados
  --wait \  # Espera hasta que la VM se cree completamente
  <NOMBRE_MAQUINA>
```

Figura 3.28.- Crear instancia desde CLI

```
openstack server create --image "Ubuntu_limpia" --flavor m1.medium --port puerto_ubuntu_base --network red_interna Ubuntu_base
```

Figura 3.29.- Ejemplo creación instancia desde CLI

Vamos a asociar dos IPs flotantes en la opción *Associate Floating IP* a las máquinas para que puedan ser accesibles desde la red física y si fuese necesario desde internet mediante NAT en el router físico. No es necesario tener IPs flotantes para que las máquinas salgan a internet. (Las IPs flotantes las podemos crear directamente desde el menú de asociación)

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status
cirros2	cirros	10.0.1.98	m1.small	cirros	Active
Ubuntu_base	Ubuntu_limpia	10.0.1.10, 192.168.1.153	m1.medium	-	Active
Ubuntu_web	Ubuntu_limpia	10.0.1.20, 192.168.1.156	m1.small	-	Active

Figura 3.30.- Instancias con sus IP flotantes

```
gayofo@pihole:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1000 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo valid_lft forever preferred_lft forever
inet6 ::1/128 scope host valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1442 qdisc fq_codel state UP
group default qlen 1000 link/ether fa:16:3e:40:ca:38 brd ff:ff:ff:ff:ff:ff altname
enp0s3 inet 10.0.1.10/24 metric 100 brd 10.0.1.255 scope global dynamic ens3
valid_lft 42947sec preferred_lft 42947sec inet6 fe80::f816:3eff:fe40:ca38/64
scope link valid_lft forever preferred_lft forever

gayofo@pihole:~$ ping 10.0.1.10
PING 10.0.1.10 (10.0.1.10) 56(84) bytes of data.
64 bytes from 10.0.1.10: icmp_seq=1 ttl=64 time=4.22 ms
64 bytes from 10.0.1.10: icmp_seq=2 ttl=64 time=2.18 ms
^C
--- 10.0.1.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 2.181/3.199/4.218/1.018 ms

gayofo@pihole:~$ ping google.es
PING google.es (142.250.178.163) 56(84) bytes of data.
64 bytes from mad41s08-in-f3.1e100.net (142.250.178.163): icmp_seq=1 ttl=113 time=13.3 ms
64 bytes from mad41s08-in-f3.1e100.net (142.250.178.163): icmp_seq=2 ttl=113 time=10.5 ms
^C
--- google.es ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 10.525/11.922/13.320/1.397 ms
```

Figura 3.31.- Comprobando conectividad

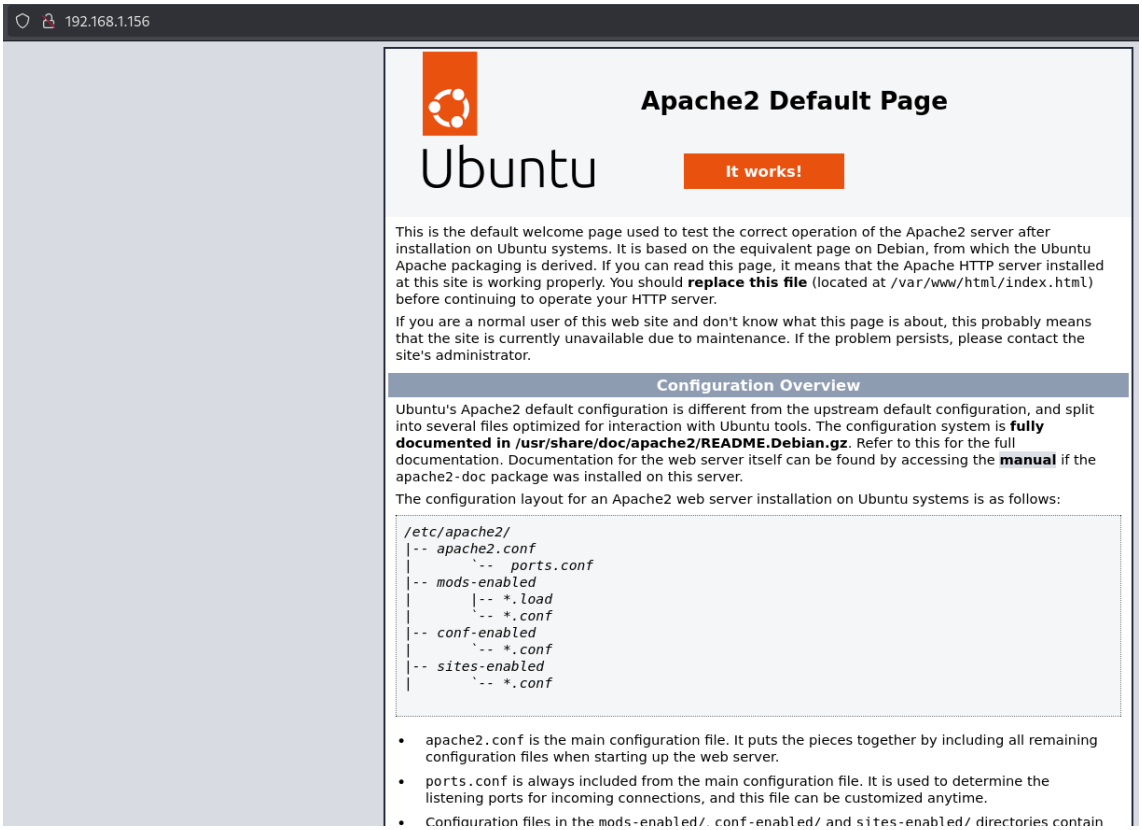

Figura 3.32.- Comprobando NAT en MicroStack

También es posible asociar una interfaz de la red externa y que reciba una IP directamente.

