UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Virtualización en la nube mediante OpenStack y Open Source MANO (OSM)

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Ortega Garcia, Joan

Tutor/a: Tavares de Araujo Cesariny Calafate, Carlos Miguel

CURSO ACADÉMICO: 2022/2023

Resum

El nostre món està de manera creixent informatitzat, amb aplicacions més complexes i que necessiten cada vegada més potència per a suportar les exigències d'aquestes aplicacions a gran escala. Des d'una botiga de roba en línia fins a aplicacions que usem diàriament com Netflix, Instagram o WhatsApp, solien funcionar en múltiples centres de dades, que es recolzaven en servidors gegants amb un gran consum, tant energètic com en maquinari específic, fet que provoca que siguen molt poc eficients i molt contaminants per a l'entorn.

Amb el temps, les empreses de programari es van adonar que aquest format era totalment insostenible, i que havien de buscar alguna manera de que, amb menys recursos, poder mantenir la càrrega de treball en nivells elevats el màxim temps possible. D'aquesta manera va sorgir la idea de la virtualització, i també el que ara anomenem "Virtualització de funcions en xarxa"o també anomenada NFV, proposada per l'Institut Europeu de Normes de Telecomunicacions (ETSI).

Aquest nou concepte té com a idea principal virtualitzar tots els elements de xarxa possibles per a aconseguir limitar al mínim el maquinari i programari específic possible, i així maximitzar la càrrega dels servidors virtualitzant petites màquines dins. Globalment, l'adopció de la virtualització en els seus múltiples dominis d'ús permet dividir eficientment la càrrega de treball dels servidors, i així maximitzar la càrrega que suporten els servidors, evitant deixar-los ociosos i eliminant aquesta part de maquinari que generava tants costos per a les empreses.

D'aquesta manera sorgeix un nou programari de codi obert amb la idea de que, amb el pas del temps, es creara una comunitat i es perfeccionara un programari capaç de virtualitzar tots els serveis en xarxa, i així aconseguir eliminar aquests problemes que generen els grans servidors.

Amb el pas del temps s'ha format la comunitat OpenStack, fent cada vegada més senzilla i accessible l'adopció d'aquesta tecnologia per a totes les empreses. A més, s'ha creat una plataforma més petita per a realitzar proves, i aconseguir que més persones s'uneixen a aquesta comunitat i ajuden a millorar les tecnologies.

En aquest TFG s'ha decidit estudiar aquesta tecnologia, i avaluar el seu potencial aprofitant l'ús de MicroStack per a poder realitzar proves de manera més senzilla. En concret, en l'estudi realitzat s'han abordat tant l'arquitectura, com la instal·lació i els requisits de sistemes més complexos, com és el cas d'OpenStack, a més d'investigar gran part del marc teòric d'aquests sistemes i realitzar certes proves de funcionament dins de MicroStack, per així demostrar la veritable utilitat i potencial d'aquests sistemes.

Paraules clau: NFV, OpenStack, MicroStack, MANO, Virtualització , Servidor, Xarxes

Resumen

Cada vez nuestro mundo se encuentra más informatizado, con aplicaciones más complejas y que necesitan más y más potencia para soportar la exigencia que tienen estas aplicaciones a gran escala. Desde una tienda de ropa online, hasta aplicaciones que usamos diariamente como Netflix, Instagram o WhatsApp, se ejecutaban en múltiples centros de datos, los cuales se apoyan en servidores gigantescos con un gran consumo, tanto energético como en hardware especifico, hecho que provoca que sean muy poco eficientes y muy contaminantes para el medio ambiente.

Con el tiempo, las empresas de software se dieron cuenta de que este formato era totalmente insostenible, y que debían de buscar alguna manera de que, con menor cantidad de recursos, poder mantener la carga de trabajo en niveles elevados el mayor tiempo posible. De esta manera surgió la idea de la virtualización, y también de lo que ahora llamamos "Virtualización de funciones en red.º también denominada NFV, propuesta por el Instituto Europeo de Normas de Telecomunicaciones (ETSI).

Este nuevo concepto tiene como idea principal virtualizar todos los elementos de red posibles para conseguir limitar al mínimo el hardware y software especifico posible, y así maximizar la carga de los servidores virtualizando pequeñas máquinas dentro. Globalmente, la adopción de la virtualización en sus múltiples dominios de uso permite dividir la carga de trabajo de los servidores de manera eficiente, y así maximizar la carga que soportan los servidores, evitando dejarlos ociosos y eliminando esa parte de hardware que generaba tanto gasto para las empresas.

De esta manera surge un nuevo software Open Source con la idea de conseguir que, con el paso del tiempo, se crease una comunidad se llegase a perfeccionar un software capaz de virtualizar todos los servicios en red, y así conseguir eliminar estos problemas que generan los grandes servidores.

Con el paso del tiempo se ha formado la comunidad OpenStack, haciendo cada vez que sea más sencilla y accesible la adopción de esta tecnología por todas las empresas. Además, se ha creado una plataforma más pequeña para realizar pruebas, y conseguir que más personas se unan a esta comunidad y ayuden a mejorar las tecnologías.

En este TFG se ha decidido estudiar esta tecnología, y evaluar su potencial aprovechando el uso de MicroStack para poder realizar pruebas de una manera más sencilla. En concreto, en el estudio realizado se han abordado tanto la arquitectura, como la instalación y los requisitos de sistemas más complejos, como es el caso de OpenStack, además de investigar gran parte del marco teórico de estos sistemas y realizar ciertas pruebas de funcionamiento dentro de MicroStack, para así demostrar la verdadera utilidad y potencial de estos sistemas.

Palabras clave: NFV, OpenStack, MicroStack, MANO, Virtualización , Servidor, Redes

Abstract

Our world is becoming increasingly digitalized, with more complex applications that require more and more power to support the demands of these large-scale applications. From online clothing stores to daily-use apps like Netflix, Instagram, or WhatsApp, they used to run in multiple data centers, relying on gigantic servers with significant energy and specific hardware consumption, which makes them highly inefficient and environmentally harmful.

Over time, software companies realized that this format was entirely unsustainable, and they needed to find a way to maintain high workloads with fewer resources for as long as possible. This gave rise to the concept of virtualization and what we now call "Network Function Virtualization" or NFV, proposed by the European Telecommunications Standards Institute (ETSI).

The main idea behind this new concept is to virtualize as many network elements as possible to minimize specific hardware and software, thus maximizing server load by virtualizing small machines within them. Globally, adopting virtualization in various use cases efficiently distributes server workloads, maximizing their capacity and eliminating the need for idle hardware that incurred significant expenses for companies.

This led to the development of open-source software aimed at creating a community capable of perfecting a system capable of virtualizing all network services over time, eliminating the issues associated with large servers.

Over time, the OpenStack community has formed, making it increasingly easy and accessible for companies to adopt this technology. Additionally, a smaller platform has been created for testing purposes, encouraging more people to join this community and contribute to technology improvements.

In this Bachelor's Thesis, the decision was made to study this technology and assess its potential by utilizing MicroStack for simpler testing. Specifically, the study addressed the architecture, installation, and requirements of more complex systems like OpenStack, in addition to researching much of the theoretical framework of these systems and conducting certain performance tests within MicroStack to demonstrate the true usefulness and potential of these systems

Key words: NFV, OpenStack, MicroStack, MANO, Virtualization, Server, Networks

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En esta primera parte del trabajo se hablará sobre lo que ha llevado a crear y desarrollar este trabajo centrado en el estudio de NFV Open Source MANO, así como los objetivos y la estructura que va a seguir esta memoria

1.1 Motivación

Últimamente, la industria de la informática está evolucionando mucho en lo que a servidores y comunicación se refiere. Esto ocurre porque cada vez se despliegan más aplicaciones en servidores, y comienza a ser necesario llevar estos servidores al límite para conseguir un buen funcionamiento de las aplicaciones o sistemas que se están desplegando, un ejemplo podría ser ChatGPT.

Por esto diferentes proveedores de servicios se dieron cuenta de que necesitan desarrollar mejores infraestructuras para conseguir llevar estas cargas de trabajo, sin tener costes de hardware desorbitados. Debido a estas surge la virtualización de servidores, y también este contexto la **Network Function Virtualization (NFV)**.

Si miramos algo atrás, la infraestructura tradicional de las redes es estática por culpa de que el hardware y el software deben de ir de la mano. Por otro lado, **NFV** permite que el software se despliegue en plataformas distintas, permitiendo que se gestione mediante diferentes interfaces, y sea compatible con múltiples fabricantes (eliminando así la compatibilidad de hardware). De esta manera se consigue eliminar hardware complejo y especializado, reemplazándolo por servicios software que se pueden ejecutar en servidores genéricos, ofreciendo mayor libertad, flexibilidad y facilidad.

La motivación que me ha llevado a crear este proyecto es poder entender cómo funciona este nuevo marco que pretende dejar obsoleta la infraestructura de las redes tal y como las conocemos hoy en día.

El estudio se centrará en el bloque **MANO**, vital dentro de NFV, al residir en este toda la gestión y orquestación del resto de bloques, y en el uso de la infraestructura de **MicroStack**, que nos permite realizar pruebas a una escala de usuario sin necesitar grandes recursos o un servidor más completo.

1.2 Objetivos

El objetivo es realizar un estudio exhaustivo de la plataforma NFV Open Source MANO (OSM). Por lo tanto, este trabajo no es únicamente un estudio teórico, sino que

gran parte de este trabajo es práctica al realizar la instalación de MicroStack para investigar y encontrar, en menor escala, las ventajas y límites de esta tecnología

Los hitos del proyecto son:

- Implementación del framework OSM.
- Implementación de OpenStack.
- Implementación de Microstack.
- Validación y pruebas en Microstack.

1.3 Estructura de la memòria

Esta memoria está dividida en siete capítulos, en los cuales se realiza una investigación en profundidad sobre OpenStack OSM y MicroStack, las principales tecnologías usadas en la virtualización de redes o NFV.

A continuación, vamos a realizar un pequeño resumen sobre lo que contiene cada capítulo dentro de esta memoria:

- **Capitulo 2 Tecnologías usadas**: en este capítulo se detallan y explican las tecnologías que se utilizan para realizar este trabajo; entre ellas se habla de NFV, el pilar central sobre lo que se basan todas las demás, OSM y OpenStack, como servicios OpenSource encargados de realizar la virtualización de las redes en los servidores, y MicroStack como entorno de pruebas pensado para pruebas de concepto en vez de servidores completos.
- **Capitulo 3 Despliegue de la arquitectura en servidores**: en él se detalla los requisitos necesarios para la instalación paso por paso del sistema de OSM y OpenStack en un servidor más complejo y completo a nivel de Hardware.
- **Capitulo 4 Despliegue de la arquitectura en equipos convencionales** : en él se detalla los requisitos necesarios para la instalación, y como instalar y preparar MicroStack para realizar las pruebas.
- **Capitulo 5 Evaluación de MicroStack**: en este apartado se realizan múltiples pruebas sobre la estructura de MicroStack, además de estudiar parte de su interfaz, y realizar una pequeña evaluación de resultados.
- **Capitulo 6 Análisis socio-económico** : en este apartado se realiza un estudio socioeconómico sobre la rentabilidad de utilizar el sistema NFV frente al uso de sistemas convencionales de servidores.
- **Capitulo 7 Conclusión** : En este último apartado se detallan las principales conclusiones relativas al sistema estudiado, destacando sus ventajas y desventajas, además de buscarle posibles ampliaciones, y comentar la relación del trabajo realizado con las asignaturas cursadas.

CAPÍTULO 2
Tecnologías usadas

2.1 NFV

Network Function Virtualization es un modelo de arquitectura de red que tiene como objetivo la virtualización de dichos servicios. Esta tecnología permite desacoplar las funciones de red tradicionales como los routers, firewalls, load balancers, etc., del hardware, y ejecutarlo únicamente como software en un servidor totalmente virtualizado.

En esta arquitectura las funciones se representan como **VNF** (Virtualized Network Function), que son instancias virtuales de software que se ejecutan en el agente encargado de la virtualización. Estas VNF son capaces de implementar la funcionalidad de varios servicios red, como los mencionados anteriormente.

Esta virtualización permite una gran ventaja respecto a las redes tradicionales, ya que proporciona una mayor flexibilidad, agilidad de despliegue, capacidad de gestión y potencia que las redes tradicionales. Además, elimina todas las dependencias que tienen los distintos hardware entre sí, logrando una mejor escalabilidad y una adaptabilidad más eficiente a medida que la demanda en la red se incrementa. Un ejemplo seria que un cliente nuevo que necesita una nueva VNF; en este caso el operador simplemente debe de activar una nueva máquina virtual y, con la misma facilidad que se creó la VM, se puede desactivar.

Esta arquitectura juega un papel fundamental dentro de las empresas, ya que permite reducir los costes de hardware para la creación de servidores, rebajándolos por el uso de la virtualización.

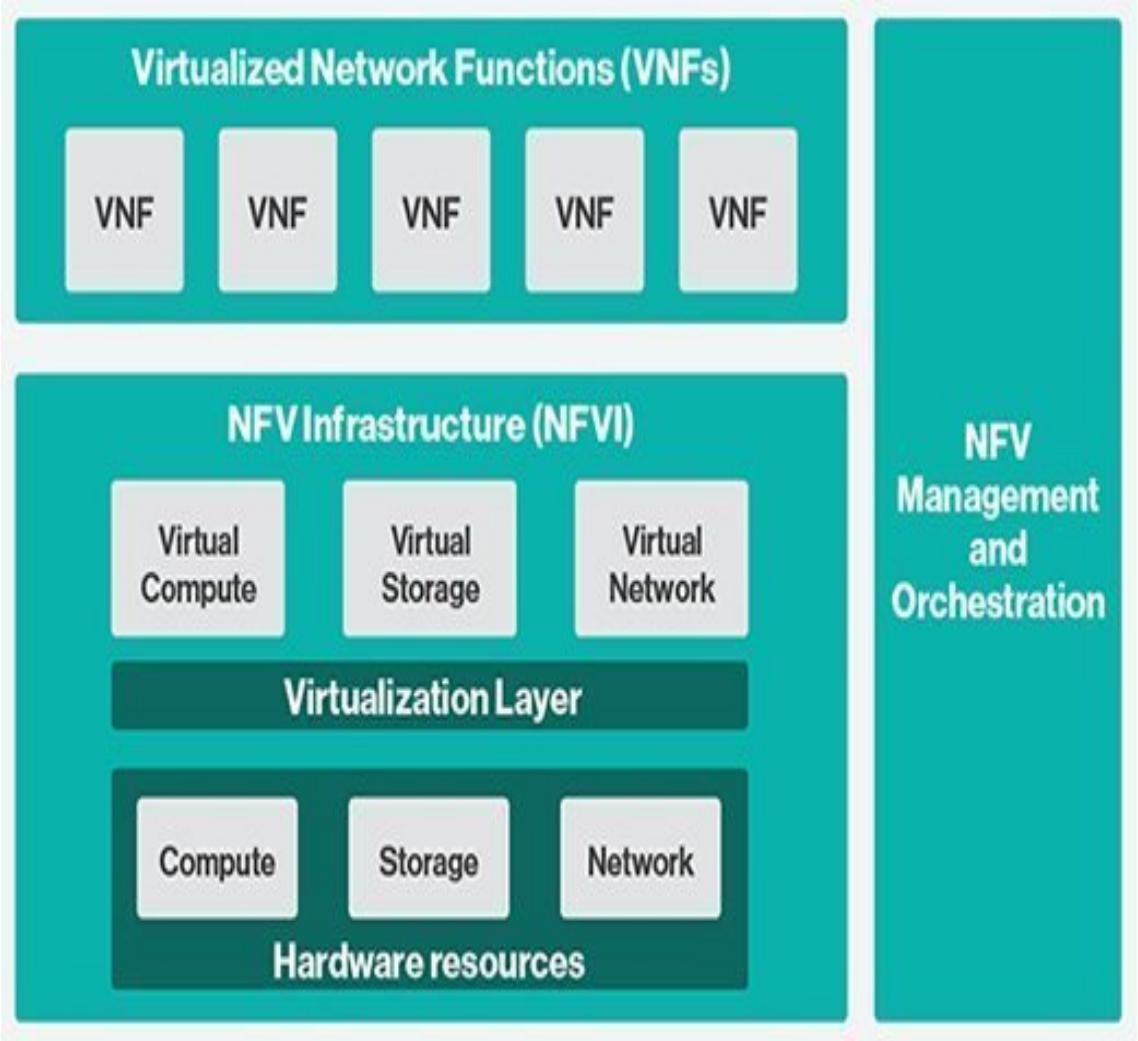

Figura 2.1: Arquitectura NFV.

2.1.1. Arquitectura

Dentro de la arquitectura NFV existen tres componentes principales: **Infraestructura de virtualización de funciones** (NFVi), **funciones de red virtualizadas** y **MANO**. Esto se puede observar en la imagen 2.1.

A continuación, procederemos a explicar cada una de estas tres partes que forman la arquitectura

Infraestructura de virtualización de las funciones de red (NFVi)

La infraestructura de virtualización de funciones o NFVI es una parte clave de la arquitectura NFV; esta proporciona la plataforma de hardware y software para alojar las máquinas virtuales. Este software hace función de hipervisor o de plataforma de gestión de contenedores, haciendo posible la virtualización y gestión de los recursos virtualizados.

Esta estructura consta de los siguientes componentes:

- Hardware físico, que proporciona la capacidad de procesamiento, por ejemplo, un servidor o un ordenador convencional.
- Dispositivos de almacenamiento, como discos, SSD ...
- Plataforma de virtualización: un hipervisor o una plataforma de gestión de contenedores.

Funciones de red virtualizadas (VNF)

Las VNFs son implementaciones de software de funciones de red que se pueden implementar dentro de una infraestructura de virtualización de red (NFVi). Estas se ejecutan en máquinas virtuales o contenedores para ejecutar el software necesario para obtener las funciones virtualizadas.

Gestión, automatización y organización de la red (MANO)

Este último componente es uno de los más importantes de la arquitectura NFV, además de ser una de las partes más importantes del estudio, ya que OSM es implementado en esta parte.

Este componente es el encargado de controlar todas las entidades dentro de la arquitectura NFV, tanto NFVi como las VNFs.

La arquitectura de MANO consta de tres componentes principales, Orquestador, gestor de funciones de red virtualizadas (VNFM), y elementos de gestión de infraestructura virtualizada (VIM)

El Orquestador (NFVO) es el componente principal de MANO, este coordina la implementación de servicios de red y la asignación de recursos para satisfacer requisitos de servicio. Este también es el responsable de gestión y coordinación de recursos.

El gestor de funciones de red virtualizadas (VNFM) es el responsable de la gestión del ciclo de vida de las VNF, incluyendo implementarlas, mantenerlas y eliminarlas.