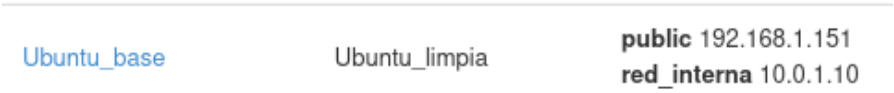

Figura 3.33.- Interfaz interna y externa en Horizon

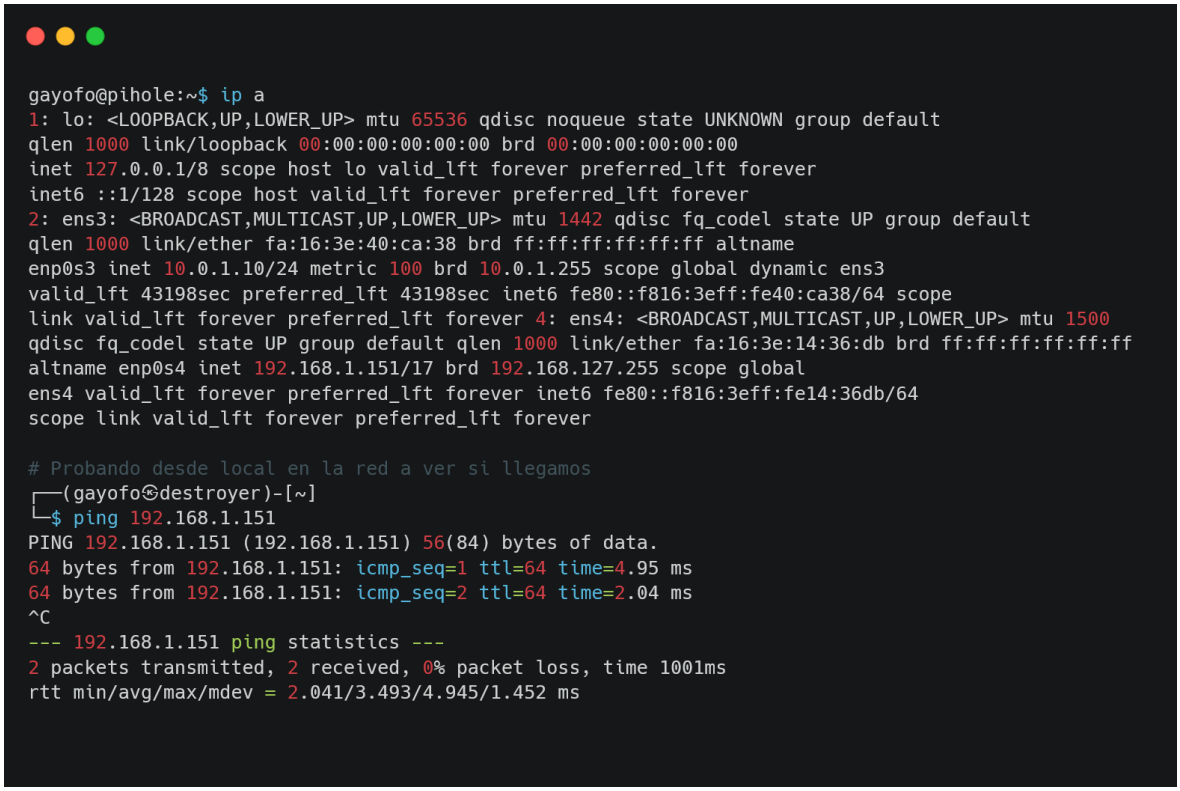

Figura 3.34.- Probando conectividad con interfaz directa a red externa

4. Pruebas de seguridad

La seguridad en el cloud computing es un aspecto clave, comprometer un sistema puede llevar, en el peor de los casos a la caída de todo un sistema afectando a numerosos clientes o comprometiendo datos. En este apartado se detallan las pruebas realizadas para evaluar la seguridad de Microstack.

Las pruebas se centran en diversas áreas clave de seguridad: explotación de vulnerabilidades conocidas (CVE), pruebas de acceso no autorizado, ataques de denegación de servicio y técnicas de envenenamiento de red. Se ha seguido una metodología basada en pruebas de concepto (PoC), reproduciendo escenarios reales de ataque para validar la efectividad de las medidas de protección implementadas en MicroStack.

Las vulnerabilidades analizadas en esta sección han sido obtenidas a partir de bases de datos públicas de CVE (Common Vulnerabilities and Exposures) y están documentadas en plataformas oficiales de seguridad como el NVD (National Vulnerability Database) y la base de datos de OpenStack. Se ha prestado especial atención a aquellas que afectan directamente a Nova y Neutron, dos de los componentes principales de OpenStack, con el objetivo de verificar si también están presentes en MicroStack y cómo se comportan en un entorno restringido por Snap. [49][50][51]

4.1.- CVE-2024-40767

La vulnerabilidad CVE-2024-40767 afecta el proceso de carga de imágenes en OpenStack Nova. El problema surge cuando un atacante suministra una imagen en formato RAW que en realidad contiene una estructura QCOW2 oculta, con un archivo de respaldo que hace referencia a la ruta del sistema host. De igual forma, una imagen en

formato VMDK puede incluir un descriptor que apunta a archivos aleatorios dentro del servidor que corre Nova. [30]

]QCOW2 tiene una característica llamada backing file, que permite que un disco virtual derive su contenido de otro archivo. En teoría, esto sirve para ahorrar espacio al compartir datos comunes entre múltiples discos. En un entorno mal configurado, esta técnica permite a un atacante autenticado leer archivos del sistema anfitrión, como /etc/passwd, claves privadas u otros archivos sensibles, comprometiendo la seguridad del sistema. [42][43

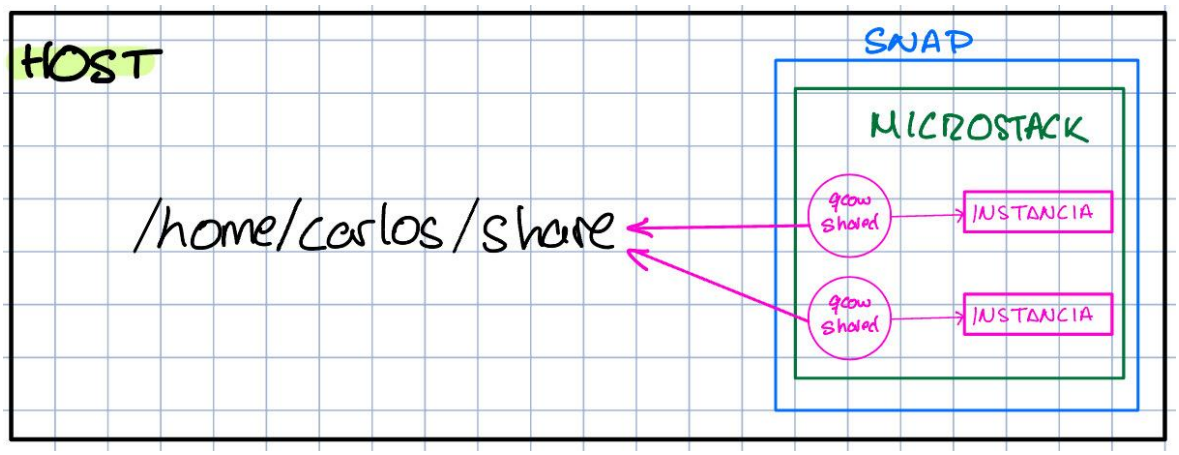

Figura 4.1.- Funcionamiento original del sistema de montajes de imágenes qcow2

Si se sube una imagen como QCOW2, Nova puede realizar validaciones específicas del formato, como comprobar si hay un backing file fuera de la ruta esperada. Pero si se sube como RAW, Nova no examina si realmente es un archivo plano y simplemente la almacena y la monta, cuando qemu la va a ejecutar detecta en las cabeceras qcow2 y la arranca como tal, sin comprobar el back file en este caso. Los pasos son:

1. Se tomó una imagen base en formato qcow2 y se le añade en back file de /etc/passwd.
2. Se transforma a RAW y se sube así.
3. Nova recibe una imagen en formato RAW para añadir como disco a una instancia para esta tener acceso a datos compartidos de manera local.

4. Qemu recibe la imagen a montar pero no bloquea referencias a archivos fuera del almacenamiento de imágenes, por lo que trata de cargar /etc/passwd como si fuera parte del disco de la instancia.
5. Resultado: /etc/passwd queda accesible dentro de la instancia, lo que permite la fuga de información del host.

[44]Esto es la teoría, en la práctica me he encontrado con que no funciona, tras varios intentos y diferentes formas de transformar imágenes no montaba correctamente el directorio en el punto de montaje. Dado que estaba intentando extrapolar el problema de OpenStack a MicroStack creo que el problema es Snap.

Snap maneja las aplicaciones en un espacio aislado conocido como sandbox basado en AppArmor, un sistema de control para Linux que extiende los permisos tradicionales al crear perfiles de aplicaciones que restringen que archivos, rutas, dispositivos y demás componentes del sistema operativo puede acceder la aplicación.