El elemento de gestión de infraestructura virtualizada (VIM) es el responsable de gestionar y coordinar los recursos del hardware y software disponibles en la infraestructura. Puede haber varios VIM, y que cada uno se encargue de gestionar una parte distinta de la estructura.

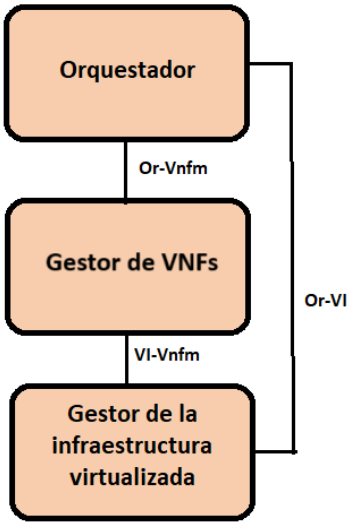

Figura 2.2: Arquitectura MANO.

2.2 OSM

Open Source MANO (OSM) es una comunidad de código abierto que tiene como objetivo entregar una pila de software libre MANO para la gestión y orquestación de la tecnología NFV, la cual pretende ser compatible con cualquier VNF, y accesible para todos los usuarios. Este software fue impulsado por múltiples proveedores de servicios, entre ellos Intel, Telefónica, Miranti, etc.

Figura 2.3: Open source mano.

Poco a poco OSM ha ido evolucionando desde su nacimiento en 2016, para adoptar principios de software nativo en la nube y adaptarse a avances en el sector, permitiendo así integrarse con múltiples VIMs, como VMWare, vCloud o OpenStack, siendo este último objeto de estudio en este trabajo.

La combinación de OpenStack y OSM para la capa MANO en NFV ofrece una solución de administración y orquestación eficiente y fácil de implementar, con una amplia comunidad de apoyo y colaboración en la innovación de características.

Gracias a todo esto, en los últimos años se ha avanzado mucho en este aspecto, hasta el punto de que grandes operadores han estado realizando pruebas sobre la incorporación de estos sistemas en sus centros de datos para ofrecer todo tipo de servicios en la nube.

2.2.1. Arquitectura

La arquitectura de Open Source Mano (OSM) es una arquitectura modular que ha ido evolucionando de manera significativa en la última década.

Estos componentes modulares trabajan juntos para proporcionar una solución completa. Cada módulo de la arquitectura se instala sobre un contenedor. Estos contenedores (o conteiner) son çajas"que disponen únicamente de los recursos necesarios para ejecutar su tarea; dentro de estos contenedores podemos observar múltiples componentes distintos que explicaremos a continuación.

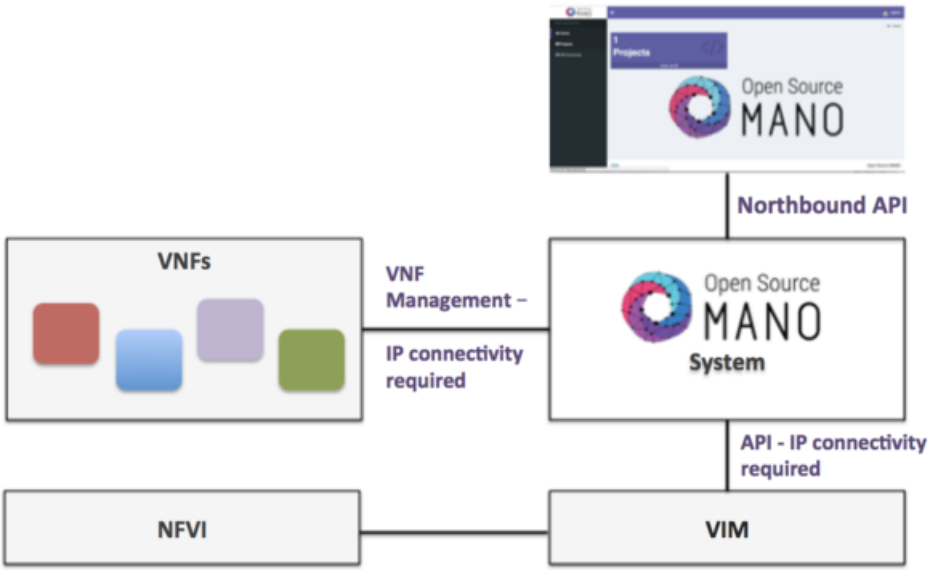

Figura 2.4: Arquitectura OSM.

Network Function Virtualization Orchestrator(NFVO)

Este módulo es el responsable de la orquestación y gestión de las VNFs, actuando como punto de entrada en la arquitectura para los Sistemas operativos de soporte y sistema de soporte de negocios (BSS). Además de esto es responsable de:

1. Gestionar el ciclo de vida y la ejecución de los servicios.

2. Validar y comprobar que tanto Network Service Descriptor (NSD) como VNF Descriptor (VNFD) se ejecutan como deben.

3. Soportar operaciones de las VNF y de los servicios red (NF) respecto a su actualización, creación, lectura y eliminación.

Para ser capaz de realizar todas las funciones, este software implementa dos componentes, que son los siguientes:

- **Resource Orchestrator (RO)**: este es uno de los componentes clave en la arquitectura de OSM, y se encarga de la provisión de servicios en un proveedor en una ubicación determinada. En otras palabras, es el módulo encargado de gestionar y coordinar los recursos.

- **Service Orchestrator (SO):** este componente se encarga de la orquestación de servicios extremo a extremo y de la administración del flujo de trabajo; es el responsable de definir los servicios de red incluyendo QoS, seguridad y routing, coordinación, gestión, y comunicación entre componentes.

VNF Configuration and Abstraction (VCA)

VCA es un componente esencial que se encarga de gestionar y orquestar la configuración y abstracción de las VNF.

Este componente está basado en çharms", que son conjuntos de scripts genéricos usados para implementar y operar software, adaptable para cada caso. Estos permiten una configuración fácil y una mejor gestión de las VNF, dando mayor flexibilidad y escalabilidad. De esta forma VCA es capaz de gestionar y actualizar las maquinas desde un unico punto.

Network Service to VNF Communication (N2VC

Este componente se encarga de la comunicación entre los servicios de la red virtualizados (NS) y las funciones virtualizadas (VNF).

N2VC nos proporciona una interfaz de comunicación entre el SO y las VNF, permitiendo que el SO gestione y controle las VNF a través del N2VC. Esto lo hace utilizando una combinación de Virtual Network Function Descriptor (VNFD) y Network Service Descriptor (NSD) para configurar y describir las VNF y los servicios de red.

N2VC es esencial para garantizar que toda la arquitectura funcione, al proporcionar una comunicación eficiente y coherente entre los servicios de red y sus funciones virtualizadas.

User Interface (UI)

Por último, podemos observar un conjunto de herramientas y elementos gráficos que permiten a los usuarios interactuar con la plataforma, gestionar servicios de red y las funciones de interfaz gráfica, la cual es más sencilla y fácil de usar, además de comprensible para los administradores de red.

Esta interfaz incluye una variedad de elementos visuales y herramientas como gráficos, estado, paneles de control, tablas de información y configuraciones, permitiendo visualizar y gestionar todo de modo mucho más sencillo.

Virtual Infrastructure Manager (VIM)

El Virtual Infrastructure Manager (VIM) es un componente clave en la arquitectura. El VIM es responsable de controlar y administrar todos los recursos de la infraestructura en una red, como cómputo, almacenamiento y red.

Las funciones principales de VIM son los siguientes:

1. Administrar el inventario de asignación de recursos virtuales a recursos físicos, permitiendo orquestación de asignación, actualización, y liberación de recursos.
2. Gestión de gráficos de reenvío de VNF con enlaces virtuales, redes y subredes, gestionando también las políticas de grupos de seguridad para garantizar el control de acceso.

3. Administración de un repositorio de recursos de infraestructura NFVI y recursos software como un hipervisor

En antiguas versiones del Open Source MANO el módulo VIM no estaba implementado, pero en las últimas actualizaciones la incorporaron para que sea más sencillo su uso.

Monitoring (MON)

Esta parte de la arquitectura se refiere a la monitorización y supervisión de sistemas para garantizar su rendimiento, disponibilidad y eficiencia, permitiendo así recopilar métricas, generar alertas, y visualizar datos en tiempo real para encontrar los problemas.

MON se puede aplicar en diferentes niveles de la arquitectura de un sistema, como nivel de infraestructura, de aplicación, o nivel de servicio. Algunas métricas relevantes incluyen el uso de CPU, memoria, almacenamiento, latencia, etc.

Policy Manager PM

Policy Manager o PM se encarga de la gestión de políticas y reglas en el entorno OSM. PM es responsable de establecer y hacer cumplir las políticas y reglas de uso de los sistemas, en las que se pueden incluir aspectos de seguridad de datos, de acceso a recursos, la utilización de esos recursos, etc.

Aunque en las primeras versiones de PM este módulo únicamente se encargaba de dar soporta a notificaciones, las cuales hacen de alarma o métrica para el módulo MON, con el paso del tiempo ha ido siendo ampliado, hasta lograr todas las funcionalidades de hoy en día.

2.2.2. Descriptores

Los descriptores son modelos de datos usados para describir funciones de red virtual y servicios de red. Estos permiten definir y describir las características y propiedades de VNF y NS.

Estos descriptores pueden incluir información de pruebas, proveedores, conectividad, etc., escribiendo todos sus resultados en formato YAML o XML.

Algún ejemplo de descriptor podría ser:

- Descripción de una VNF que incluya detalles de requisitos o interfaces.
- Descripción de un NS que indica como debe conectar y configurar múltiples VNF para lograr el servicio de red deseado.

2.3 OpenStack

OpenStack es la plataforma de Cloud computing de software libre más importante. Esta plataforma de computación proporciona un sistema operativo en la nube para controlar y gestionar grandes conjuntos de recursos de computación, almacenamiento y redes. Se suele implementar como servicio (IaaS).

Figura 2.5: OpenStack.

Algunas de sus características más importantes son:

1. **Escalabilidad**: permite escalar los recursos según necesidades del usuario, dando acceso a más recursos en segundos.

2. **API y panel de control**: ofrece una interfaz de programación de aplicaciones que permite gestionar y aprovisionar recursos por comandos y servicio RESTful.

3. **Orquestación y gestión de servicios**: aparte de las funciones que hemos definido en el apartado anterior, ofrece algunos componentes adicionales de orquestación.

2.3.1. Arquitectura para servidores

La arquitectura de Open Stack está formada por varios nodos fundamentales, que trabajan juntos para proporcionar los servicios de computación en la nube.

Existen múltiples tipos de nodos distintos. pero nosotros nos centraremos en cuatro nodos: el nodo controlador, nodo de cómputo, nodo de almacenamiento y nodos de redes, siendo los dos más importantes el nodo controlador y el nodo de cómputo.

Controller Node

El nodo de control o Çontroller Node", es el nodo central de la arquitectura de OpenStack y se encarga de gestionar y coordinar todos los servicios y componentes. Algunos de los servicios que se ejecutan en este nodo son:

- **Keystone**: servicio de autenticación y autorización.
- **Nova**: servicio de computación.
- **Neutron**: servicio de redes.
- **Glance**: servicio de imágenes.

Compute Node

El nodo de cómputo es el responsable de ejecutar las instancias de máquinas virtuales y proporciona la capacidad de cómputo en la red. Los nodos se comunican con el nodo de control para obtener instrucciones sobre creación, inicio y terminación de instancias. Cada nodo puede tener varios **hipervisores** para ejecutarse, como podrían ser un hipervisor KVM o uno de VMWare.

Storage Nodes

Los nodos de almacenamiento son los que proporcionan el almacenamiento persistente para instancias de máquinas virtuales y otros recursos de OpenStack. Se utilizan múltiples tecnologías para ofrecer almacenamiento en bloques (Ceph, Swift...). Este nodo se comunica con el nodo de control y el de cómputo para dar el almacenamiento necesario a cada zona.

Network nodes

Los nodos de red nos proporcionan la conectividad de red que requiere la arquitectura. Ejecutan servicios como **Neutron** para gestionar correctamente las redes y los enrutadores virtuales necesarios.

2.3.2. Servicios generales de Openstack

Como hemos dicho antes, cada nodo tiene múltiples servicios, que se encargan de una tarea muy concreta, y con la puesta en común de varios de ellos se permite que todo funcione correctamente.

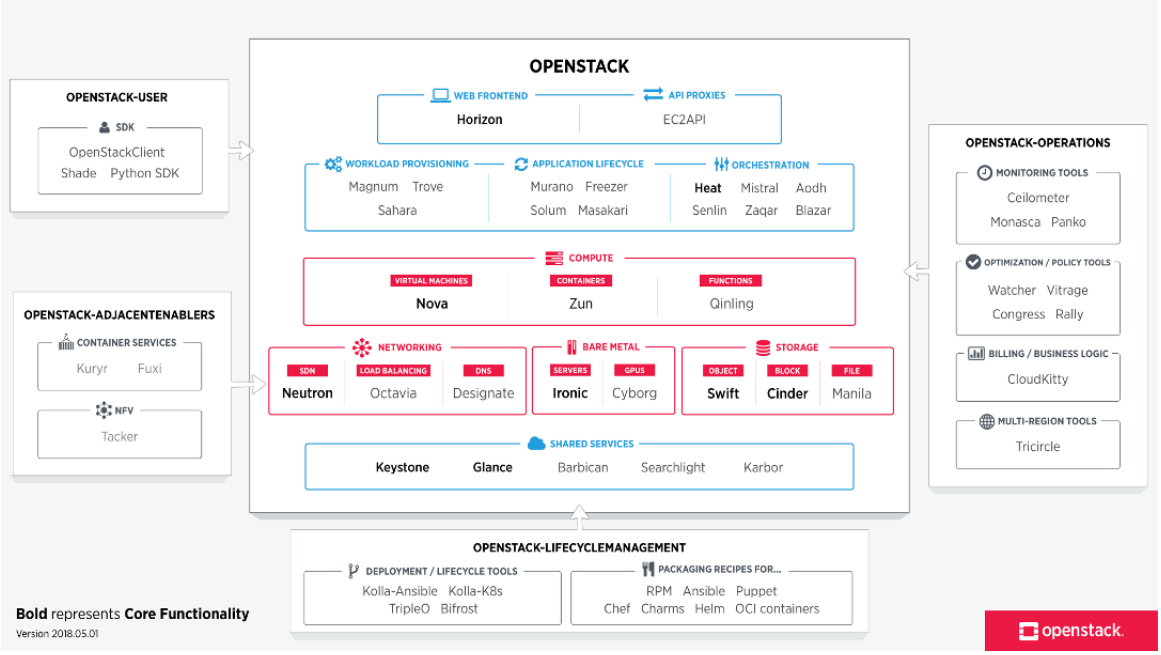

Figura 2.6: Servicios de OpenStack.

A continuación, describiremos alguno de los servicios con más relevancia dentro de la arquitectura, y que suelen estar presentes en todos los despliegues de OpenStack.

OpenStack Compute (Nova)

Este servicio se encarga de gestionar y ejecutar las máquinas virtuales dentro del apartado de OpenStack. Está compuesto por varios servicios y procesos que se ejecutan juntos para crear, iniciar o supervisar las instancias de máquinas virtuales; algunos ejemplos de estos servicios serían:

- **Servicio de planificación (scheduler)**.
- **Servicio de cómputo (compute service)**.
- **Servicio de red (network service)**.

OpenStack Object Storage (Swift)

Este componente se encarga del almacenamiento. Para ello utiliza un sistema distribuido y escalable de almacenamiento para dar redundancia y alta disponibilidad a los datos, almacenando en los contenedores mediante un API RESTful.

OpenStack Networking (Neutron)

Neutron se encarga de gestionar las redes virtuales en OpenStack. Proporciona una API para crear y gestionar redes virtuales, subredes, enrutadores y otros componentes de red necesarios. También permite la configuración de reglas de seguridad y asignación de dirección IP

OpenStack Image Service (Glance)

Glance se encarga de gestionar las imágenes de máquinas virtuales en OpenStack. Permite carga, almacenamiento y gestión de imágenes, que serán usados como plantillas.

OpenStack Identity (KeyStone)

El servicio de KeyStone, se encarga de la autenticación y autorización en OpenStack. Gestiona a los usuarios, proyectos y roles, dando tokens de acceso para permitir a los usuarios entrar.

2.4 MicroStack

MicroStack es una distribución de OpenStack creado por Canonical, empresa detrás de Ubuntu. Su tecnología es capaz de simplificar y facilitar el despliegue de OpenStack, convirtiéndolo en una tarea mucho más sencilla al ser desplegada con comandos de instalación de snap en una base de Ubuntu Server.

Esta tecnología se puede utilizar en múltiples contextos, desde micro-nubes (consisten en un pequeño número de nodos) hasta pipelines de integración y entrega continuas. No obstante, tiene muchas limitaciones, ya que, en el caso de necesitar un mayor número de nodos, como los que se puede encontrar en centros de datos, no sería capaz de alcanzar la potencia necesaria, por lo que se debería de pasar a distribuciones de OpenStack más completas.

2.4.1. Arquitectura

En lo que respecta a la arquitectura de MicroStack, está basada en la arquitectura de microservicios, siendo estos una colección de servicios autónomos y pequeños. Todo dentro de MicroStack viene empaquetado en pequeñas "snaps", y estas snaps son paquetes de software universal que contienen todas las dependencias requeridas, permitiendo que la app o servicio funcione en cualquier sistema Linux que soporte snaps. Esto tiene la ventaja de desacoplar completamente OpenStack del sistema operativo subyacente, lo que permite una mayor flexibilidad y evita conflictos de dependencia.

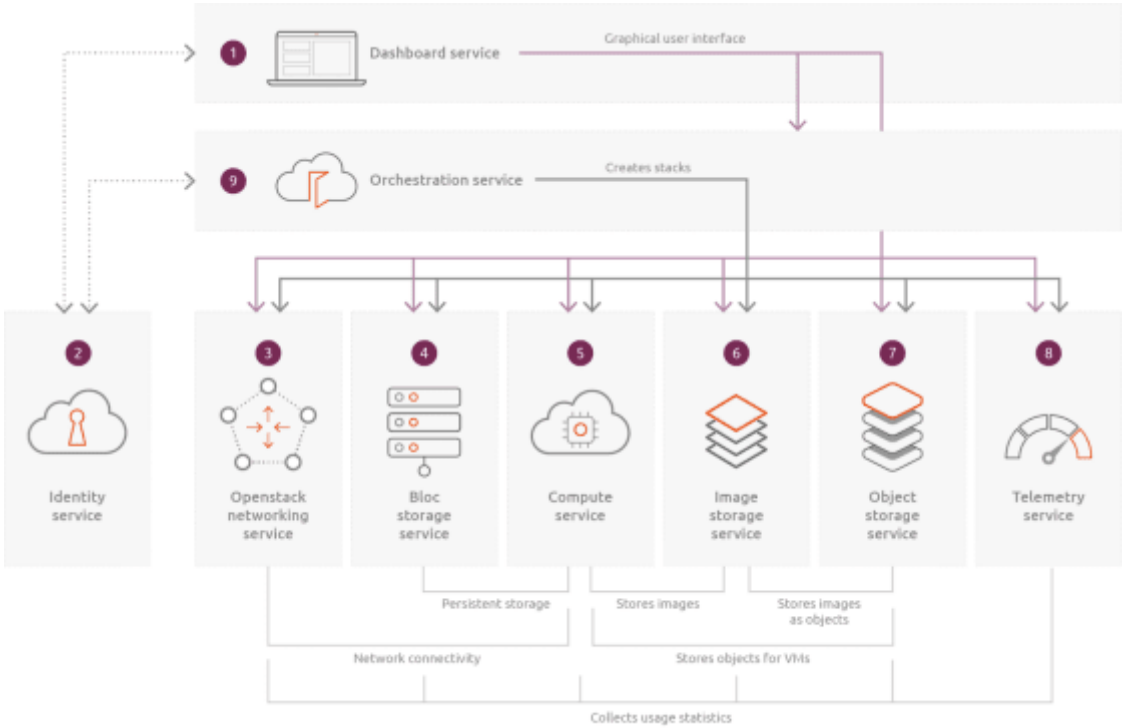

Figura 2.7: Arquitectura de MicroStack

CAPÍTULO 3
Despliegue de la arquitectura en servidores

En esta parte de la memoria se describen los pasos necesarios para conseguir montar la estructura de necesaria en un servidor, que incluye la creación de la plataforma OSM y OpenStack. Aunque vayamos a realizar las pruebas en Microstack, es muy interesante saber cómo se realizaría la instalación de la arquitectura en un servidor real al ser un proceso más complejo.

En último lugar indicaremos como integrar OSM y OpenStack para que se comporten como una única herramienta, y conseguir así una arquitectura MANO sólida.

3.0.1. Aclaraciones previas

Antes de comenzar la instalación, es importante definir la estructura que vamos a seguir, y en qué sistema operativo nos estamos basando.

En primer lugar, todas las instalaciones se realizarán dentro de un ordenador con Ubuntu como SO principal, por la poca cantidad de recursos, y porque toda la instalación la vamos a realizar sobre máquinas virtuales para:

- **Minimizar riesgos**: al estar realizando pruebas, no es muy seguro utilizar un hipervisor, ya que en algún punto del desarrollo nos obligará a empezar de cero o generará problemas al equipo.
- **Seguridad**: posibilidad de realizar copias de seguridad más fácilmente.

Una vez sabemos que vamos a realizar el despliegue usando máquinas virtuales, tendremos que saber cuál utilizaremos para realizar las pruebas: VirtualBox, KVM o VMWare. En este caso, al tratarse de un equipo Ubuntu, se ha optado por utilizar KVM, ya que en Linux es el que mayor rendimiento ofrece, siendo es superior a otras herramientas.

Instalación de KVM

En primer lugar, vamos a instalar KVM en nuestra máquina Ubuntu. Para realizar esto seguiremos los siguientes pasos:

1. Abrimos un terminar y ejecutamos el siguiente comando:

```
egrep '(vmx|svm)' /proc/cpuinfo
```

En caso de que la respuesta sea vacía, el hardware no es capaz de virtualizar, y no se podrá instalar las herramientas.

2. En caso de poder virtualizar colocaremos el siguiente comando que instalará KVM junto a algunas herramientas como **qemu** (emulador de hardware) o **libvirt** (biblioteca para interactuar con el API).

```
joan@joan-MS-7A34:~$ sudo apt install qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils virtinst
```

Figura 3.1: Comando de instalación de KVM.

3. Una vez instalado, verificaremos que todo se ha instalado correctamente ejecutando el comando de la figura 3.2

```
joan@joan-MS-7A34:~$ kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
```

Figura 3.2: Comando de verificación de KVM.

En caso de que todo haya funcionado correctamente, nos aparecerá el mensaje que se ve al final de la figura anterior.

4. Por último, y para que se nos haga todo el proceso más sencillo, podemos instalar en el equipo un gestor de máquinas virtuales, que hará más sencillo y visual todo el proceso de crear las máquinas virtuales necesarias para este proyecto.

Creación de máquinas virtuales

Este proceso se ha seguido a la hora de crear cualquier máquina virtual necesaria para este proyecto, y lo único que variará entre una máquina a otra son los requisitos necesarios.

1. En primer lugar crearemos una nueva máquina virtual con un medio de instalación local, el cual hemos instalado previamente en nuestro dispositivo; en todos los casos usaremos Ubuntu Server 20.04.

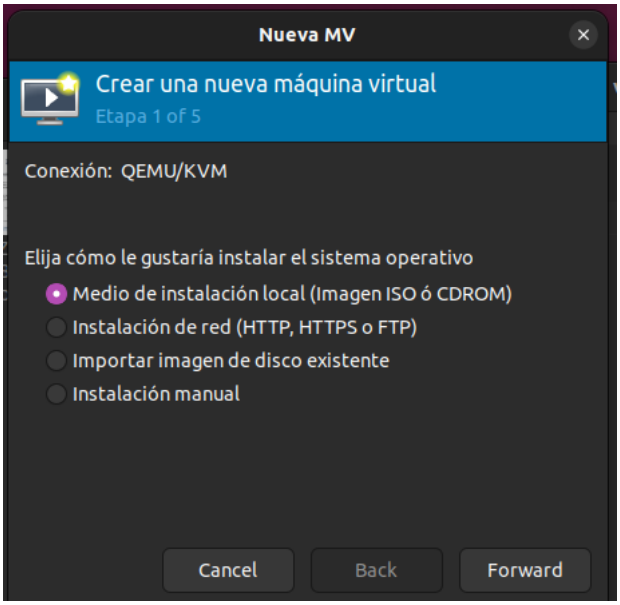

Figura 3.3: Creación de la máquina virtual.

2. En segundo lugar escogeremos la ISO y la versión a instalar en la máquina virtual.

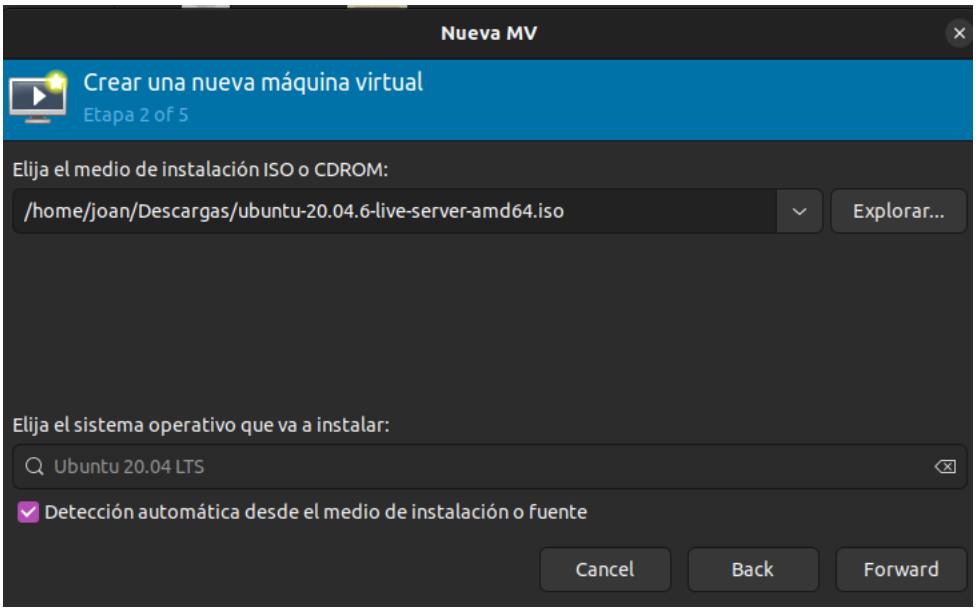

Figura 3.4: Eligiendo ISO de la máquina virtual.

3. En tercer lugar, escogeremos la cantidad de memoria y de núcleos de CPU que tendrá la máquina virtual

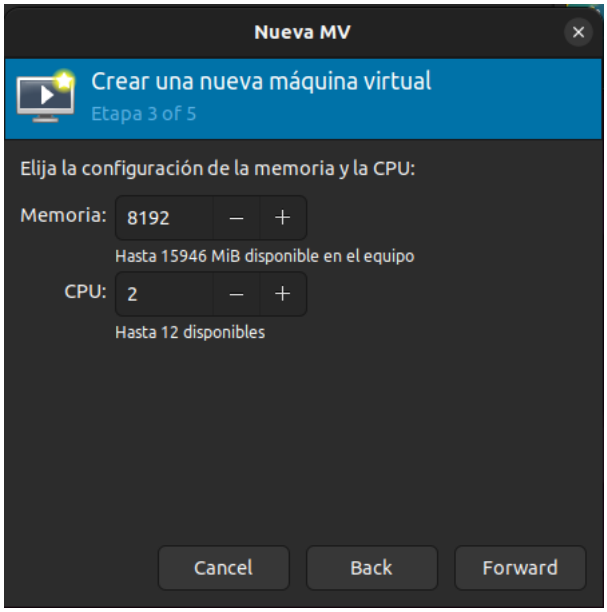

Figura 3.5: Eligiendo configuración de memoria y CPU.

4. En cuarto lugar crearemos una imagen en el disco para la máquina virtual, escogiendo algo más del espacio requerido por las partes de la instalación.

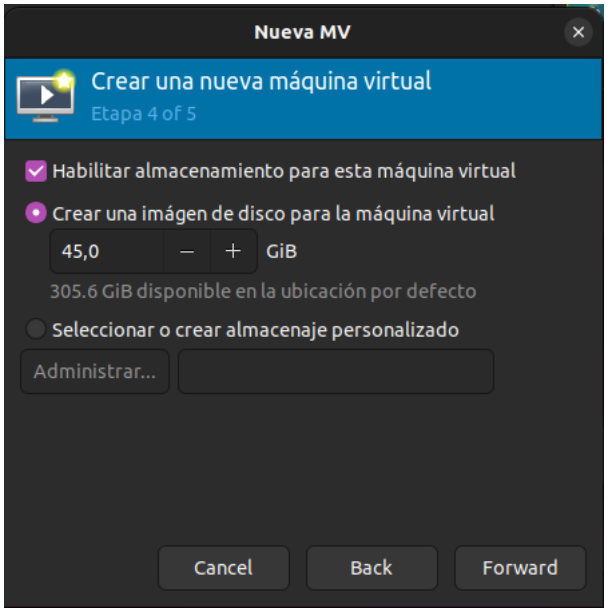

Figura 3.6: Eligiendo configuración de almacenamiento.

5. Por último le colocaremos un nombre a nuestra máquina virtual y la crearemos.

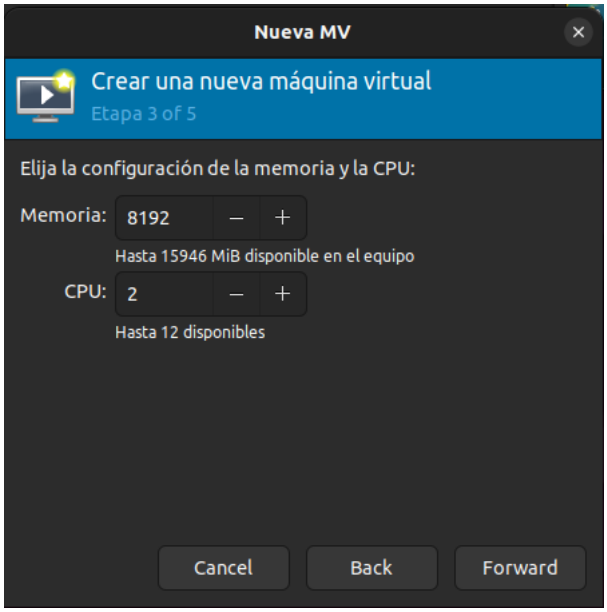

Figura 3.7: Paso final de creación de la maquina.

Instalación de Ubuntu Server

Una vez hemos creado la máquina virtual y se ha arrancado con Ubuntu Server, comienza el proceso para instalar Ubuntu Server con las extensiones que son necesarias para todas las máquinas.

1. En primer lugar escogeremos el idioma tanto del servidor como la disposición del teclado que queremos utilizar. En este paso podemos escoger el idioma en el que más a gusto estemos.

2. En segundo lugar decidiremos si queremos cambiar la disposición del espacio, que en este caso no será necesario al querer utilizar todo el espacio para una única función.

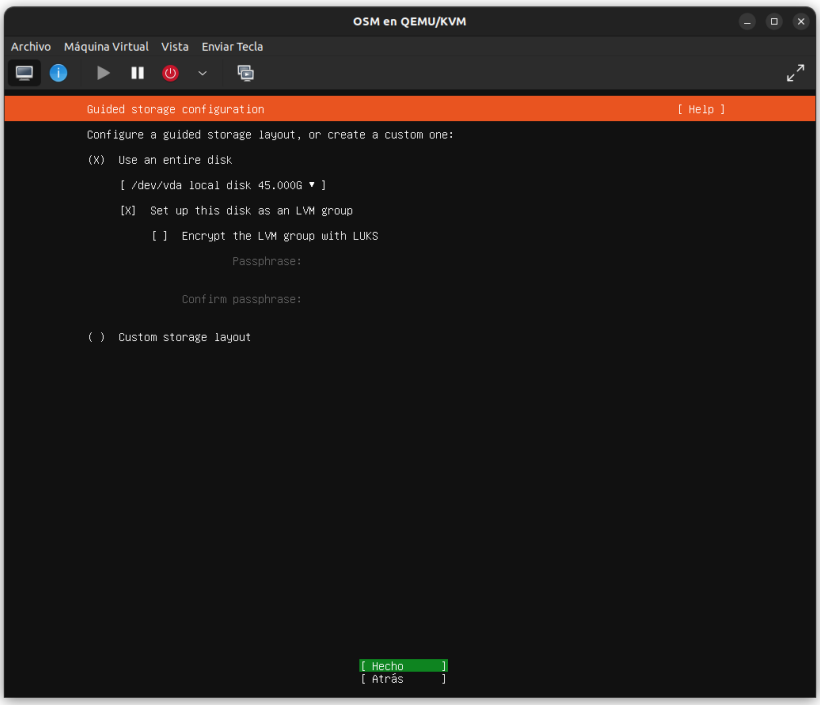

Figura 3.8: Configuración del almacenamiento.

3. En tercer lugar procederemos a instalar tanto un servidor SSH como todos los programas necesarios para realizar las funciones que queramos, en este caso al tratarse del servidor para OSM no instalaremos ninguno.

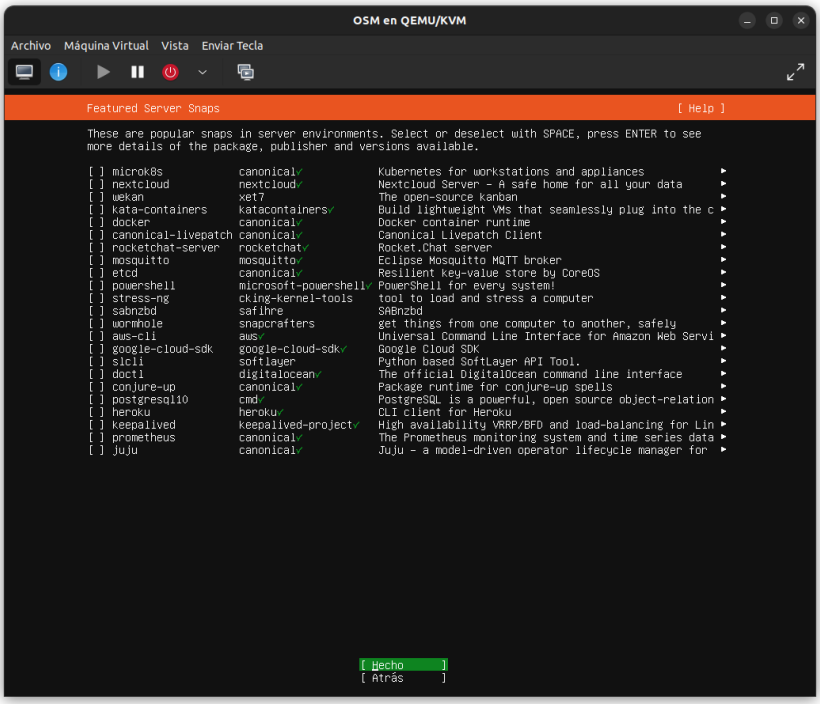

Figura 3.9: Instalando programas necesarios.

4. En cuarto lugar colocaremos los nombres del servidor, del usuario y una contraseña para poder acceder tanto al sistema como a la versión de administrador.

5. Por último, nos aparecerá un resumen sobre la instalación que hemos realizado, se instalará y se reiniciará el servidor. Una vez reiniciado ya podremos utilizar nuestra nueva máquina virtual.

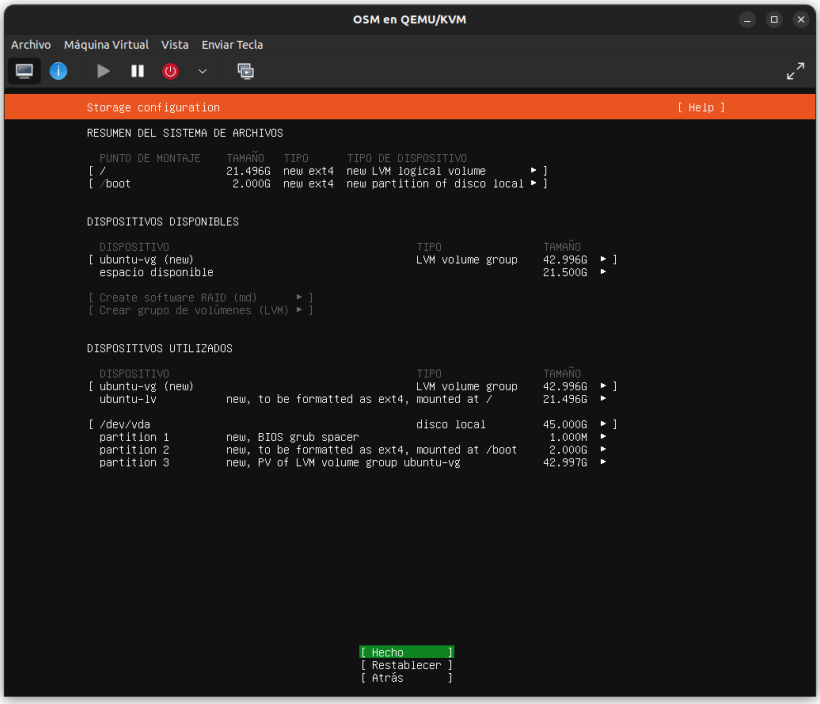

Figura 3.10: Resumen de la instalación.