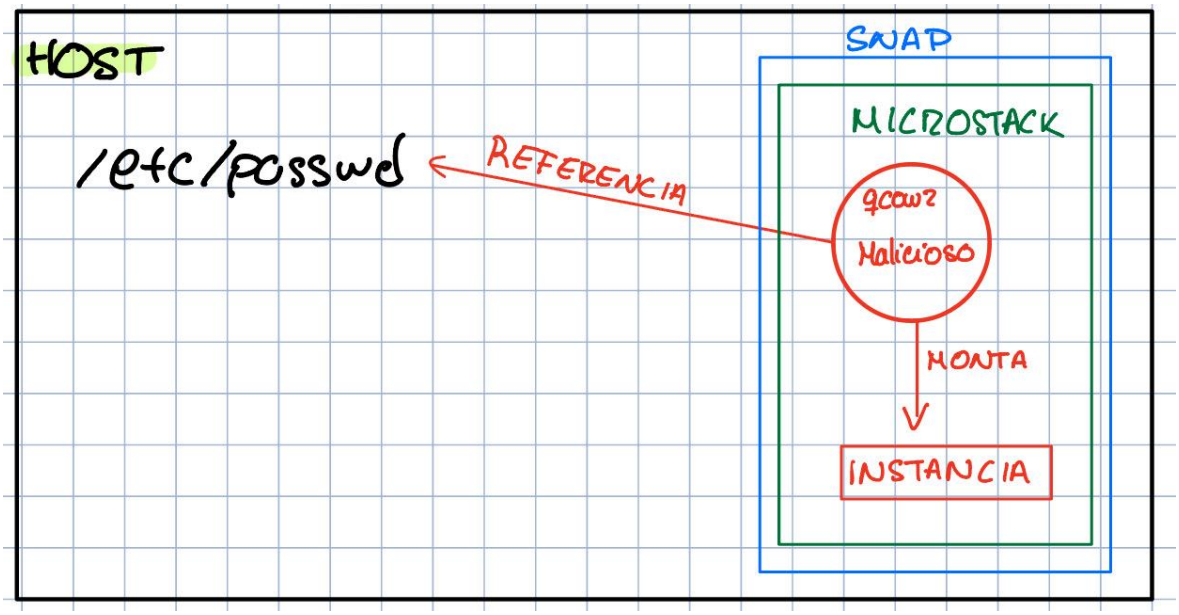

Figura 4.2.- Modificación del esquema inicial

Una vez lanzada la imagen nos dirigimos al punto de montaje donde debería estar el sistema de archivos del host, pero no encontramos nada. Dado que la versión nova que ejecuto es anterior a las publicadas, lo que debería suponer vulnerabilidad. El hecho de que MicroStack se instale a través de Snap hace que se encuentre en un sandbox el cual no tiene acceso a datos del sistema host.

4.2.- CVE-2024-53916

La seguridad en OpenStack se basa en un modelo de permisos basado en proyectos e inquilinos, asegurando el aislamiento entre ellos. Sin embargo, en Neutron, el motor de políticas de etiquetado de servicios verifica de manera insuficiente el ID del proyecto del recurso padre o del recurso superior en versiones anteriores a 25.0.1. Esto introduce una vulnerabilidad que permite a un atacante eludir las restricciones de acceso y modificar etiquetas en recursos de red sin tener permisos para ello. [32]

Para esta prueba de concepto he creado un usuario CVE con permisos solo de lectura en mi proyecto TFG, además de una nueva red para evitar estropear el escenario creado en apartados anteriores, su nombre es CVE_1.

El objetivo es ver si somos capaces de realizar una acción que a priori no tenemos permisos, conocer las medidas de mitigación e investigar si es posible corregir la vulnerabilidad.

Para el desarrollo he descargado el archivo con las variables de entorno del usuario para usar desde la CLI y las he importado. Luego he listado las redes con intención de obtener el id de la red objetivo y he ejecutado el comando para cambiar el tag.

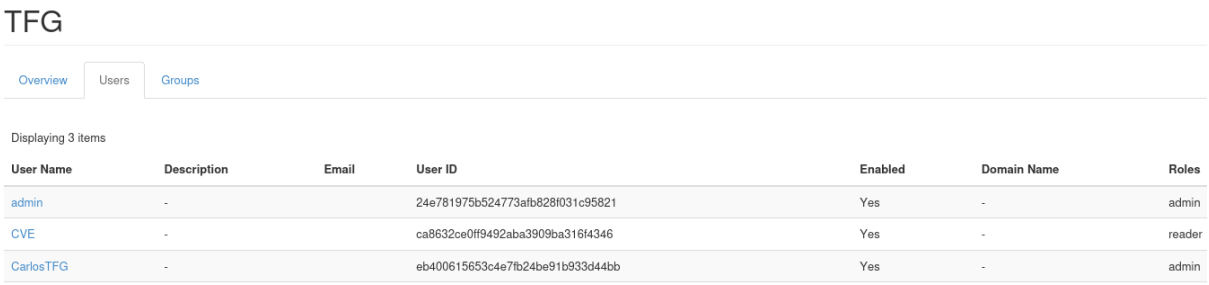

Figura 4.3.- Usuarios MicroStack y sus ID

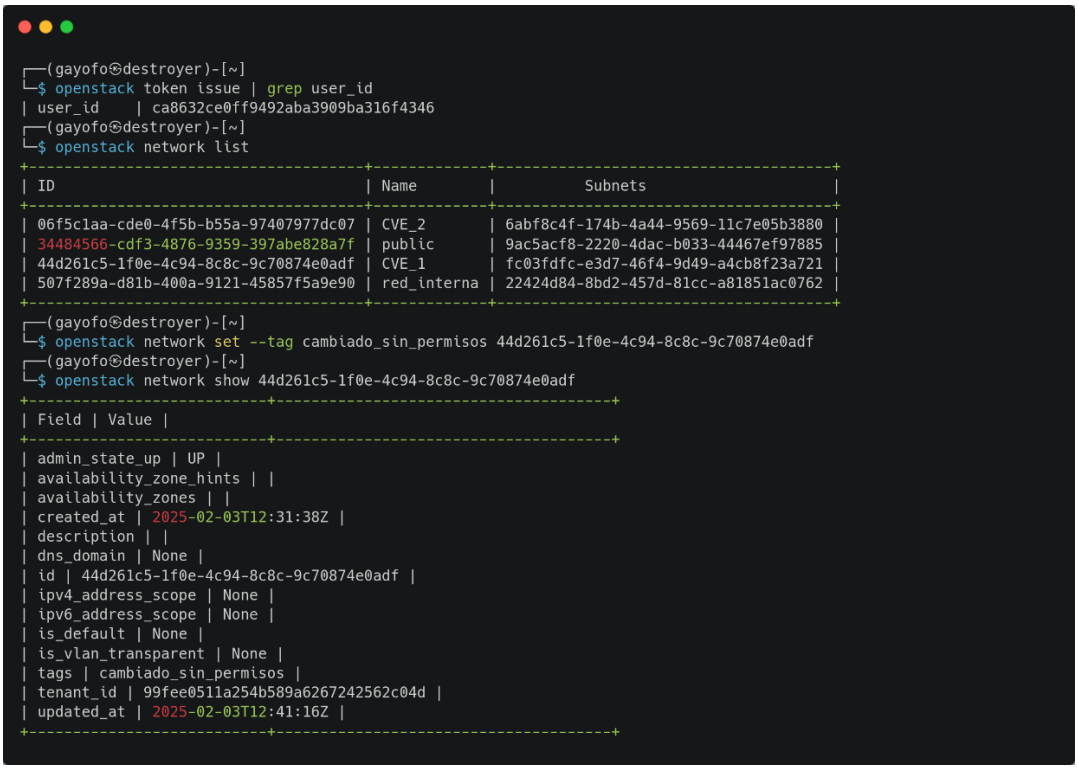

Figura 4.4.- Trabajando desde CLI con usuario CVE reader

La política de roles está fallando, esto me lleva a pensar que puede fallar para más casos. Voy a intentar crear una red en el proyecto con el mismo usuario reader.