3.1 OSM

En este apartado, hablaremos del procedimiento para instalar uno de los apartados principales del proyecto OSM. Este software cuenta con múltiples versiones de software, pero nosotros trabajaremos con la versión más estable y actual, la versión THIRTEEN.

Esta versión presenta una arquitectura escalable, utilizando una versión nativa de la nube de Apache Airflow y Prometheus. Incluye mejoras significativas en áreas clave respecto a versiones anteriores en cuanto a servicios de red, en su ciclo de vida, y en la experiencia de instalación

3.1.1. Requisitos

Con el paso del tiempo, los recursos de instalación han ido disminuyendo poco a poco, ya que era necesario conseguir las mismas funcionalidades con la mitad de los recursos, cosa que se ha ido consiguiendo.

A continuación, veremos tanto los requisitos mínimos como los recomendados para realizar la instalación de OSM:

- Mínimos: 2 CPUs, 6Gb RAM, 40 GB de almacenamiento
- Recomendado: 2CPUs, 8 Gb RAM, 40GB de disco.

Nosotros nos hemos ceñido a los requisitos recomendados basándonos en la imagen Ubuntu Server 20.04 de 64 bits, con lo cual hemos creado una máquina virtual desde KVM para instalar el software de OSM con las siguientes características:

- Núcleos de CPUs: 2 núcleos
- Memoria RAM : 8GB
- Disco duro: 40GB
- Sistema Operativo: Ubuntu 20.04 (64 bits)

3.1.2. Instalación

En esta sección vamos a realizar los pasos para instalar OSM dentro de una máquina virtual con Ubuntu Server. Esta máquina virtual se ha creado siguiendo los pasos del capítulo 3.0.1 y con los requisitos recomendados para la versión **THIRTEEN** de OSM.

Una vez tenemos la máquina operativa, realizaremos los siguientes pasos para instalar y poner en funcionamiento OSM:

En primer lugar nos colocaremos en nivel de super usuario en nuestro servidor utilizando “sudo -i” e indicaremos la contraseña de administrador que hemos colocado en la instalación.

En segundo lugar, ejecutaremos los siguientes comandos uno detrás del otro.

```
1  wget http://osm-download.etsi.org/ftp/osm-13.0-thirteen/install_osm.sh
2  chmod +x install_osm.sh
3  ./install_osm.sh
```

El primero de los comandos (wget) sirve para descargar archivos desde la web, concretamente estamos instalando el archivo para instalar la versión 13 de OSM desde la URL oficial de OSM.

El segundo comando (chmod...) lo que hace es cambiar los permisos del archivo para que pueda ejecutarse y convertir así ïnstall-osm.sh.[en] un script ejecutable.

Y por último ejecutamos el script para así instalarlo. Este último comando se puede cambiar por dos comandos para así obtener un log de lo que ha sucedido; este paso es interesante usarlo para saber si ocurre algún error durante el proceso de instalación.

```
./install_osm.sh 2>&1 | tee osm_install_log.txt
```

Una vez ejecutados los comandos, comenzará la instalación del programa; al poco nos pedirá permiso para instalar algunos programas necesarios para instalar OSM, y a esto le responderemos indicando los programas que queremos que se instale (ver figura 3.11).

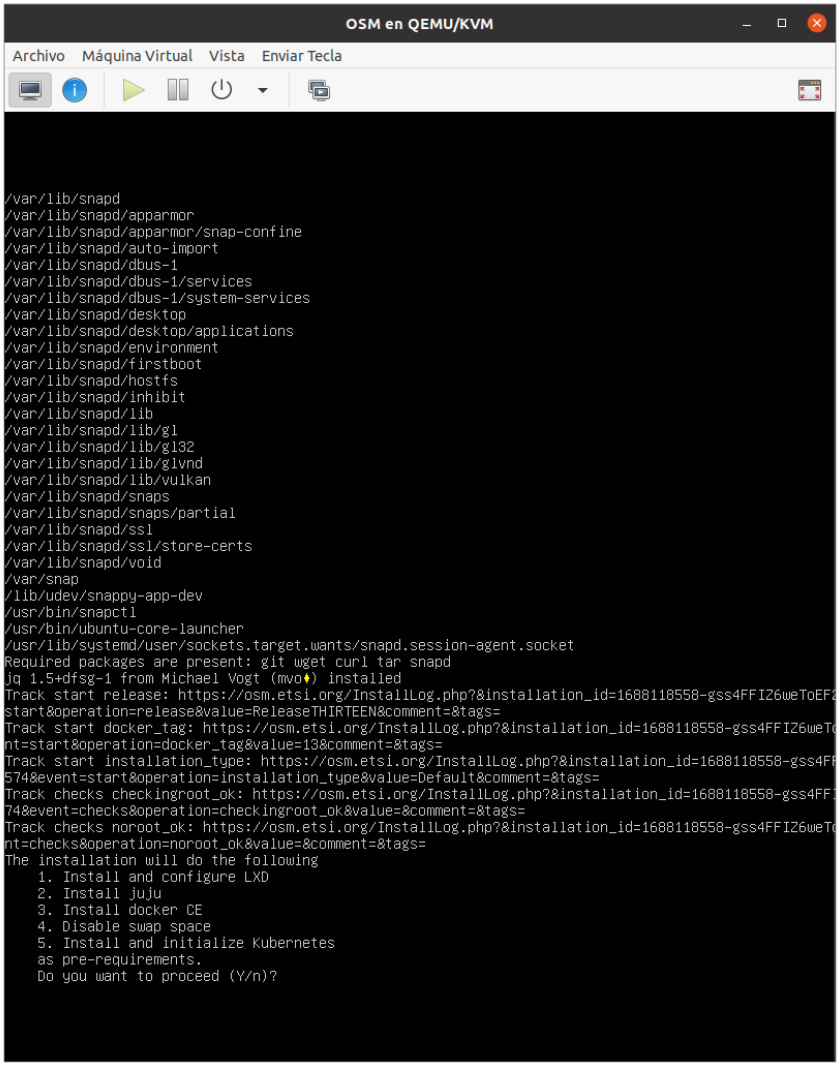

Figura 3.11: Programas a instalar en OSM.

Una vez aceptado el proceso anterior, comenzara la instalación de OSM, y pasado unos minutos de la instalación veremos algo como en la figura **??**; esto no debe de preocuparnos ya que se trata de unas pruebas de lanzamiento que se realizan durante el proceso de instalación para comprobar que todo funcione a la perfección.

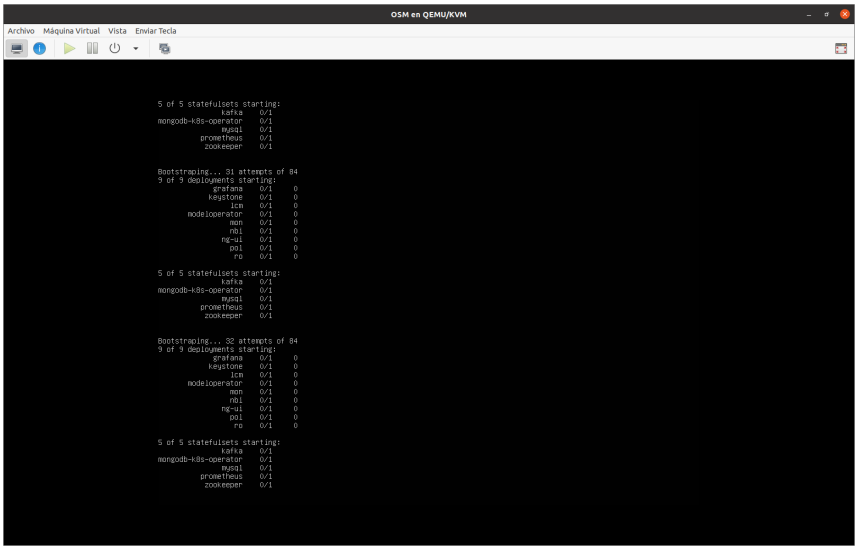

Figura 3.12: Pruebas automáticas en la instalación de OSM.

Una vez pasamos estas pequeñas pruebas nos indicara como se ha completado y instalado todo con éxito, como vemos en la imagen 3.13 .

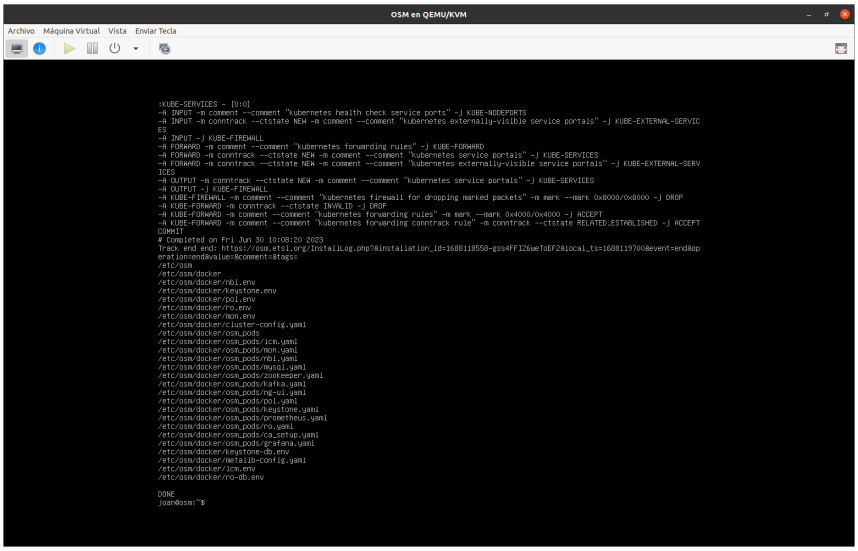

Figura 3.13: Final de la instalación de OSM

Una vez termine la instalación podemos comprobar que todo funciona correctamente entrando desde nuestra máquina host a la IP de la máquina virtual desde un navegador; podremos observar una pestaña de inicio de sesión para acceder a toda la plataforma de gestión de OSM.

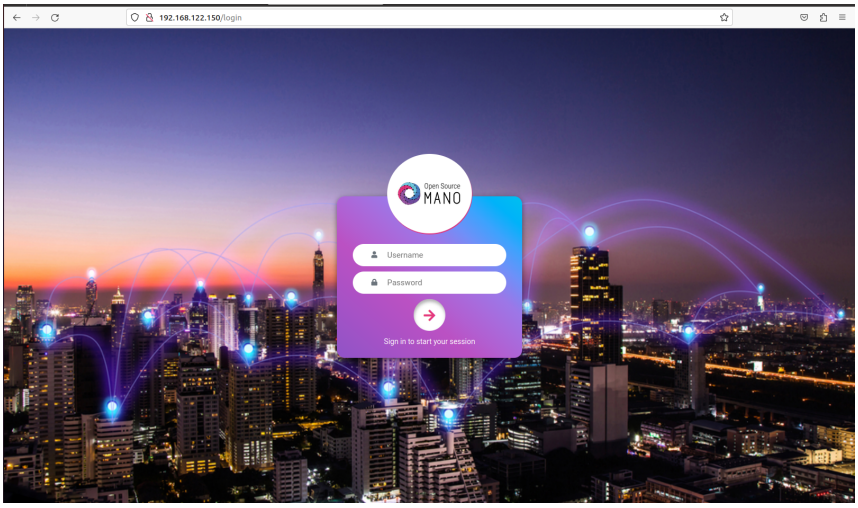

Figura 3.14: Ventana de login de OSM.

Una vez accedemos nos encontramos con un dashboard parecido al siguiente, en el cual encontraremos múltiple información. La siguiente imagen proviene de la web oficial de Open Source MANO, ya que esta parte del estudio se realiza desde MicroStack.

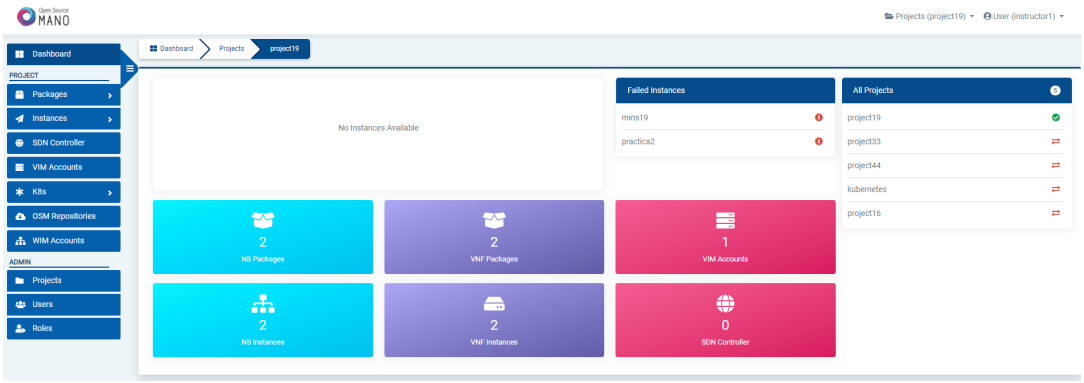

Figura 3.15: Dashboard de OSM.

3.2 OpenStack

El procedimiento de instalación de OpenStack puede variar dependiendo de nuestra distribución de Linux, y de las herramientas que queramos implementar a nuestra instalación. En los siguientes apartados explicaremos detalladamente los requisitos y paso por paso todo lo necesario para crear nuestra red en estas últimas versiones de OpenStack.

3.2.1. Requisitos

Como la arquitectura de OpenStack no está del todo predefinida, ya que podemos elegir la cantidad de nodos a implementar, estos requisitos varían dependiendo del número de nodos distintos que instalemos.

Como en este trabajo se basa en realizar algunas pruebas, nos hemos centrado en los dos nodos imprescindibles, el nodo *compute* y el nodo *controller*.

Requisitos del nodo de control

Como ya hemos explicado en apartados anteriores, el nodo de control (Node controller) es el responsable de gestionar y coordinar todos los servicios de OpenStack. Los requisitos pueden variar según el tamaño y la carga de trabajo que le coloquemos, pero en nuestro caso nos ceñiremos a los requisitos recomendados, que son:

- CPU: 2 núcleos de CPU dedicados
- RAM: 4 GB de RAM
- Almacenamiento: 40 GB de espacio de disco

También debemos tener en cuenta que estos son los requisitos "mínimos"; después, dependiendo de las necesidades que tengamos, pueden aumentar de manera substancial.

Requisitos del nodo de cómputo

Este nodo, como ya hemos explicado en apartados anteriores, es el nodo encargado de ejecutar las instancias de máquinas virtuales. Los requisitos, igual que en el nodo de control, pueden variar según el tamaño y la carga de trabajo que tenga, aunque normalmente se recomienda comenzar por:

- CPU: 2 núcleos de CPU dedicados
- RAM: 4 GB de RAM
- Almacenamiento: 40 GB de espacio de disco

3.2.2. Instalación

El proceso de instalación de OpenStack para servidores es muy extensa y compleja. No obstante, lo que se pretende conseguir es una arquitectura similar a la que encontramos en la figura 3.16.

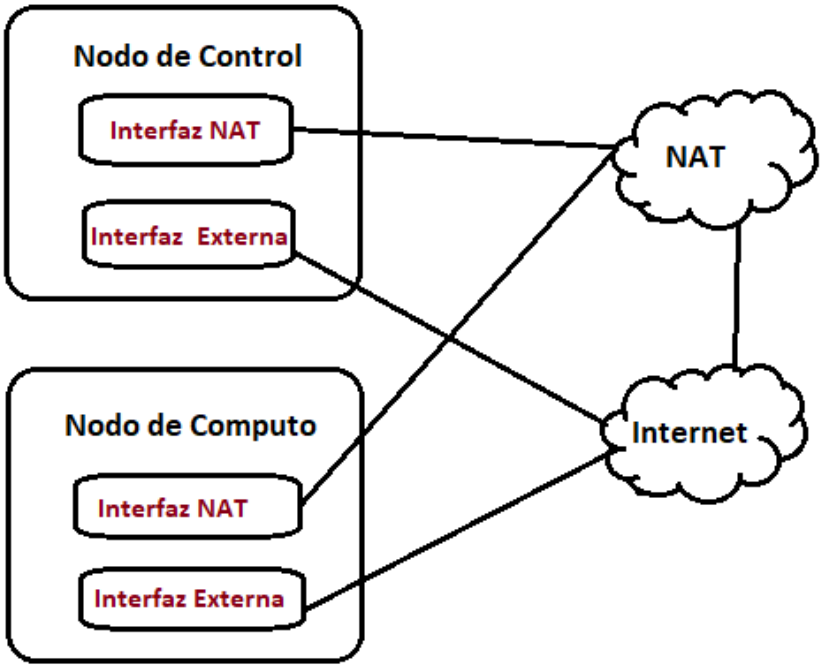

Figura 3.16: Arquitectura de red sencilla en OpenStack.

En primer lugar, debemos de crear las dos máquinas virtuales con los requisitos recomendados para cada nodo.

Una vez creadas las máquinas virtuales procederemos a hacer la instalación del Ubuntu Server, igual que hicimos en el apartado anterior, pero en ambos casos nos guardaremos las IP de cada máquina virtual para un futuro paso.

Figura 3.17: Configuración de la dirección IP del nodo de cómputo.

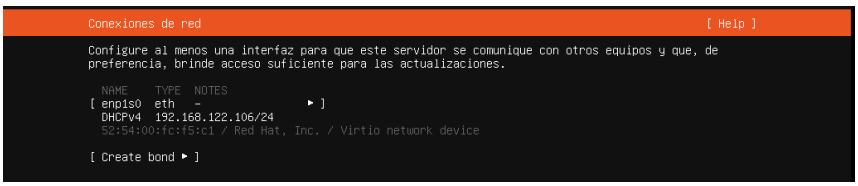

Figura 3.18: Configuración de la dirección IP del nodo de cálculo.

Al terminar la instalación de los sistemas, instalaremos en ambas máquinas virtuales los componentes necesarios para OpenStack. Estos componentes suelen variar dependiendo del sistema operativo que se use, pero normalmente suele incluir algunos componentes como Python, RabbitMQ, MariaDB, y otros paquetes. Algunos de estos paquetes serían los siguientes:

```
1 yum install -y centos-release-openstack-queens
2 yum install -y python-openstackclient
3 yum install -y openstack-selinux
```

```
4     yum install -y rabbitmq-server
```

Una vez terminada la instalación de los componentes en ambos nodos, procederemos a configurar primero el nodo de control.

El nodo de control es el cerebro de una instalación de OpenStack, ya que maneja todas las operaciones de orquestación, y es responsable de coordinar todas las partes de OpenStack. Configurar este nodo implica instalar y configurar múltiples servicios explicados anteriormente, entre ellos KeyStone, Glance, Nova, Neutron y Horizon.