Nuevamente hay un error en la comprobación de roles y es permitido que cree una red sin ningún tipo de restricción. De la misma forma pruebo a borrarla y me lo permite.

Figura 4.5.- Creando redes con usuario CVE reader

Quiero entender que pasa, para ir más rápido quiero trabajar desde Horizon, me registro y verifico que sí, puedo crear redes, instancias e imágenes.

He querido seguir indagando y me hago la pregunta ¿si un usuario reader puede hacer esto, que puede hacer un usuario admin? Para ello cargo las variables de entorno de CarlosTFG y listo las redes, para mi sorpresa, aparecen redes de otros proyectos, más en concreto el proyecto admin. Para asegurarme, creo otro proyecto (CVE) con otro usuario con rol admin (David) y añado una red (CVE), todo esto desde el usuario CarlosTFG. Estoy modificando la infraestructura cloud desde el usuario CarlosTFG, admin sí, pero en su proyecto.

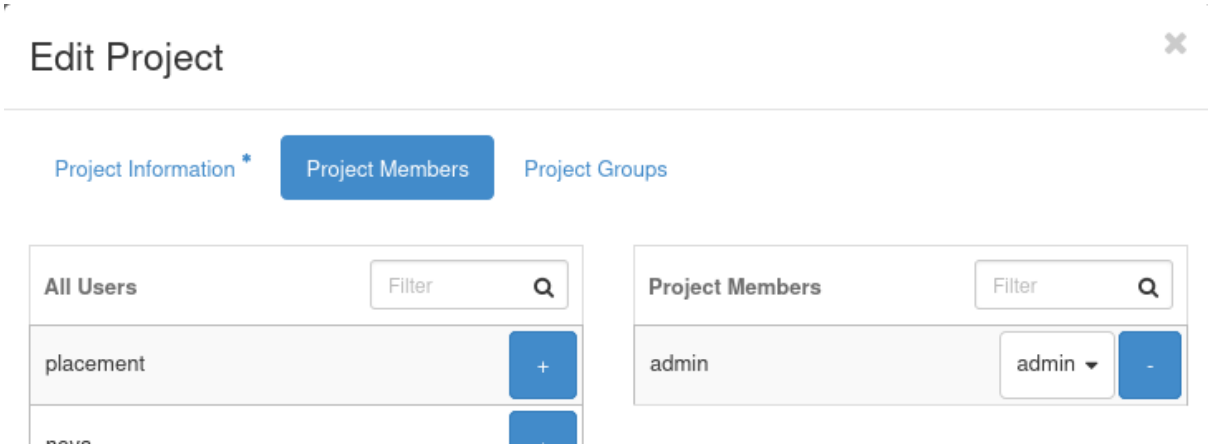

Figura 4.6.- Captura miembros y permisos proyecto admin

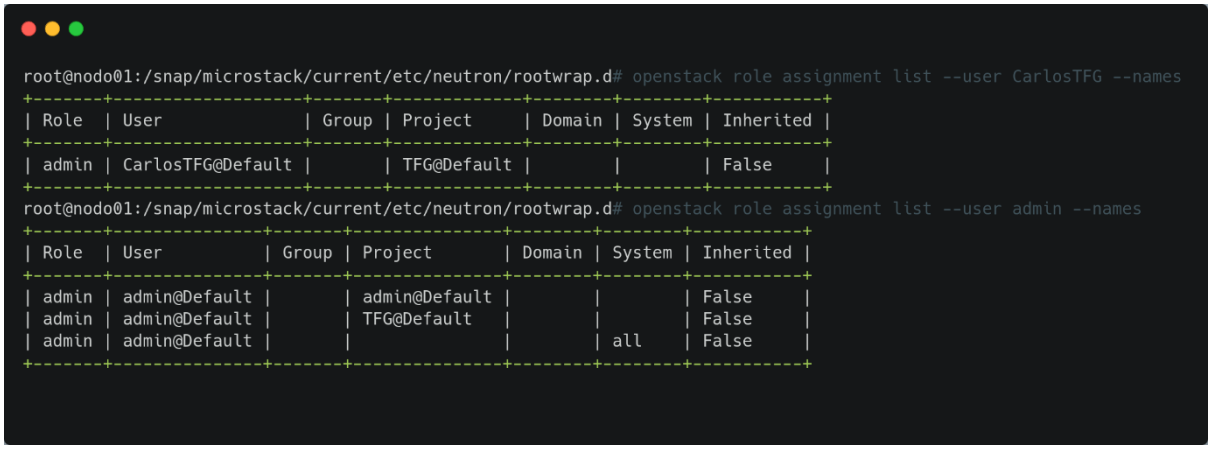

Figura 4.7.- Comprobando roles en proyectos de CarlosTFG y admin

Después de hacer la prueba, queda claro que la política de roles en MicroStack está rota y permite hacer cosas que no deberían ser posibles. Lo primero que comprobé fue la vulnerabilidad en Neutron, donde el sistema no valida bien el ID del proyecto al etiquetar recursos de red. Esto ya es un problema grave porque deja a cualquier usuario modificar etiquetas sin permiso.

Pero el problema va más allá. Creé un usuario con solo permisos de lectura y, aun así, pude crear y borrar redes sin ninguna restricción. Para asegurarme, entré en Horizon y vi que también podía lanzar instancias y subir imágenes, lo que no tiene sentido si el rol está bien configurado. Esto ya indica que el control de permisos en MicroStack no funciona como debería.

Lo peor vino cuando probé con un usuario admin dentro de su proyecto y me di cuenta de que podía ver y modificar redes de otros proyectos. Es decir, el aislamiento entre proyectos no existe, cualquier admin puede tocar recursos que no le corresponden, lo cual es un fallo de seguridad muy serio.

El impacto de esto es enorme. Un usuario con solo permisos de lectura podría subir una imagen con una puerta trasera o un ransomware oculto, y cualquier instancia creada con esa imagen quedaría comprometida. Buscando soluciones, vi que en OpenStack se puede configurar *policy.yaml*, pero en MicroStack no hay nada parecido. Además, intenté modificar permisos por CLI y tampoco me dejó, lo que me lleva a pensar que MicroStack no está pensado para producción y por eso ni siquiera permite ajustar bien los roles. En definitiva, no es un entorno seguro si se necesita control real sobre los permisos.

4.3.- Denegación de servicio (DoS) Horizon

Como cualquier aplicación web basada en Django y HTTP, Horizon es susceptible a ataques de denegación de servicio si no está bien protegida.

El objetivo es saturar Horizon con peticiones HTTP hasta que deje de responder y evaluar como MicroStack gestiona la carga. Por otro lado, averiguar si queda registro del ataque y también afecta a otros servicios de MicroStack de forma colateral. [34]

La prueba la realizo con slowloris, una herramienta específica de ataques de denegación de servicio (DDoS) confeccionada para colapsar servidores web manteniendo varias conexiones abiertas y enviando las cabeceras HTTP de manera fragmentada y lenta. El flujo empieza con la apertura de una conexión HTTP y el envío de una petición incompleta haciendo que el servidor se quede esperando por el total, poco a poco envía

encabezados parciales, haciendo que el servidor no cierre conexión ya que cada poco recibe un cacho más de la petición. [33]

```
# Peticion HTTP que fragmenta linea a linea
GET /index.html HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0
Accept-Encoding: gzip, deflate
X-Forwarded-For: 192.168.1.100
            .