Podemos instalar estos servicios mediante los siguientes comandos:

```
1  sudo apt-get install -y keystone apache2 libapache2-mod-wsgi
2  sudo apt-get install -y glance
3  sudo apt-get install -y nova-api nova-conductor nova-consoleauth nova-
       novncproxy nova-scheduler
```

Una vez instalados todos los componentes, debemos habilitarlos y iniciarlos. Para ello podemos utilizar los siguientes comandos:

```
1  sudo systemctl enable apache2.service
2  sudo systemctl start apache2.service
3  sudo systemctl enable glance-api.service glance-registry.service
4  sudo systemctl start glance-api.service glance-registry.service
5  sudo systemctl enable nova-api.service nova-consoleauth.service nova-
       scheduler.service nova-conductor.service nova-novncproxy.service
6  sudo systemctl start nova-api.service nova-consoleauth.service nova-
       scheduler.service nova-conductor.service nova-novncproxy.service
```

Al terminar la instalación del nodo de cómputo, pasaremos al nodo de cálculo, el cual es donde se ejecutan las cargas de trabajo en OpenStack. En este nodo es donde instalaremos el servicio NOVA, y lo configuraremos para que trabaje en paralelo con el nodo de control. Para ello, en primer lugar instalaremos el servicio:

```
1    sudo apt-get install -y nova-compute
```

Seguidamente habilitaremos e iniciaremos los servicios:

```
1    sudo systemctl enable nova-compute.service
2    sudo systemctl start nova-compute.service
```

Una vez terminados estos pasos, pasaremos a verificar en ambos nodos que todo está correcto ejecutando el comando:

```
1     nova-manage service list
```

el cual comprueba que todo esté funcionando perfectamente.

Estos pasos corresponden a la instalación más básica de ambos nodos en un sistema Ubuntu Server. La configuración real es mucho más compleja al tener que revisar las necesidades específicas de todo lo que se va a alojar, y tener que realizar toda la integración con OSM.

CAPÍTULO 4

Despliegue de la arquitectura en equipos convencionales

En este apartado de la memoria describiremos los pasos necesarios para desplegar la estructura de Microstack a nivel local, y realizaremos múltiples pruebas para entender mejor su funcionamiento y rendimiento.

4.1 Microstack

MicroStack es una solución de instalación de OpenStack simplificada y de un solo nodo, diseñada para entornos de desarrollo, pruebas y demostraciones. Esta solución es un todo en uno de OpenStack, incluyendo los servicios esenciales preconfigurados y empaquetados en snaps para una mejor instalación y configuración. Esta instalación no es adecuada para producción o para servicios a gran escala, pero no obstante es una gran manera de explorar openstack sin pasar por el proceso complejo, y con mayor necesidad de recursos, ya que puede ser ejecutado en un único nodo, siendo ideal para entornos de desarrollo y pruebas para pequeñas empresas que puedan necesitar algo más privado y asequible; además, al incorporar una interfaz web, hace más sencilla la gestión de los recursos.

En lo que a nuestro estudio concierne, nosotros vamos a realizar la instalación con la versión 1.10.0.

4.1.1. Requisitos

En el tema de requisitos, Microstack está mucho más predefinido que OpenStack al tratarse de un entorno pensado para pruebas, con lo cual tiene unos requisitos mínimos más claros que en el caso anterior, y que son los siguientes:

- CPU: 4 núcleos de CPU dedicados
- RAM: 16 GB de RAM
- Almacenamiento: 50 GB de espacio de disco

No obstante, en la instalación que vamos a realizar en el siguiente apartado hemos ampliado el almacenamiento de disco de 50GB a 100GB para realizar alguna prueba más compleja.

4.1.2. Instalación

El proceso de instalación de Microstack es mucho más sencillo que los procesos de OSM o OpenStack, al tener la base de snaps que ya comentamos.

En primer lugar, creamos una máquina virtual desde KVM con los requisitos especificados y con Ubuntu Server 22.04, versión de Ubuntu más estable para la versión de Microstack que vamos a instalar.

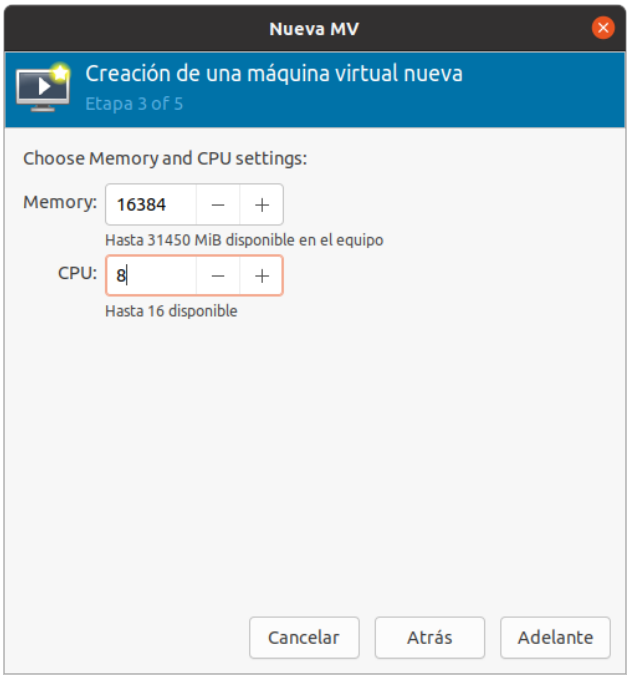

Figura 4.1: Requisitos de la máquina virtual Microstack.

Una vez creada la máquina virtual y puesta en marcha, lo único que deberemos de hacer será guardar la IP (ver figura 4.2) que tenga el servidor, ya que esta será necesaria para poder entrar a algunas máquinas virtuales o al gestor gráfico desde nuestro navegador; no será necesario modificar ni instalar nada desde el proceso de instalación del sistema operativo.

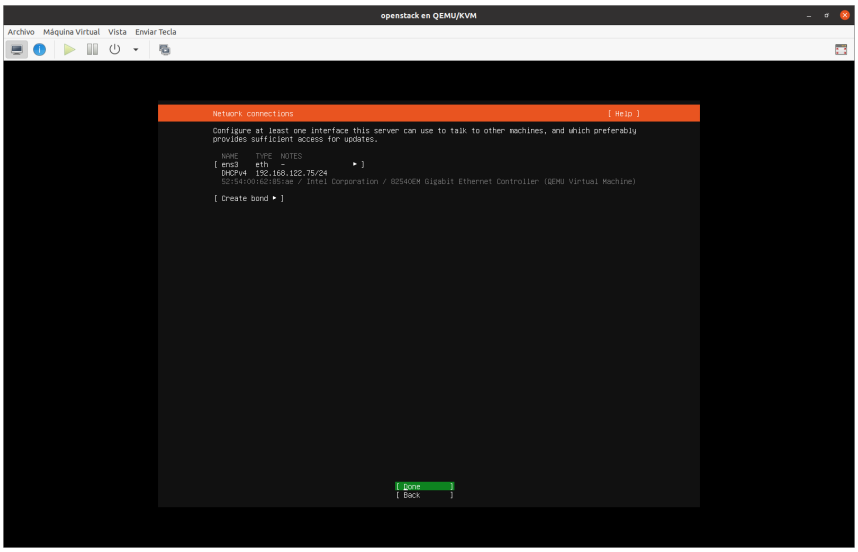

Figura 4.2: IP de la máquina virtual Microstack.

Una vez se ha completado, procederemos a entrar en modo de super-usuario y a instalar ciertos paquetes necesarios para el correcto funcionamiento del sistema.

```
1 apt install net-tools
2 apt install mlocate traceroute git python3 python3-pip wiresharck xterm
    -y
```

A continuación procederemos a instalar microstack en el sistema mediante con el siguiente comando:

```
1 snap install microstack
```

Al terminar el proceso de instalación comprobamos si se ha instalado correctamente, hecho que podemos observar en la figura 4.3, y luego lanzaremos el proceso de inicio automático de microstack. Este proceso puede tardar unos minutos en terminarse, hecho que observamos en la figura 4.4.

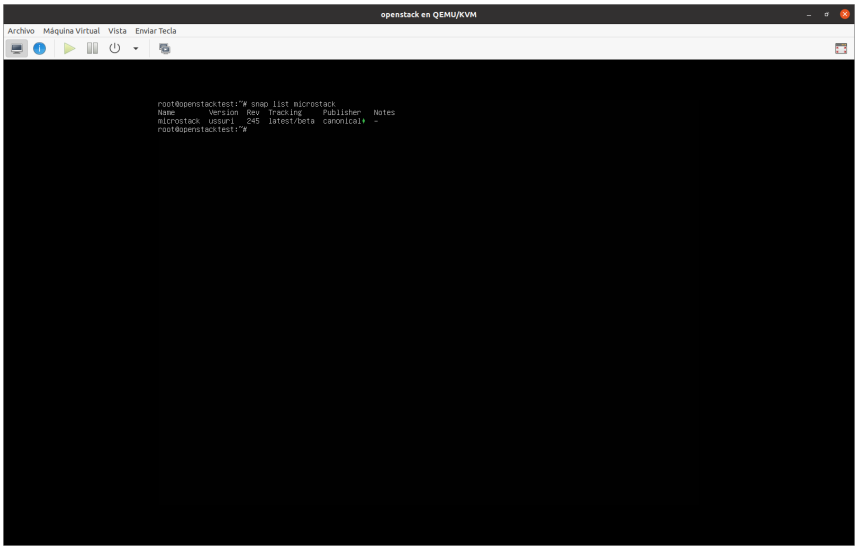

Figura 4.3: Comprobación de Microstack.

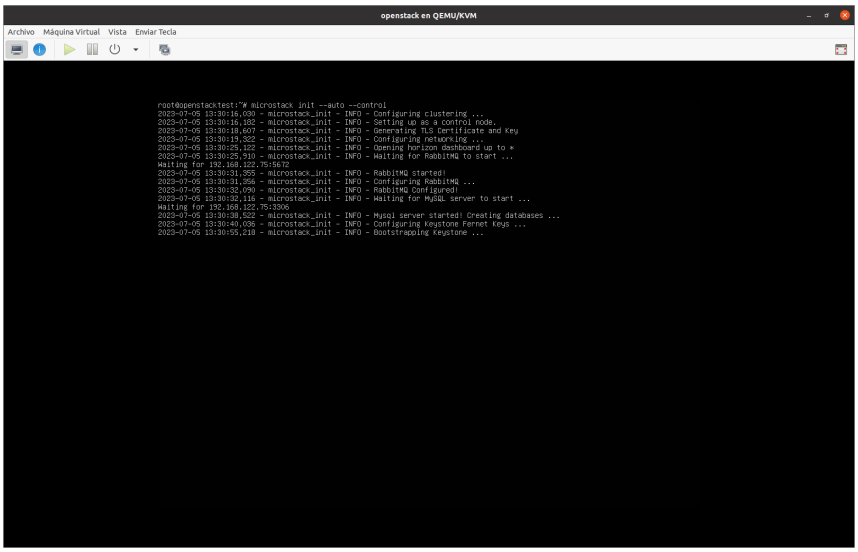

Figura 4.4: Iniciando Microstack.

Una vez terminado el inicio del sistema, ya tenemos el sistema funcionando en pleno rendimiento, pero para poder realizar cualquier modificación debemos de encontrar nuestras credenciales; para ello ejecutaremos el siguiente comando en modo superusuario

```
1    snap get microstack config.credentials
```

Esto nos devolverá todas las credenciales del sistema, pero nosotros vamos a especificar algo más para que nos devuelva únicamente la contraseña de administrador para acceder al panel principal, añadiendo al comando anterior ".keystone-password". Esto nos dará como resultado algo parecido a la figura 4.5.

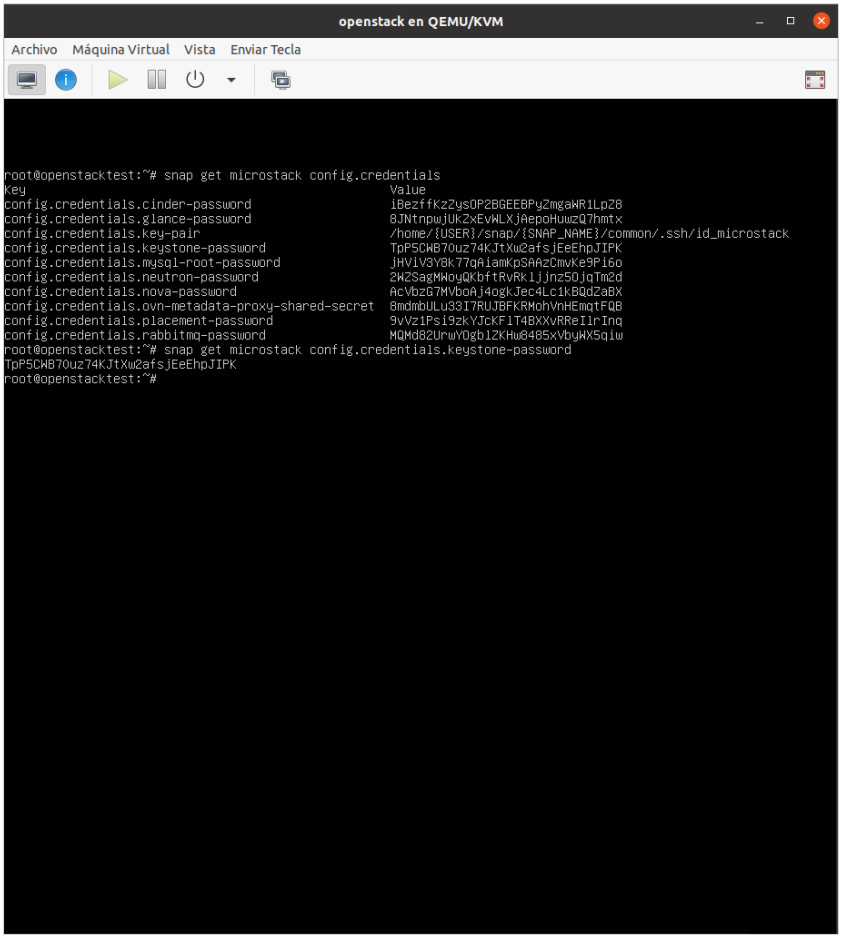

Figura 4.5: Credenciales y contraseñas Microstack.

Una vez obtenidas las credenciales podemos entrar en nuestro navegador y colocar la dirección IP que hemos obtenido al principio de la instalación. Esto nos mostrará un panel como el de la figura 4.6, donde una vez ingresemos nuestro usuario y contraseña obtenida en el apartado anterior accederemos al panel de control de Microstack 4.7.

Figura 4.6: Login de Microstack.

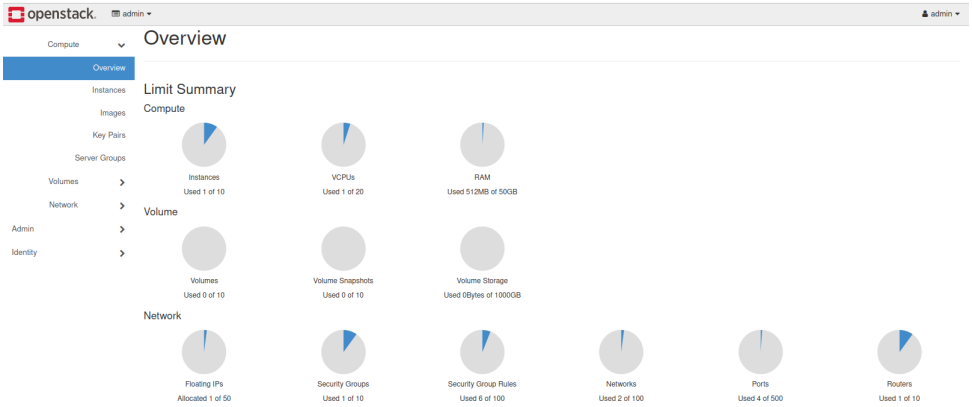

Figura 4.7: Panel de control de Microstack.

CAPÍTULO 5
Evaluación de Microstack

En este apartado vamos a realizar múltiples pruebas sobre Microstack, para poder evaluar la plataforma, entender mejor su funcionamiento, y evaluar su potencial. Estas pruebas son similares a las que se podrían realizar con la estructura de OSM y OpenStack en un servidor, o en un caso en el que se necesite mayor potencia, pero, como ya hemos comentado, Microstack está pensado como entorno de pruebas.

5.1 Pruebas de funcionamiento

5.1.1. Prueba #1: creación del servidor de prueba

Nuestra primera prueba va a constar en la creación de una primera instancia de prueba muy sencilla, conectarnos a ella y probar múltiples funciones como pararla, reiniciarla o eliminarla. Esta prueba se suele llevar a cabo una vez terminada la instalación del programa para poder comprobar si la instalación ha sido correcta y funciona todo a la perfección, para ello vamos a crear una máquina llamada test con una imagen del sistema operativo Cirros (sistema por defecto en el programa), que es una versión minimalista de Linux.

Para crear esta instancia, ejecutaremos dentro de nuestra máquina virtual desde modo superusuario el siguiente comando:

```
1    microstack launch cirros --name test
```

Una vez ejecutado el comando, nos aparecerá algo parecido a la imagen 5.1, la cual indica que la instancia se ha creado correctamente, y nos indica la forma que tenemos para conectarnos a ella mediante el uso de SSH.

```
root@openstack2:~# microstack launch cirros --name test
Creating local "microstack" ssh key at /home/root/snap/microstack/common/.ssh/id_microstack
Launching server ...
Allocating floating ip ...
Server test launched! (status is BUILD)

Access it with `ssh -i /home/root/snap/microstack/common/.ssh/id_microstack cirros@10.20.20.46`
You can also visit the OpenStack dashboard at https://10.20.20.1:443
root@openstack2:~# _
```

Figura 5.1: Primera instancia en Microstack.

Una vez vemos que el proceso de crear la máquina ha terminado, vamos a tratar de conectarnos al mismo utilizando el comando SSH que nos ha indicado anteriormente. En todos los casos, al tratar de conectarnos, nos pedirá la contraseña. En caso de que no

especifiquemos ninguna, se generara automáticamente. Esta contraseña la podemos ver desde nuestra interfaz web más fácilmente, siendo en este caso "gocubsgo". Una vez indiquemos la contraseña entraremos en nuestra nueva máquina, lo cual se puede observar en la figura 5.2.

```
root@openstack2:~# microstack launch cirros --name test
Creating local "microstack" ssh key at /home/root/snap/microstack/common/.ssh/id_microstack
Launching server ...
Allocating floating ip ...
Server test launched! (status is BUILD)

Access it with `ssh -i /home/root/snap/microstack/common/.ssh/id_microstack cirros@10.20.20.46`
You can also visit the OpenStack dashboard at https://10.20.20.1:443
root@openstack2:~# _
```

Figura 5.2: Entrando en la primera instancia.

Una vez entramos dentro, podemos ejecutar múltiples comandos para ver ciertos parámetros de la máquina, como üptime", que nos devuelve el tiempo que lleva en ejecución la máquina, "whoami", que nos da el tipo de sistema operativo, o üname -a"que nos indica como Cirros se derivó de alguna rama de Ubuntu; todos estos comando y algunos más los podemos ver en la figura 5.3.

```
$ uptime
 10:30:08 up 19 min,  1 users,  load average: 0.00, 0.00, 0.00
$
$
$ whoami
cirros
$
$
$ id
uid=1000(cirros) gid=1000(cirros) groups=1000(cirros)
$
$
$ uname -a
Linux test 4.4.0-28-generic #47-Ubuntu SMP Fri Jun 24 10:09:13 UTC 2016 x86_64 GNU/Linux
$
$
$
$ cat /etc/os-release
NAME=Buildroot
VERSION=2015.05-g31af4e3-dirty
ID=buildroot
VERSION_ID=2015.05
PRETTY_NAME="Buildroot 2015.05"
$
```

Figura 5.3: Datos sobre la primera instancia.

Después de realizar esta comprobación podemos verificar si esta instancia aparece como activa o no en las listas de nuestra aplicación; para hacer esa prueba podemos hacerlo tanto desde la terminal, como desde la parte visual de la aplicación:

- En caso de utilizar el terminal, debemos de ingresar el comando que nos aparece en la figura 5.4 que, como podemos observar, nos muestra una lista con los servidores y algo de información de ellos, como si está activo o inactivo, su nombre, etc.

```
root@openstack2:~# microstack.openstack server list -f json
[
  {
    "ID": "66cc438d-29fd-448d-820b-a8218dce24b1",
    "Name": "test",
    "Status": "ACTIVE",
    "Networks": "test=192.168.222.62, 10.20.20.46",
    "Image": "cirros",
    "Flavor": "m1.tiny"
  }
]
```

Figura 5.4: Lista de instancias.

- En caso de querer utilizar la interfaz gráfica, podemos observar desde el panel principal como tenemos una instancia activa, la cantidad de VCPUs que está utilizando, o la memoria que está gastando.

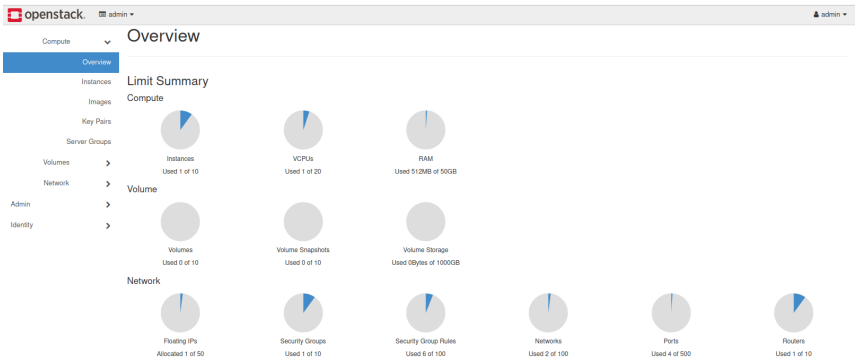

Figura 5.5: Panel principal desde el navegador.

Además de todo esto, mediante los siguientes comandos podemos parar, reiniciar o eliminar la instancia de prueba creada, colocando el ID que hemos visto antes en la figura 5.4 para saber a qué máquina nos referimos.

```
1  microstack.openstack server start 66cc438d-29fd-820b-a8218dce24bq
2  microstack.openstack server stop 66cc438d-29fd-820b-a8218dce24bq
3  microstack.openstack server delete 66cc438d-29fd-820b-a8218dce24bq
```

Todos estos comandos también se pueden realizar desde la interfaz web pulsando múltiples iconos.

5.1.2. Prueba #2: gestión de Flavors

Nuestra segunda prueba va a consistir en el estudio de los "Flavors", una parte muy importante dentro de toda esta estructura.

Los "Flavors"definen la capacidad de cómputo, memoria y almacenamiento de un servidor virtual o instancia. En pocas palabras, es la configuración de hardware disponible para un servidor, siendo estos análogos a los tipos de instancia de EC2 de Amazon al definir las cantidades de varios recursos de hardware (cómputo, red, almacenamiento) asignados a una instancia. Esto permite aprovecharemos para ver cómo crearlas y borrarlas, así como las opciones por defecto que nos genera MicroStack

En primer lugar, vamos a observar y conocer todos los flavors por defecto que tiene Microstack. Para ello ejecutaremos en nuestra máquina virtual el comando:

```
1  microstack.openstack flavor list -f json
```

Este comando nos mostrará la lista de Flavors que tiene nuestra máquina, donde nos indica el nombre de cada uno, la cantidad de RAM, disco, hilos de CPU, etc. Al ejecutar esto en nuestra máquina el resultado es el que podemos observar en la figura 5.6.

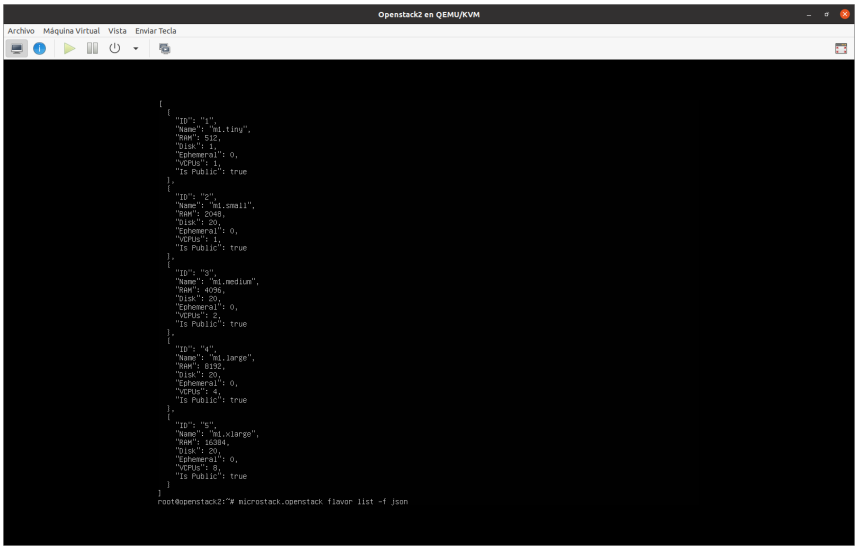

Figura 5.6: Lista de Flavors.

Aparte de la lista, también podemos obtener información detallada de alguna de ellas escribiendo el comando que aparece en la figura 5.7. Esta información puede ser interesante antes de crear alguna máquina para conocer todas las propiedades del sistema.

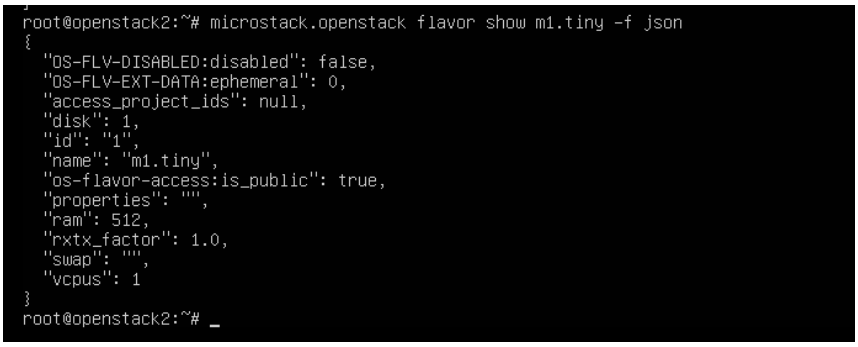

Figura 5.7: Información detallada de Flavors.

Una vez vista la información general, vamos a crear nuestro propio flavor para luego utilizarlo en una futura instancia que creemos; a este flavor le vamos a llamar m1.web para mantener la nomenclatura que se suele utilizar, y vamos a colocarle con 1 CPU, 1 GB de RAM y 20 GB de disco duro. Para ello ejecutaremos el comando que se puede ver en la parte superior de la figura 5.8 el cual, al terminar, nos mostrará los detalles del flavor creado.

Figura 5.8: Creación de flavor personalizado.

Por último, podemos probar a repetir nuestra primera prueba, solo que ahora vamos a cambiar de m1.tiny (flavor por defecto) a m1.web. Si hemos realizado todo correctamente, nos debería de crear un sistema perfectamente igual al anterior, pero con las nuevas características; esto lo podemos observar en la figura 5.9.

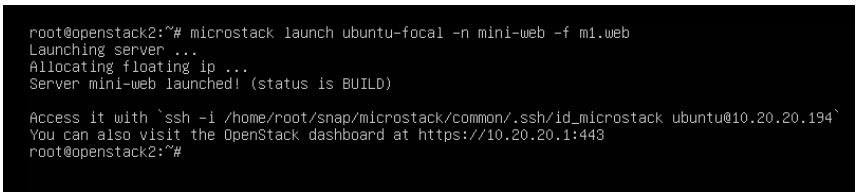

Figura 5.9: Creando máquina con el nuevo flavor.

5.1.3. Prueba #3: creación y uso de imágenes

Dentro de nuestra estructura, las imágenes son archivos que contienen un disco virtual de un sistema operativo, y la configuración mínima para poder operar dentro de una nube. Estas imágenes pueden ser genéricas de sistemas operativos comunes, o algunas más sencillas, como el ejemplo de Cirros que tenemos por defecto en el sistema.

En esta prueba vamos a ver el funcionamiento de las imágenes, instalar nuevas imágenes, y comprobar su funcionamiento en nuestras instancias.

En primer lugar, como sucedía con las flavors, podemos usar el comando list o show para mostrar una lista o información más detallada de los sistemas operativos. En este caso podemos ver en la figura 5.10 el uso del comando show sobre cirros. En esta imagen podemos ver múltiples datos, como el formato de disco que tiene, donde está guardado, su dueño, o múltiples propiedades sobre el sistema.

Figura 5.10: Información sobre cirros.

En segundo lugar, vamos a crear una nueva imagen con la idea de que este servidor sea el host de una futura web. Para ello descargaremos el sistema operativo Ubuntu 20.04 LTS cloud, usando el siguiente comando.

```
1 $ wget https://cloud-images.ubuntu.com/focal/current/focal-server-cloudimg-
    amd64-disk-kvm.img
```

Una vez se termina de descargar el sistema podemos proceder a crear la imagen como se ve en la figura 5.11, pero antes de crearla debemos de especificar múltiples parámetros, como el formato de disco, si es público o no, o donde se encuentra el archivo

Figura 5.11: Creando la nueva imagen.

Para comprobar que el sistema está instalado y funciona a la perfección, la mejor manera es crear una instancia utilizando este sistema operativo; para esta instancia vamos a utilizar tanto la flavor creada en el apartado anterior como el nuevo sistema, para ello podemos utilizar el siguiente comando:

```
1 microstack launch ubuntu-focal -n ubuntu-focal -f m1.web
```

Cuando termine de crearse la máquina, podemos conectarnos a la misma siguiendo los mismos pasos realizados en la primera prueba; podemos observar cómo funciona el sistema perfectamente (ver Figura 5.12).

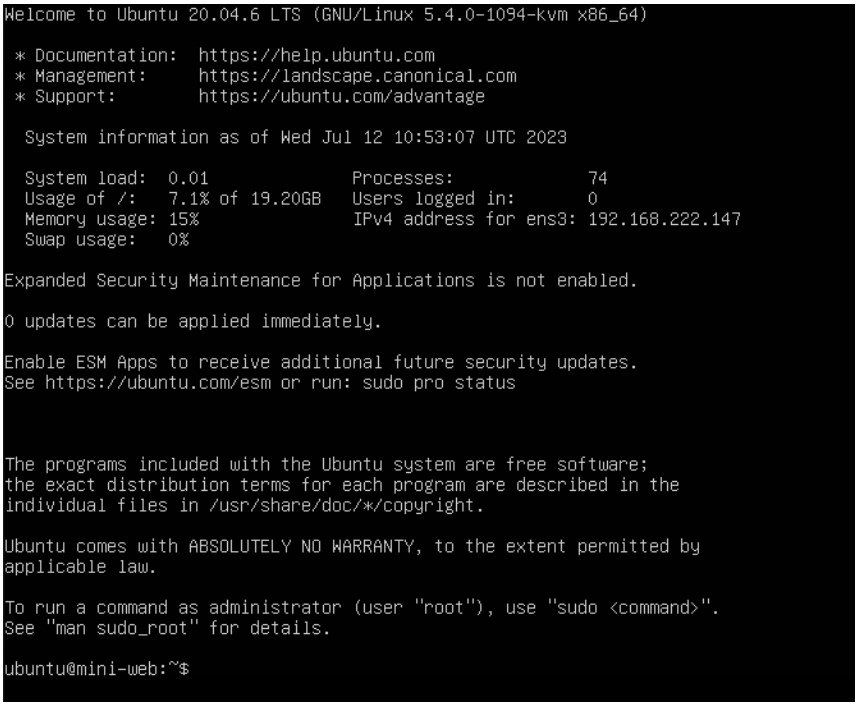

Figura 5.12: Instancia con imagen Ubuntu.

También podemos observar esto dentro de nuestra interfaz web: si entramos en las instancias activas, podemos ver como tenemos una nueva instancia con imagen Übuntu-focalçon flavor "m1.web", las cuales acaban de ser creadas (ver Figura 5.13).

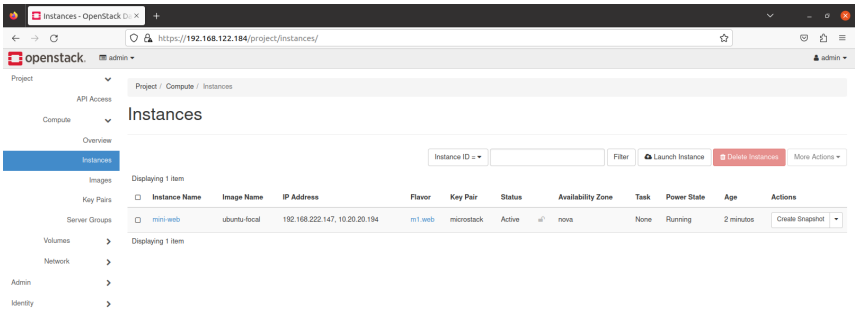

Figura 5.13: Instancia con imagen Ubuntu en interfaz web.

5.1.4. Prueba #4: uso de redes y subredes

Las redes dentro de Microstack representan las redes virtuales que dan conectividad a nuestras máquinas virtuales dentro de la estructura. Estas redes pueden ser tanto públicas como privadas. Por otro lado, las subredes son segmentos de estas redes que agrupan un rango de direcciones IP. Estas permiten dividir una red en segmentos más pequeños para mejorar la administración de la red, máscara o rango; se suelen utilizar para mejorar la administración de la red, optimizar el tráfico o mejorar la seguridad.

En esta prueba vamos a crear múltiples redes y subredes, estudiar su funcionamiento, y ver algunas funciones que nos ofrece Microstack. Para ello comenzaremos creando una nueva red o network, a la cual llamaremos çomplex-netza que vamos a realizarle múltiples cambios para hacerla algo más completa que la predeterminada de Microstack. Para ello ejecutaremos el comando que vemos en la figura 5.14, que se encargará de generar la nueva red.

```
root@openstack2:~# microstack.openstack network create complex-net
```

Figura 5.14: Creando network complex-net.

Una vez generada la nueva red, vamos a crearle dos subnet, las cuales vamos a llamar "subnetprueba1z "subnetprueba2". Estas subnets van a tener diferentes rangos de direcciones IP, la primera cubrirá el rango de 192.168.1.0/24 y la segunda 192.168.2.0/24, para así estudiar las subnet en varias direcciones IP. Para llevar a cabo este proceso podemos hacerlo usando el siguiente comando:

```
1    microstack.openstack subnet create --network complex-net --subnet-range
         192.168.1.0/24 subnetprueba1
2 microstack.openstack subnet create --network complex-net --subnet-range
    192.168.2.0/24 subnetprueba2
```

Después de crear nuestras subredes, debemos crear un router, ya que sin él no tendríamos comunicación entre nuestras dos subredes creadas; para ello en primer lugar vamos a crear un router y le llamaremos router1üsando el comando que aparece en la

parte superior de la figura 5.15. Como podemos observar, una vez creado nos aparece más información sobre el router, como la fecha de creación, el estatus, si es Gateway, etc.

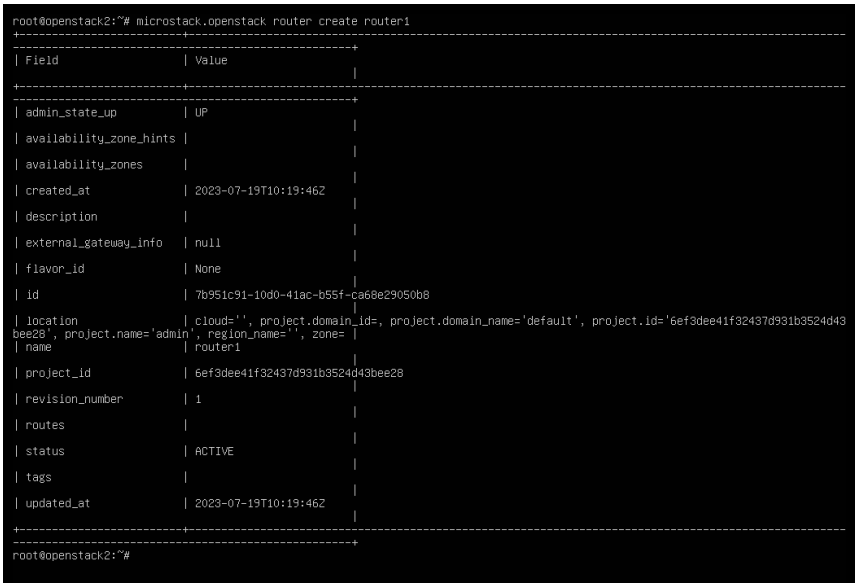

Figura 5.15: Creando router1.

Una vez hemos creado el router, podemos unir las dos subnets a nuestro router; para este paso debemos ejecutar el siguiente comando:

```
1
2 microstack.openstack router add subnet router1 subnetprueba1
3 microstack.openstack router add subnet router1 subnetprueba2
```

Una vez hecho, ya tendremos toda nuestra red conectada y si tratamos de observarlo desde la interfaz web, podemos observar como todo está creado correctamente: como se ha creado la red çomplex-netçon sus dos subredes, y estas dos subredes están conectadas por el “router1".