            .
            .
Connection: close
```

Figura 4.8.- Cabecera petición HTTP

El comando para ejecutar es el siguiente

```
slowloris -s 10000 -p 443 --https --sleeptime 1 192.168.1.10
[04-02-2025 20:18:45] Sending keep-alive headers...
[04-02-2025 20:18:45] Socket count: 1021
[04-02-2025 20:18:45] Creating 9305 new sockets...
```

Figura 4.9.- Ataque son slowloris

- -s 500: Mantiene 500 conexiones abiertas al mismo tiempo.
- -p 443: Ataca el puerto 443 (HTTPS de Horizon).
- --https: Usa HTTPS en lugar de HTTP.
- -x 5: Intervalo de 5 segundos entre cada encabezado enviado (para evitar timeout).

- -r 100: Envia 100 encabezados por conexión antes de cerrarla.
- -o 10.0.0.5: IP del servidor que corre Horizon.

Probé primero con 500 y 1000 conexiones simultáneas, sin notar cambios. Luego subí a 10.000 y ahí sí que Horizon dejó de cargar un momento, pero no se cayó del todo, así que no lo consideraría un DoS real, sino más bien una sobrecarga puntual.

Investigando un poco más, vi que Nginx, que hace de proxy inverso para MicroStack, limita las conexiones abiertas. Eso significa que por muchas conexiones lentas que intente abrir Slowloris, nunca va a conseguir bloquear completamente el servicio porque Nginx simplemente no le deja jugar con más conexiones de las permitidas. [35]

En resumen, Horizon aguantó el ataque, pero no porque sea inmune, sino porque Nginx ya impone una restricción que hace que Slowloris no tenga tanto efecto. Aun así, si el ataque fuese diferente (por ejemplo, algo más agresivo a nivel de aplicación o distribuido como un DDoS), habría que ajustar mejor la configuración para reforzar la seguridad y evitar que el sistema se sature.

Para mejorar la protección contra ataques más agresivos, habría que revisar y ajustar las configuraciones de Nginx y el sistema en general. Una buena estrategia sería endurecer los límites de conexiones y tiempos de espera en Nginx, de forma que cualquier intento de mantener conexiones abiertas durante demasiado tiempo se cierre antes de que cause impacto. También se podría habilitar rate limiting para controlar cuántas peticiones puede hacer un cliente en un periodo de tiempo determinado.

Aparte de Nginx, valdría la pena integrar un WAF (Web Application Firewall) que detecte patrones de ataque y bloquee IPs sospechosas en tiempo real. Si el objetivo es defenderse de DDoS más serios, se podría añadir un proxy de protección externo, como Cloudflare o un servicio similar, para filtrar tráfico antes de que llegue a Horizon.

Por último, sería interesante revisar **los logs y métricas del sistema**, asegurándose de que todo intento de ataque queda registrado para poder afinar futuras medidas de mitigación y tener una alerta temprana si el tráfico empieza a subir de forma anómala.

```
root@nodo01:/home/carlos# cat /var/snap/microstack/common/log/nginx-error.log | tail -n 20
2025/02/04 20:26:58 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:26:59 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:00 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:02 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:03 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:04 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:05 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:06 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:07 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:09 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:10 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:10 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:10 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:10 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:10 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:10 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:10 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:10 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:10 [alert] 38424#0: 768 worker_connections are not enough
2025/02/04 20:27:24 [alert] 38424#0: 768 worker_connections are not enough
```

Figura 4.10.- Logs de Nginx

4.4.- ARP Spoofing

El ataque ARP Spoofing es una técnica de envenenamiento de caché ARP que permite a un atacante interceptar, modificar o redirigir tráfico en una red local. En este ejercicio, se evaluará la seguridad del router de **MicroStack (Neutron)** frente a este tipo de ataque, identificando vulnerabilidades y proponiendo soluciones para su mitigación. [36]

Como objetivos están la evaluación del router MicroStack frente ataques de este tipo, determinar si las instancias pueden ser víctimas de MITM e identificar medidas de protección.

Para el desarrollo se dispone de una instancia llamada victima con IP 192.168.222.137 y MAC fa:16:3e:ca:96:d9, otra maquina atacante con IP 192.168.222.168 y MAC fa:16:3e:2d:b6:f9 y el router con la IP 10.20.20.1 y MAC fa:16:3e:7f:0a:34. Desde la máquina atacante vamos a decir a la máquina víctima que la IP de su gateway corresponde con la MAC del atacante, de esa forma cuando intente comunicarse con la gateway para salir a internet podremos interceptar el paquete.

Figura 4.11.- Tabla ARP de la víctima

Comenzamos localizando la máquina victima en la red, aparecen tres equipos, sabemos que la primera es el gateway, pero la segunda tenemos que tenerdotes de investigación y tras un vistazo rápido en la red descubrimos que es el servidor DHCP, por lo tanto, sabemos que la 139 es la víctima.

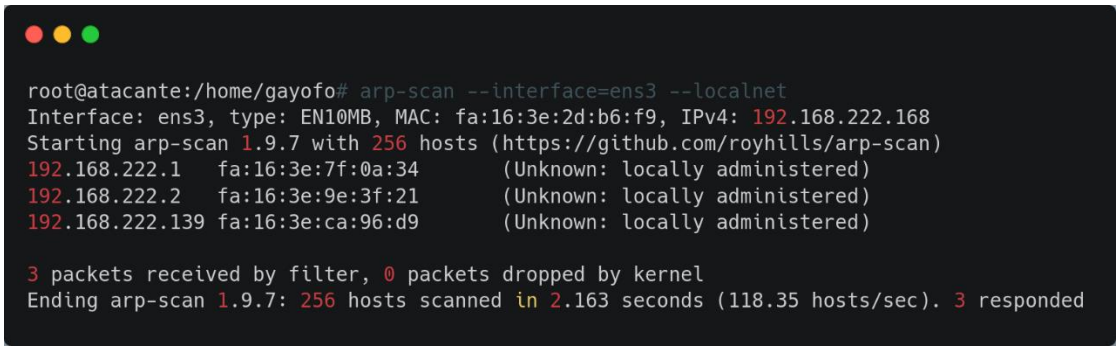

Figura 4.12.-Descubriendo hosts en la red desde el equipo atacante

Ahora iniciamos el envenenamiento de la tabla con la herramienta dsniff [37]

Figura 4.13.- Lanzando ataque de arp spoofing

- -i <INTERFAZ> → Define la interfaz de red a usar (ejemplo: eth0 o ens33).
- -t <VICTIMA_IP> → Indica la IP de la víctima, a quien se quiere engañar para que envíe tráfico al atacante.
- -r <GATEWAY_IP> → Indica la IP del router/gateway, al que también se quiere engañar.