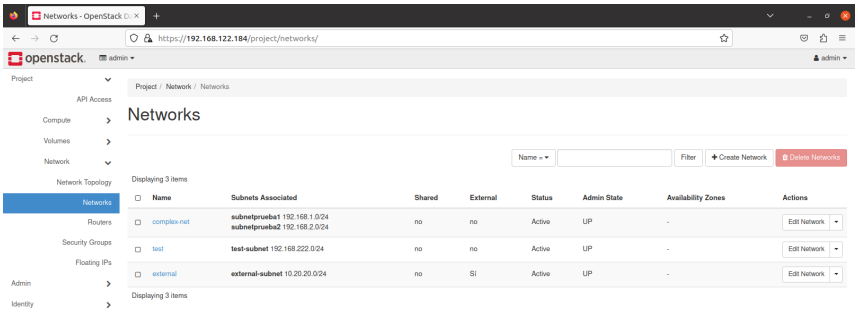

Figura 5.16: Comprobación de la red en la interfaz.

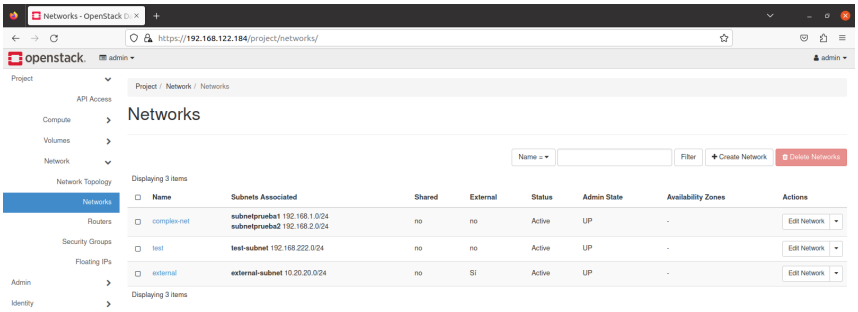

Figura 5.17: Comprobación de la unión de las subnet mediante el router.

Por último, y para tener una prueba muy completa del uso de networks en Microstack, vamos a crear una red igual que la de antes, pero esta vez vamos a llamarla .externalnetc", ya que esta red va a ser capaz de comunicarse directamente con el exterior. Para ello ejecutaremos el siguiente comando:

```
1    microstack.openstack network create --external --provider-network-type flat
         --provider-physical-network external-netc
```

Esto lo que hace es crear una red que tenga acceso al exterior, es decir a internet. Así que, al crearla, debemos repetir los mismos pasos que en la instalación anterior, tanto para crear una subnet como para crear un router, para poder así conectar una máquina a esta red y probar si de verdad conecta con el exterior. Una vez repetido todo el proceso, procedemos a crear una nueva máquina virtual sencilla utilizando el comando que se ve en la figura 5.18. Una vez se termine de crear nos conectamos a ella y probamos a hacer ping a 8.8.8.8 (servidores DNS de Google). Como se observa en la Figura 5.19, funciona a la perfección

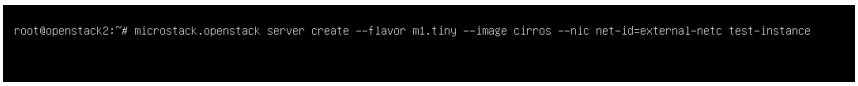

Figura 5.18: Creando instancia con acceso a internet.

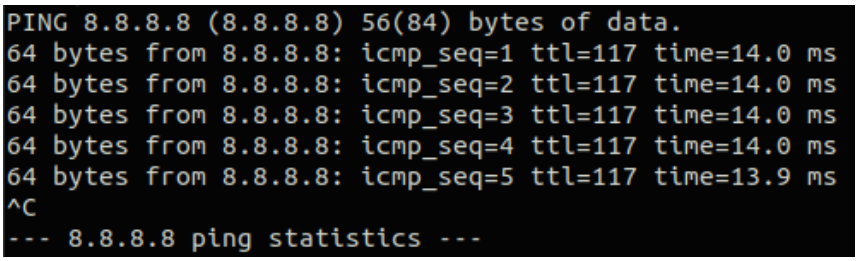

Figura 5.19: Ping desde la nueva instancia.

5.1.5. Prueba #5: grupos de seguridad

Los grupos de seguridad dentro de Microstack son conjuntos de reglas que definen el acceso a las redes para las distintas instancias de un proyecto. Estas reglas son específicas de cada proyecto, y pueden ser editadas por los miembros de este. Estos grupos son utilizados para aplicar reglas de IP, creando nuevos grupos con las reglas deseadas, o modificando el conjunto de estas. Cada proyecto o grupo de pruebas tiene un grupo de seguridad predeterminado, y todas sus instancias se incluyen en este grupo. Actualmente, los grupos predeterminados permiten todo el tráfico saliente, pero rechazan todo

el tráfico entrante a las instancias; por ello se pueden cambiar las reglas de filtrado de IP para las instancias del proyecto, permitiendo crear un nuevo grupo de seguridad, o modificar las reglas definidas en el grupo.

En esta prueba vamos a crear nuevos grupos de seguridad y realizarles pequeñas modificaciones. En primer lugar crearemos el nuevo grupo de seguridad; para ello usaremos el comando que se ve en la parte superior de la figura 5.20. Al igual que con la creación del router, nos aparece algo de información extra sobre el grupo que estamos creando, como la id del grupo, la descripción, y algunas de las normas predeterminadas, entre otros.

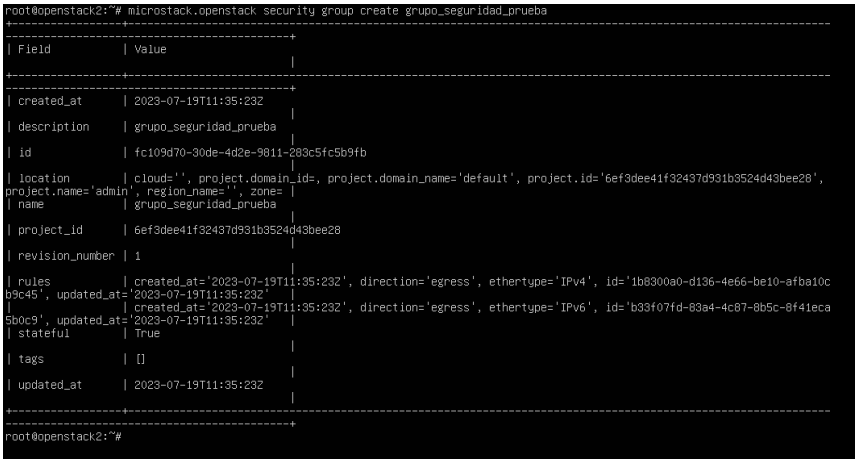

Figura 5.20: Creación de grupo de seguridad.

En este grupo de seguridad vamos a definir 3 normas que tocarán los pilares principales de nuestro estudio:

1. La primera norma que vamos a añadir va a ser permitir el tráfico SSH entrante de cualquier IP, siempre y cuando entre por el puerto 22. Para ello escribiremos el siguiente comando en el terminal:

```
microstack.openstack security group rule create --proto tcp --dst-port 22:22 --remote-ip 0.0.0.0/0 grupo_seguridad_prueba
```

2. La segunda norma que vamos a crear va a ser permitir todo el tráfico HTTP entrante por el puerto 80 entre todas las direcciones IP; para ello escribiremos el siguiente comando en el terminal:

```
microstack.openstack security group rule create --proto tcp --dst-port 80:80 --remote-ip 0.0.0.0/0 grupo_seguridad_prueba
```

3. Por último vamos a permitir todo el tráfico entrante desde una IP específica, permitiendo todo el tráfico desde la IP de una de nuestras máquinas (192.168.222.236/32):

```
microstack.openstack security group rule create --proto tcp --remote-ip 192.168.222.236/32 grupo_seguridad_prueba
```

Todas estas normas que hemos creado, podemos verlas desde la interfaz web, y comprobar que todo se ha creado correctamente; además, también podemos crear cualesquier normas de una manera más sencilla y visual desde la interfaz web. Esto lo vemos tanto en la figura 5.21 como en la figura 5.22.

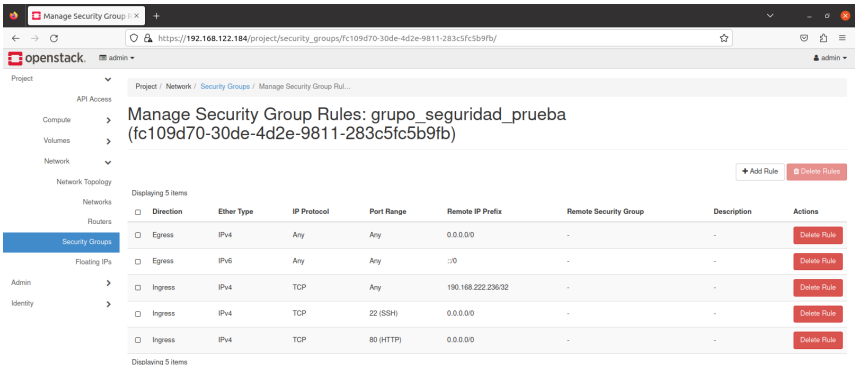

Figura 5.21: Interfaz con las normas del grupo de seguridad.

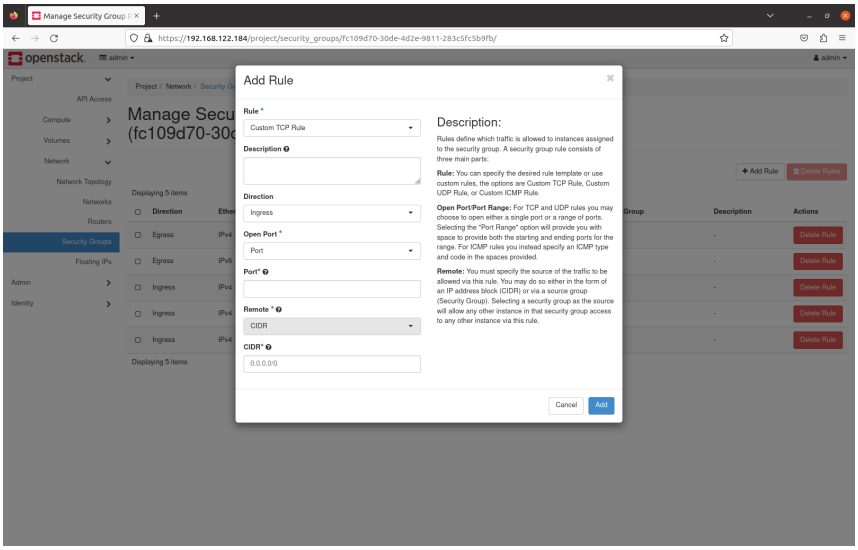

Figura 5.22: Añadiendo normas desde la interfaz.

Por último, igual que en el resto de los casos, podemos crear un servidor para que utilice este grupo de seguridad, o desde el propio panel podemos agregárselo a alguno ya existente.

5.1.6. Prueba #6: Máquinas Virtuales

En esta prueba vamos a analizar y comprobar el nivel más importante que tiene esta estructura: el uso de máquinas virtuales como hosts de sistemas, ya sea base de datos, webs o aplicaciones.

En primer lugar, vamos a crear una instancia con el flavor y la imagen Ubuntu que hemos creado en las pruebas 2 y 3, usando el comando:

```
1  microstack.openstack launch ubuntu-focal -n mini-server -f m1.web --nic net
       -id=external-netc
```

Como podemos ver en el comando, hemos usado tanto el sistema operativo usado, la nueva flavor, y la red con acceso externo; por otro lado, no usaremos los nuevos grupos de seguridad para evitar posibles limitaciones de las normas creadas.

Una vez creado, entramos dentro de nuestro sistema, como en pruebas anteriores, y descargaremos varios paquetes dentro del sistema. En este caso vamos a descargar

una página web de pruebas. Para ello descargaremos una aplicación de un repositorio personal usando los siguientes comandos:

```
    $ sudo add-apt-repository ppa:wester1235/app-dcu/src
$ sudo apt update
$ sudo apt install -y wester1235-com
$ sudo systemctl enable --now  wester1235-com.service
```

El uso de estos comandos hace que se instale a nivel local nuestra "aplicación", y desde nuestro sistema podríamos comprobar si está funcionando desde "localhost"; aunque no se pueda visualizar de manera visual, al ejecutar el siguiente comando nos aparecerá algo como lo siguiente, que indica que la aplicación creada en React está en funcionamiento.

```
    $ wget -qO - https://localhost --no-check-certificate
            import React from 'react';
            import { createRoot } from 'react-dom/client';
            import App from './App';
            import * as serviceWorkerRegistration from './
                serviceWorkerRegistration';
            import reportWebVitals from './reportWebVitals';

            const container = document.getElementById('root');
            const root = createRoot(container!);
            root.render(
              <React.StrictMode>
                <App />
              </React.StrictMode>
            );
```

Como podemos ver, esto se puede llegar a realizar con cualquier tipo de sistema, es decir, puedes hostear cualquier tipo de sistema, incluso en múltiples máquinas virtuales, para desplegar pequeños sistemas o aplicaciones no muy complejas. Además, el tiempo de despliegue es muy rápido, no llegando a tardar ni un par de segundos; concretamente, desplegar este sistema y lanzarlo le ha costado aproximadamente 3 segundos, aunque esto puede variar mucho dependiendo de las características hardware del sistema utilizado.

5.2 Interfaz de MicroStack

Como ya comentamos antes, MicroStack tiene una interfaz gráfica desde la cual puedes realizar todas las pruebas que hemos hecho anteriormente mediante esta, pero además de esto nos permite el acceso a varias zonas de gestión.

En este apartado vamos ha ver alguna de las partes que tiene esta interfaz, concretamente explicaremos para que sirve tanto la zona de identidad como el panel de administrador.

5.2.1. Panel de administrador

Dentro de MicroStack, si somos administradores nos aparecerá una opción debajo de proyecto llamada .Admin". Dentro de esta sección podemos encontrar mucha información extra sobre ciertos sectores, y alguna parte necesaria dentro de la estructura.

Un ejemplo de esto sería la zona de cómputo, en la cual, como podemos ver en la figura 5.23, nos aparece mucha más información de los Hypervisores, como la información

de que está activo en este momento, la cantidad de hilos de ejecución que están funcionando en dicho instante, la cantidad de RAM que está consumiendo el sistema, y la que tiene disponible, entre otros.

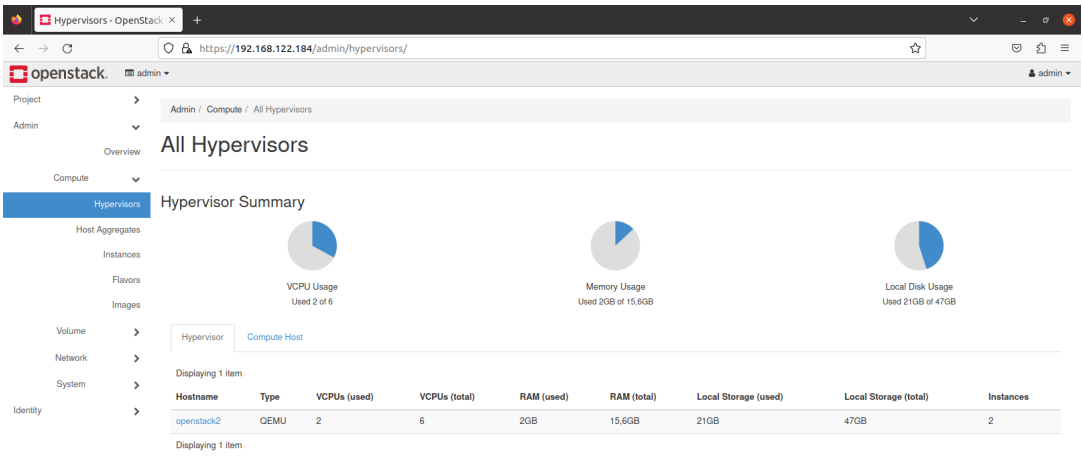

Figura 5.23: Panel de admin de la zona de cómputo.

Otra de las zonas muy importantes dentro del panel de administrador es la zona de sistema, donde podemos obtener toda la información del sistema. Como ejemplo, en la figura 5.24 podemos ver todo los servicios que se están ejecutando, y sus Endpoints. Además, podemos ver todos los servicios que se han nombrado anteriormente.

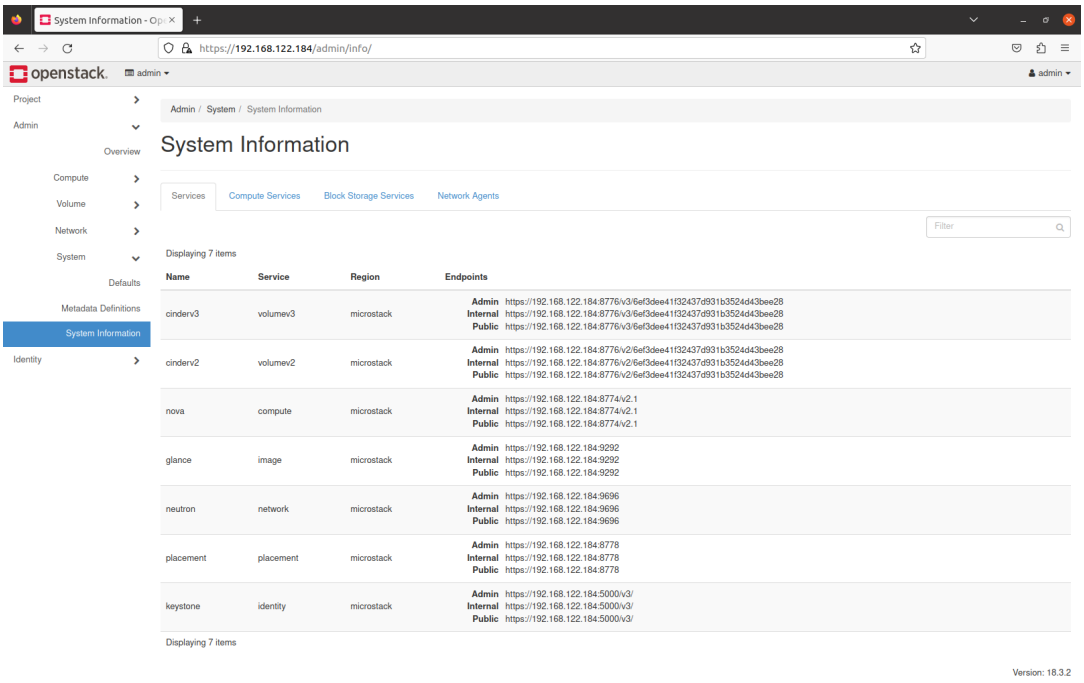

Figura 5.24: Datos generales del sistema.

Otra información que podemos ver en esta pestaña es información detallada sobre cada servicio.