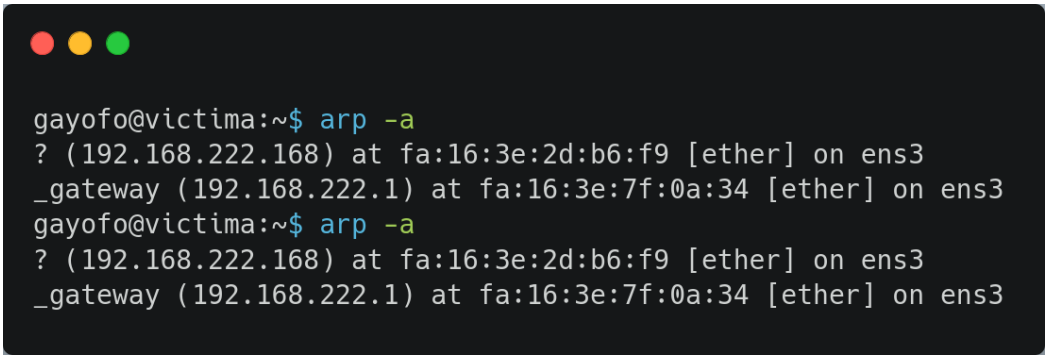

Figura 4.14.- Tabla ARP de la víctima después del ataque

Como resultado vemos que el ataque no tiene efecto. Puede ser debido a, el port security está activo en la interfaz por defecto, es controlado por OVS que es consciente de la IP/MAC del gateway e impide tráfico que lo suplante. Se puede desactivar, pero el puerto no puede tener un grupo de seguridad activo, es decir, está bien protegida. [38]

Allowed address pairs es una opción en el puerto que permite que una misma interfaz tenga varias IP o MAC, es decir, si añadimos que mi IP puede tener la MAC de router funcionaría.

Como en el anterior apartado hemos visto como con un usuario reader podemos manipular instancias, vamos a modificar el port security eliminándolo, es requisito

desactivar los grupos de seguridad asociados al puerto y sin darme cuenta, acabo de dejar a la maquina sin conexión ya que el flujo de red es a través de whitelist. Por tanto, este método hasta donde he alcanzado no es posible de realizar.

Intentando allowed address pairs pasa algo similar ya que para poder activarlos necesitamos tener un grupo de seguridad y esto directamente activa el port security.

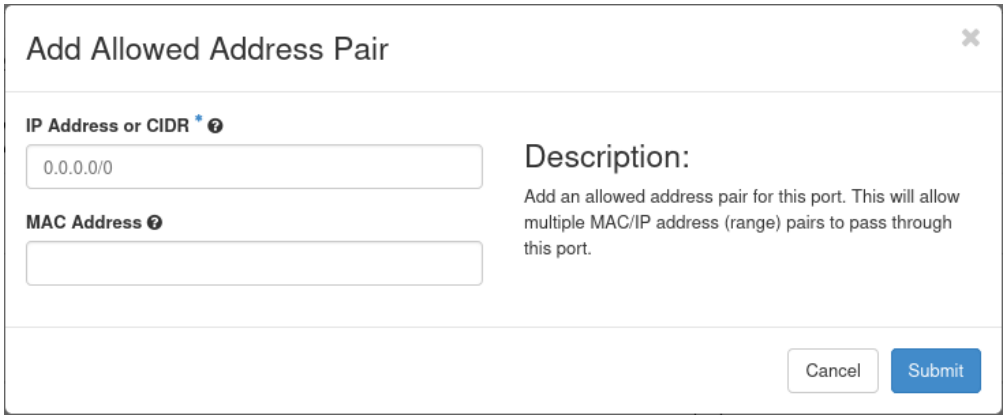

Figura 4.15.- Formulario añadir *pairs address*

Los resultados del ataque ARP Spoofing en el router de MicroStack (Neutron) demuestran que, por defecto, la infraestructura está bien protegida frente a este tipo de ataques. El port security activado en las interfaces impide el tráfico malicioso que suplanta la IP/MAC del gateway, gracias a la gestión de Open vSwitch (OVS), que controla la autenticidad de los paquetes. Además, la funcionalidad de Allowed Address Pairs, que permitiría asociar varias direcciones IP/MAC a una interfaz, no es viable sin activar simultáneamente un grupo de seguridad, lo que reactiva el port security.

Se ha comprobado que la eliminación del port security requiere desactivar previamente los grupos de seguridad, lo que rompe la conectividad de la instancia al aplicar una política de whitelist en el flujo de red. Por lo tanto, la manipulación de estas configuraciones en un entorno estándar de MicroStack resulta inviable sin afectar la operatividad del sistema.

En conclusión, MicroStack presenta medidas de seguridad efectivas contra ataques ARP Spoofing, dificultando la realización de un ataque MITM sin modificaciones profundas en la configuración. Sin embargo, en entornos donde se requiera mayor flexibilidad en la red, sería importante evaluar estrategias adicionales para mantener la seguridad sin afectar la funcionalidad.

5. Recomendaciones de seguridad

En este apartado he querido aclarar una directrices básicas y consejos de seguridad para administradores que creo convenientes en un servicio que va a estar expuesto al mundo a través de internet.

Uno de los principios clave es el de menor privilegio, se basa en que cada usuario tenga acceso únicamente a los recursos y permisos estrictamente necesarios. Esto reduce el impacto de un problema de seguridad. Como complemento clave, la autenticación multifactor se convierte en un mecanismo esencial para proteger el acceso a cuentas privilegiadas, reduciendo el riesgo de accesos no autorizados.

El cifrado de datos en tránsito (mientras circulan por red) y en reposo es una medida necesaria para preservar la confidencialidad de la información. Es recomendable utilizar TLS/SSL[56] (su aplicación más conocida es HTTPS) para las comunicaciones y herramientas como LUKS[57] para cifrar discos de almacenamiento. Además, mantener MicroStack actualizado y aplicar parches de seguridad de manera periódica es una estrategia básica para evitar vulnerabilidades explotables.

Para mejorar la seguridad en la autenticación, debemos configurar Keystone correctamente, deshabilitando la creación de usuarios anónimos y declarando políticas de rotación de contraseñas. La validación de tokens con tiempos de expiración cortos incrementa la seguridad en las sesiones activas, pero sin ser demasiado cortos para evitar interferencias en el normal desarrollo de una tarea. Integrar Keystone con sistemas de autenticación externos como LDAP o OpenID facilita la administración centralizada de identidades y fortalece el control de acceso.

Las amenazas a la seguridad de red también son un punto delicado y sensible. La implementación de medidas anti-DDoS en el router y firewall previene ataques de denegación de servicio. La inspección ARP ayuda a evitar ataques de suplantación de

direcciones. Utilizar VLANs[58] o redes privadas para aislar servicios críticos es una estrategia eficaz para minimizar la exposición a amenazas externas. Adicionalmente, la monitorización activa del tráfico con herramientas como Suricata[59] o Zeek[60] permite detectar actividades sospechosas en la red.

Implementar la autenticación mediante claves SSH[61] y deshabilitar el acceso por contraseña refuerza la protección de los servidores. Aplicar reglas de firewall con iptables o nftables permite granular las conexiones entrantes y salientes. Eliminar servicios innecesarios en las instancias y configurar AppArmor o SELinux proporciona una capa adicional de aislamiento para los procesos

Para garantizar la seguridad a largo plazo, la monitorización y auditoría continua de la infraestructura son necesarias. Implementar herramientas como Fail2Ban[62] ayuda a mitigar ataques de fuerza bruta en servicios como SSH. Utilizar un SIEM como Wazuh[63] o ELK[64] permite correlacionar eventos de seguridad y detectar análisis de patrones de ataque en tiempo real. La auditoría periódica de los logs de Keystone, Horizon y Nova contribuye a identificar anomalías y reforzar la postura de seguridad de la infraestructura. Finalmente, el uso de soluciones de monitorización como Zabbix[65], Prometheus[66] o Grafana[67] facilita la recolección y visualización de datos clave para la administración y respuesta ante incidentes de seguridad.

6. Planificación temporal

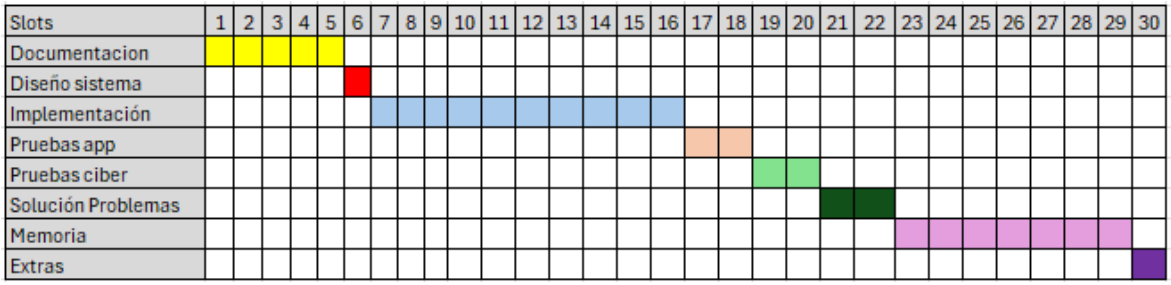

Figura 8.1.- Diagrama de Gantt

Etapa	Duración
Estudio documentación	50 horas
Diseño sistema	10 horas
Implementación	100 horas
Pruebas	60 horas
Memoria	40 horas
Solución problemas	40 horas
Total	300 horas

Tabla 8.2.- Tabla de planificación de horas

Quiero aclarar que las horas de solución de problemas y pruebas se han comido parte de las inicialmente planificadas para memoria y extras, en extras se pretendía llevar a cabo un despliegue más amplio de instancias y servicios para realizar pruebas de rendimiento, la cantidad de problemas a los que me he enfrentado han supuesto un total desvío de los objetivos iniciales y del alcance del trabajo. En otras circunstancias con un plazo más atrasado, podría haber descartado esas horas y reinvertirlas nuevamente desde un punto de partida más adelantado.

7. Conclusiones

MicroStack se presenta como una solución práctica para la implementación rápida de entornos cloud, proporcionando una plataforma simplificada basada en OpenStack. Su facilidad de despliegue y administración lo convierte en una herramienta ideal para entornos de prueba y aprendizaje.

La falta de documentación ha hecho que la investigación fuese muy compleja y tuviese que utilizar guías y tutoriales de OpenStack que en muchos casos no eran transferibles a MicroStack debido a su naturalidad de contenedor Snap. Por lo que en lo personal ha sido un trabajo lleno de obstáculos y complicaciones, pero que me ha servido para aplicar la insistencia y constancia que esta carrera me ha enseñado.

Las pruebas de pentesting han permitido identificar vulnerabilidades relacionadas con la asignación de roles y la gestión de recursos entre proyectos. Se ha demostrado que los usuarios con permisos restringidos pueden realizar modificaciones que, en un entorno de producción, podrían comprometer el sistema. Sin embargo, también se ha comprobado que MicroStack incorpora mecanismos de protección efectivos frente a ciertos tipos de ataques, como ARP Spoofing y DoS, en gran parte debido a su arquitectura basada en Open vSwitch y Nginx.

En conclusión, si bien MicroStack ofrece una solución accesible y funcional para la virtualización de infraestructuras, después de estar trabajando en el laboratorio con mucho esfuerzo, documentándome y resolviendo problemas que en su mayoría se tenían que resolver reinstalando la plataforma, no la recomiendo en absoluto más que para tener un recurso para el aprendizaje en casa de OpenStack.

8. Bibliografía

[1] https://sixe.es/noticias/explorando-MicroStack-una-plataforma-de-nube-privada-y-eficiente?srsltid=AfmBOoruiBvr37SMJhDgx9gY8E0ReRxkrbCxhXJQd_ddYyHVDU-WBFAu Accedido por última vez 25/01/2025

[2] https://www.redhat.com/es/topics/cloud-computing/iaas-vs-paas-vs-saas Accedido por última vez 25/01/2025

[3] https://docs.OpenStack.org/python-OpenStackclient/latest/cli/man/OpenStack.html Accedido por última vez 27/01/2025

[4] https://docs.OpenStack.org/keystone/latest/ Accedido por última vez 27/01/2025

[5] https://docs.OpenStack.org/keystone/latest/getting-started/architecture.html Accedido por última vez el 27/01/2025

[6] https://virtualizadesdezero.com/roles-en-OpenStack/ Accedido por última vez el 27/01/2025

[7] https://docs.OpenStack.org/keystone/latest/admin/tokens-overview.html Accedido por última vez 30/01/2025

[8] https://docs.OpenStack.org/keystone/latest/admin/identity-concepts.html Accedido por última vez el 27/01/2025

[9] https://docs.OpenStack.org/keystone/latest/admin/cli-manage-projects-users-and-roles.html Accedido por última vez 28/01/2025

[10] https://docs.OpenStack.org/keystone/latest/ Accedido por última vez 03/02/2025

[11] https://docs.OpenStack.org/glance/latest/ Accedido por última vez 03/02/2025

[12] https://docs.OpenStack.org/glance/latest/user/formats.html#disk-format Accedido por última vez 03/02/2025

[13] https://docs.OpenStack.org/glance/rocky/admin/useful-image-properties.html Accedido por última vez 03/02/2025

[14]https://docs.redhat.com/en/documentation/red_hat_OpenStack_platform/16.2/html/creating_and_managing_images/ch-image-service#section-create-images Accedido por última vez 10/01/2025

[15]https://docs.OpenStack.org/nova/latest/admin/architecture.html Accedido por última vez el 01/02/2025

[16] https://www.maquinasvirtuales.eu/componentes-de-OpenStack-OpenStack-compute-nova/ Accedido por última vez el 01/02/2025

[17]https://docs.OpenStack.org/nova/queens/reference/rpc.html Accedido por última vez el 01/02/2025

[18] https://docs.OpenStack.org/nova/rocky/user/flavors.html Accedido por última vez el 25/01/2025

[19]https://blog.nashtechglobal.com/how-to-create-an-instance-using-OpenStack-cli-2/ Accedido por última vez el 30/01/2025

[20] https://docs.OpenStack.org/neutron/latest/admin/deploy-ovs.html Accedido por última vez el 01/02/2025

[21] https://docs.OpenStack.org/neutron/latest/admin/intro-basic-networking.html Accedido por última vez el 01/02/2025

[22] https://docs.OpenStack.org/neutron/latest/admin/intro-network-components.html Accedido por última vez el 01/02/2025

[23] https://docs.OpenStack.org/neutron/latest/admin/intro-nat.html Accedido por última vez el 01/02/2025

[24] https://docs.OpenStack.org/neutron/latest/admin/intro-os-networking.html Accedido por última vez el 01/02/2025

[25] https://iesgn.github.io/cloud/curso/u8/red-privada Accedido por última vez el 01/02/2025

[26] https://docs.OpenStack.org/horizon/latest/ Accedido por última vez 20/01/2025

[27] https://rnascimento.com/OpenStack-single-node-MicroStack/ Accedido por última vez 20/01/2025

[28] https://help.ovhcloud.com/csm/es-es-public-cloud-compute-prepare-OpenStack-api-environment?id=kb_article_view&sysparm_article=KB0051003 Accedido por última vez el 24/01/2025

[29] https://docs.OpenStack.org/horizon/latest/user/ Accedido por última vez 20/01/2025

[30] https://www.cvedetails.com/cve/CVE-2024-40767/ Accedido por última vez 28/01/2025

[31] https://nvd.nist.gov/vuln/detail/cve-2024-40767 Accedido por última vez 28/01/2025

[32] https://nvd.nist.gov/vuln/detail/CVE-2024-53916 Accedido por última vez 01/02/2025

[33] https://www.akamai.com/es/glossary/what-is-a-slowloris-ddos-attack Accedido por última vez 01/02/2025

[34] https://www.cloudflare.com/es-es/learning/ddos/ddos-attack-tools/slowloris/ Accedido por última vez 01/02/2025

[35] https://blog.nginx.org/blog/mitigating-ddos-attacks-with-nginx-and-nginx-plus Accedido por última vez 01/02/2025

[36] https://www.ionos.es/digitalguide/servidores/seguridad/arp-spoofing- ataques-desde-la-red-interna/ Accedido por última vez 01/02/2025

[37] https://europeanvalley.es/noticias/practica-de-auditoria-de-redes-dsniff/ Accedido por última vez 01/02/2025

[38] https://specs.OpenStack.org/OpenStack/neutron-specs/specs/yoga/allowed-address-pair-match-any-mac-address.html Accedido por última vez 01/02/2025

[39] https://help.ovhcloud.com/csm/en-public-cloud-compute-firewall-security?id=kb_article_view&sysparm_article=KB0051166 Accedido por última vez 01/02/2025

[40] https://docs.OpenStack.org/developer/dragonflow/specs/mac_spoofing.html Accedido por última vez 01/02/2025

[41]https://serbangroup.com/blog/todo-lo-que-debes-saber-sobre-cloud-computing-2023 Accedido por última vez el 04/02/2025

[42]https://libvirt.org/kbase/backing_chains.html#importance-of-proper-backing-chain-setup Accedido por última vez 05/02/2025

[43]https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/7/html/virtualization_deployment_and_administration_guide/sect-using_qemu_img-re_basing_a_backing_file_of_an_image Accedido por última vez 05/02/2025

[44] https://ubuntu.com/server/docs/apparmor Accedido por última vez 05/02/2025

[45] https://es.wikipedia.org/wiki/OpenStack Accedido por última vez 05/02/2025

[46] https://ubuntu.com/blog/k8s-native-microstack Accedido por última vez 05/02/2025

[47] https://www.arsys.es/blog/historia-cloud Accedido por última vez 05/02/2025

[48] https://www.redhat.com/es/topics/openstack Accedido por última vez 05/02/2025

[49] https://security.openstack.org/ossalist.html Accedido por última vez 02/02/2025

[50] https://nvd.nist.gov/ Accedido por última vez 02/02/2025

[51] https://www.cve.org/ Accedido por última vez 02/02/2025

[52]https://docs.redhat.com/en/documentation/red_hat_openstack_platform/8/html/storage_guide/index Accedido por última vez 05/02/2025

[53]https://docs.openstack.org/openstack-ansible/queens/reference/architecture/storage-arch.html Accedido por última vez 03/02/2025

[54] https://docs.openstack.org/cinder/latest/ Accedido por última vez 03/02/2025

[55] https://docs.openstack.org/cinder/latest/admin/index.html Accedido por última vez 03/02/2025

[56] https://www.digicert.com/es/what-is-ssl-tls-and-https Accedido por última vez 04/02/2025

[57] https://es.wikipedia.org/wiki/LUKS Accedido por última vez 04/02/2025

[58] https://isnum.com/glosario-ciberseguridad/vlan/ Accedido por última vez 04/02/2025

[59] https://suricata.io/ Accedido por última vez 04/02/2025

[60] https://keepcoding.io/blog/que-es-zeek/ Accedido por última vez 04/02/2025

[61] https://jumpcloud.com/es/blog/what-are-ssh-keys Accedido por última vez 04/02/2025

[62] https://www.arsys.es/blog/instalar-fail2ban Accedido por última vez 04/02/2025

[63]https://es.wikipedia.org/wiki/Wazuh Accedido por última vez 04/02/2025

[64]https://davinciti.com/elastic/?utm_source=social&utm_medium=twitter&utm_campaign=elastic-web&utm_content=elastic-davinci Accedido por última vez 04/02/2025

[65] https://www.zabbix.com/documentation/current/es/manual/introduction/about Accedido por última vez 04/02/2025

[66]https://formadoresit.es/prometheus-software-que-es-y-cuales-son-sus-principales-ventajas/ Accedido por última vez 04/02/2025

[67] https://ausum.cloud/que-es-grafana-y-como-se-usa-en-la-monitorizacion/ Accedido por última vez 04/02/2025

9. Apéndice siglas

CLI: *Command Line Interface* (Interfaz de Línea de Comandos)

IaaS: *Infrastructure as a Service* (Infraestructura como Servicio)

PaaS: *Platform as a Service* (Plataforma como Servicio)

SaaS: *Software as a Service* (Software como Servicio)

API: *Application Programming Interface* (Interfaz de Programación de Aplicaciones)

SNAT: *Source Network Address Translation* (Traducción de Direcciones de Red de Origen)

DNAT: *Destination Network Address Translation* (Traducción de Direcciones de Red de Destino)

QCOW2: *QEMU Copy On Write v2* (Formato de imagen de disco para virtualización)

VMDK: *Virtual Machine Disk* (Formato de disco virtual de VMware)

NAT: *Network Address Translation* (Traducción de Direcciones de Red)

PAT: *Port Address Translation* (Traducción de Direcciones de Puerto)

OVN: *Open Virtual Network* (Red Virtual Abierta)

OVS: *Open vSwitch* (Switch Virtual Abierto)

RBAC: *Role-Based Access Control* (Control de Acceso Basado en Roles)

LDAP: *Lightweight Directory Access Protocol* (Protocolo Ligero de Acceso a Directorios)

SAML: *Security Assertion Markup Language* (Lenguaje de Marcado para Aserciones de Seguridad)

OpenID: *OpenID Connect* (Protocolo de autenticación basado en OAuth 2.0)

DHCP: *Dynamic Host Configuration Protocol* (Protocolo de Configuración Dinámica de Host)

ARP: *Address Resolution Protocol* (Protocolo de Resolución de Direcciones)

VNC: *Virtual Network Computing* (Computación de Red Virtual)

RDP: *Remote Desktop Protocol* (Protocolo de Escritorio Remoto)

AMQP: *Advanced Message Queuing Protocol* (Protocolo Avanzado de Cola de Mensajes)

RPC: *Remote Procedure Call* (Llamada a Procedimiento Remoto)

HTTP: *HyperText Transfer Protocol* (Protocolo de Transferencia de Hipertexto)

HTTPS: *HyperText Transfer Protocol Secure* (Protocolo de Transferencia de Hipertexto

Seguro)

ISO: *International Organization for Standardization* (Organización Internacional de Normalización)

VM: *Virtual Machine* (Máquina Virtual)

CPU: *Central Processing Unit* (Unidad Central de Procesamiento)

RAM: *Random Access Memory* (Memoria de Acceso Aleatorio)

NIC: *Network Interface Card* (Tarjeta de Interfaz de Red)

SSH: *Secure Shell* (Protocolo Seguro de Acceso Remoto)

IP: *Internet Protocol* (Protocolo de Internet)

DNS: *Domain Name System* (Sistema de Nombres de Dominio)

TUN: *Network Tunneling Device* (Dispositivo de Túnel de Red)

VLAN: *Virtual Local Area Network* (Red de Área Local Virtual)

VXLAN: *Virtual Extensible LAN* (LAN Extensible Virtual)

GRE: *Generic Routing Encapsulation* (Encapsulación Genérica de Enrutamiento)

Geneve: *Generic Network Virtualization Encapsulation* (Encapsulación Genérica para Virtualización de Red)