En la zona de servicio de cómputo, que vemos en la figura 5.25, se detallan los servicios de cómputo activados, el estado, o la última vez que el mismo se ha comprobado. Esta información es muy importante para un administrador para poder monitorizar el

estado del sistema, y así comprobar que todo esté funcionando a la perfección. Lo mismo sucede con en las figuras 5.26 y 5.27

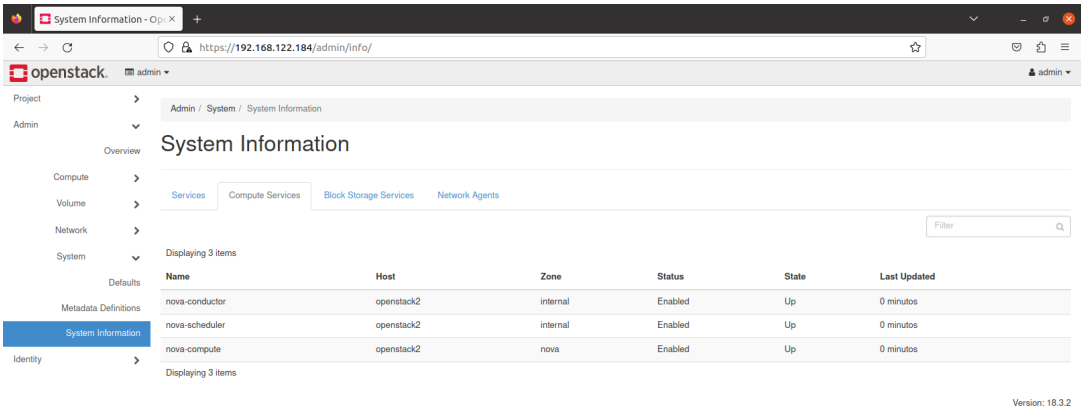

Figura 5.25: Datos del servicio de cómputo.

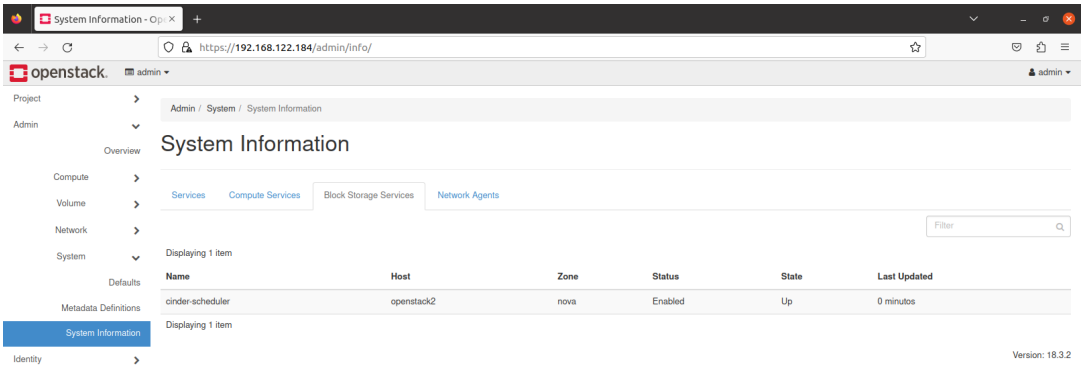

Figura 5.26: Datos del servicio de almacenamiento.

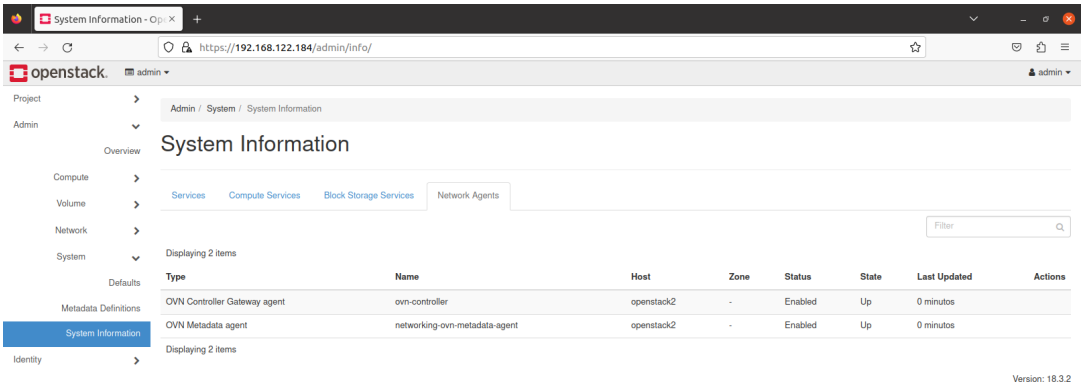

Figura 5.27: Datos del servicio de redes.

5.2.2. Zona de identidad

Dentro de nuestra interfaz podemos observar otra de las partes muy importantes que es la zona de identidad, donde podemos ver varios apartados.

Uno de estos apartados es la zona de usuarios, donde podemos dar acceso a múltiples usuarios a un proyecto concreto con una cantidad de permisos limitados.

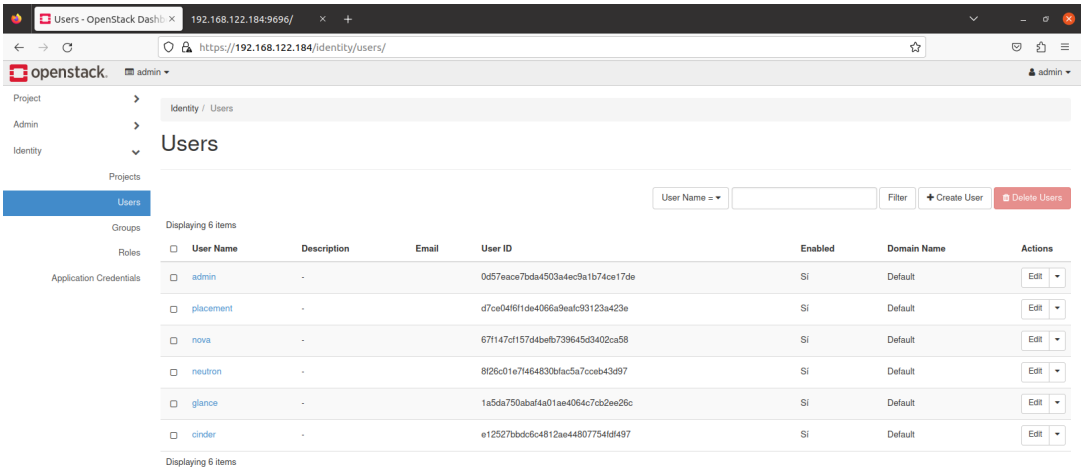

Figura 5.28: Zona de usuarios.

Otra zona que podemos observar es la zona de grupos, donde podemos crear grupos para asignarles roles a estos. El crear grupos es muy interesante para nosotros, ya que, podemos unificar roles por grupos de personas, y así evitar tener permisos por usuarios, además de que el uso de grupos permite realizar una gestión más sencilla de los mismos. El proceso de creación o de añadir nuevos elementos se puede ver en las figuras 5.29 y 5.30.

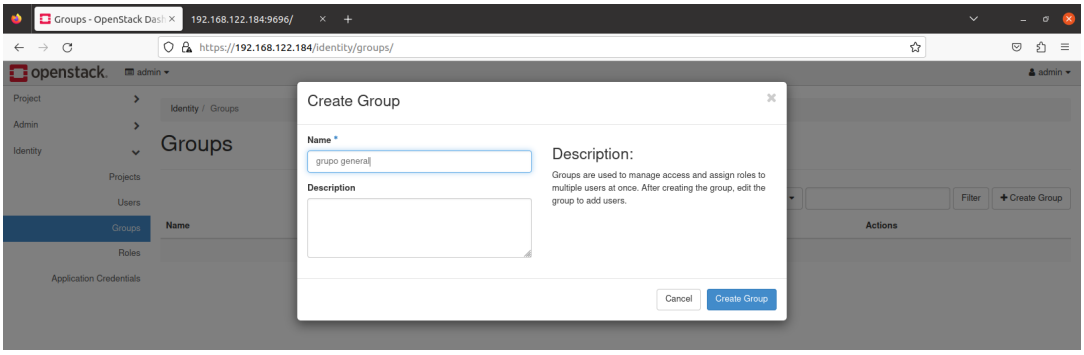

Figura 5.29: Creando grupos.

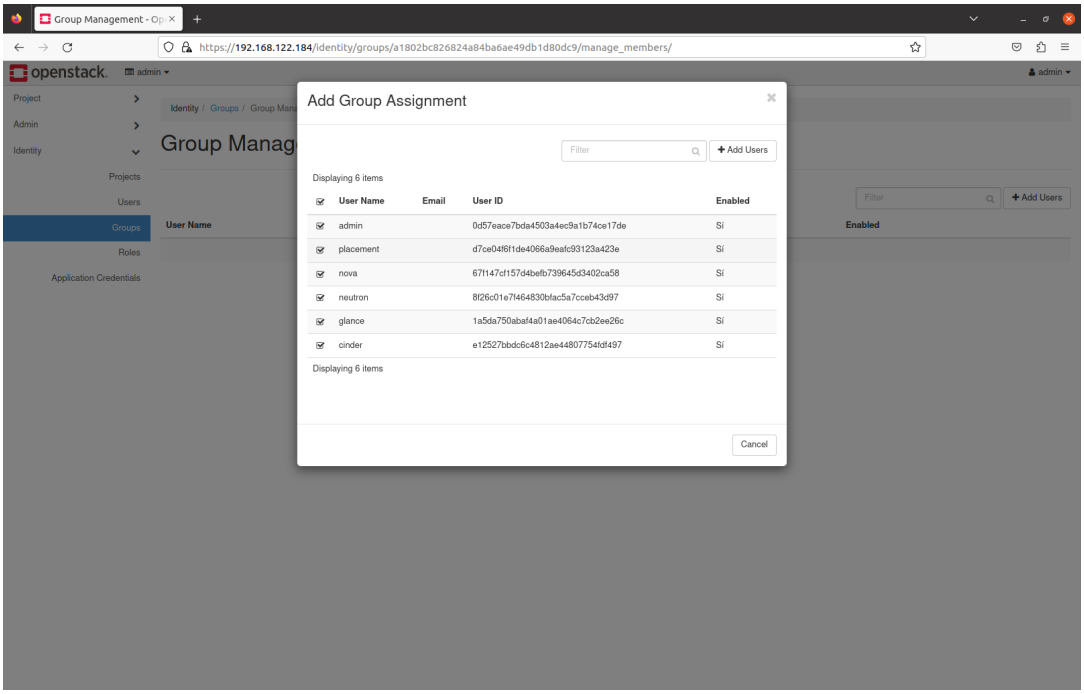

Figura 5.30: Uniendo usuarios a los grupos.

Por último, temeos la zona de roles, zona muy importante a la hora de gestionar usuarios, ya que estos roles limitan el uso de nuestra aplicación a cada usuario o grupo, dependiendo de los permisos que se les dé. En primer lugar aparecen 3 roles: el de administrador, el de miembro, y el de lector. El permiso de miembro te permite ver únicamente la zona de proyecto, y trabajar sobre un proyecto asignado, mientras que la zona de lector únicamente te permite ver el proyecto y monitorizarlo, sin posibilidad de hacer ningún cambio sobre el mismo. También puedes crear y modificar cualquier permiso con las normas que se consideren.

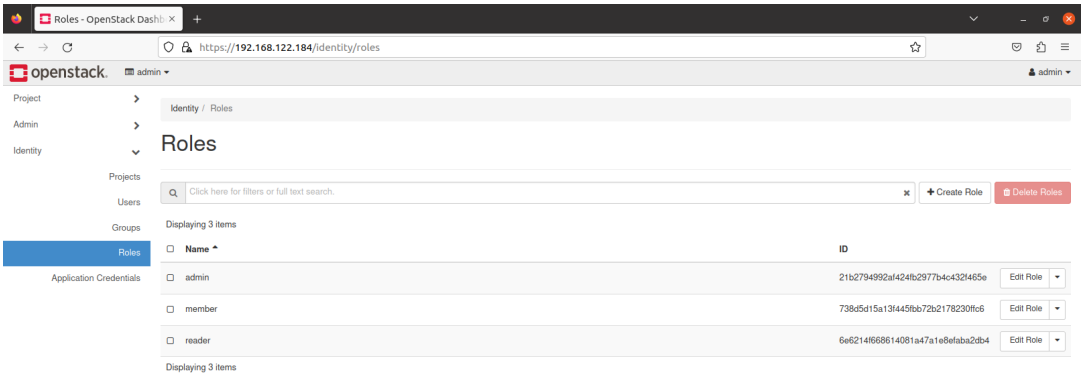

Figura 5.31: Zona de roles.

5.3 Evaluación de resultados

En este apartado vamos a realizar la evaluación del rendimiento de este sistema a la hora de realizar el despliegue de aplicaciones. Aunque estos resultados no son generalizables, ya que dependen totalmente de las características del equipo, los detallamos para una configuración en concreto, que es la siguiente:

- Procesador: AMD Ryzen 7 5700G 4.6GHz
- Tarjeta Gráfica: Modelo de gráficos Radeon™ Graphics
- Memoria RAM: 32 GB DDR4
- Almacenamiento: Kioxia Exceria G2 Unidad SSD 500GB M.2

Atendiendo a que las características de esta máquina ofrecen un buen rendimiento, todo el despliegue y pruebas ha ido a una velocidad relativamente elevada.

5.3.1. Análisis de rendimiento

Para poder determinar el rendimiento completo de MicroStack, hemos evaluado los tiempos de despliegue de múltiples procesos asociados a la plataforma:

- Instalación de la plataforma.
- Arranque de la plataforma.
- Despliegue de instancias.

Es muy importante destacar que los aspectos que se han medido no son los únicos que se podrían evaluar, ya que, existen varios que podrían analizarse para determinar el rendimiento real de MicroStack. Se podría, por ejemplo, medir lo que se tarda en crear nuevos Flavors, descargar nuevos sistemas, o actualizar permisos.

Instalación de la plataforma

Como ya hemos visto en el proceso de instalación de MicroStack, este es un proceso relativamente sencillo, aunque lo más complejo es todo el tiempo que tarda en descargarse y estar listo para iniciarse. Es muy importante que esto funcione a la perfección, ya que, si hay algún problema durante la instalación, MicroStack puede no funcionar como debería, o generará múltiples problemas, como falta de funciones, perdida de conexión, etc.

El tiempo de instalación ha sido de unos 15 minutos aproximadamente; si investigamos un poco sobre el tiempo de instalación en otros casos, podemos ver como la media son aproximadamente 20 minutos para instalarlo y dejarlo perfecto para su funcionamiento. Esto nos deja que, en nuestro caso, hemos recortado casi un 20 por ciento del tiempo medio gracias a las características de nuestro equipo.

Arranque de la plataforma

Una vez hemos instalado la plataforma, tenemos que colocar un comando para iniciar la plataforma y comenzar a utilizarla. Este comando solo se debe de utilizar una única vez, y luego una vez iniciada la máquina virtual, se inicia directamente el sistema. De esta manera, vamos a medir dos cosas: en primer lugar, el tiempo de arranque inicial, y el tiempo de arranque al volver a encender el sistema.

La primera vez que iniciamos el sistema normalmente suele tardar alrededor de un minuto en estar conectado y funcionando, pero en nuestro caso tardó alrededor de unos 45 segundos. Esto nos deja con una mejora de aproximadamente un 20 por ciento de mejora, similar a la que hemos visto en la instalación de la plataforma.

Después una vez ejecutado, se activa el sistema nada más encenderse la máquina virtual, siendo un tiempo mucho menor, normalmente una vez iniciada la máquina virtual, la cual se inicia a los 10/15 segundos de media, pero en nuestro sistema no llegó a tardar ninguna de las veces más de 6 segundos, hecho que nos deja con casi un 50 por ciento de mejora frente a la media.

Despliegue de instancias

Otra de procesos relevantes es el tiempo en el que se tarda en generar una nueva máquina virtual o instancia. Este tiempo es muy importante, ya que, en caso de tener una máquina con mucha carga, podrías clonarla para dividir la carga en dos máquinas, y ganar velocidad y uso de recursos.

En nuestro caso, considerando las pruebas que hemos realizado, ha tardado muy poco en crear una instancia, incluso a la hora de duplicarlo. No obstante, debemos de tener en cuenta que se han usado unas instancias con muy poca RAM y espacio; en caso de necesitar una con más carga tardará más, y el tiempo será más cercano a los medidos.

En nuestro equipo, ha tardado menos de 4 o 5 segundos en crear una nueva instancia, cuando el tiempo medio es de unos 7 o 8 segundos. Esto mantiene el margen de mejora que hemos visto en anteriores apartados, pero no es tan determinante como los anteriores porque depende del tipo de máquinas utilizadas y sus características.

Conclusión

Como podemos ver en la figura 5.32, las características que tiene el sistema son muy importantes, hasta el punto de que en casi todo lo medido hemos sacado un 20 por ciento de mejora respecto a la media.

Estos datos demuestran que con el paso del tiempo estos sistemas serán mucho más interesantes de utilizar y más necesarios en todos los despliegues, demostrando así la importancia de utilizarlos y mejorarlos.

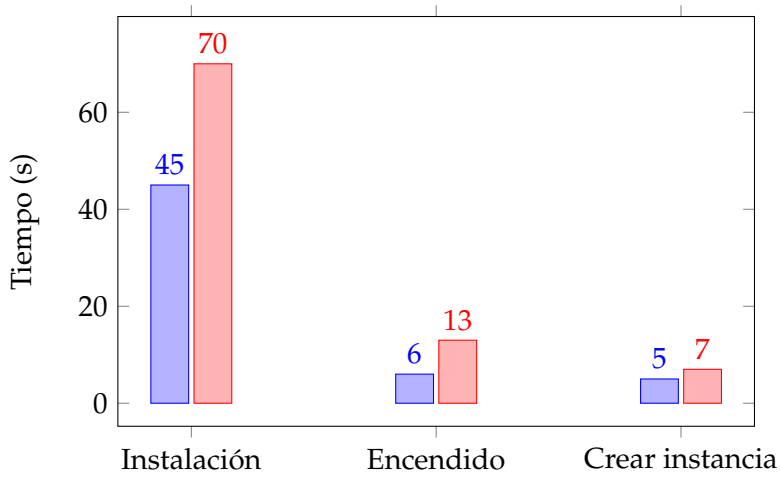

Figura 5.32: Análisis del tiempo medido vs. tiempo medio de ejecución.

CAPÍTULO 6
Análisis socio-económico

En este apartado del TFG vamos a analizar en entorno socio-económico, el cual incluye el presupuesto necesario para elaborar ambas estructuras estudiadas en este trabajo, y un análisis del impacto socio-económico que se podría esperar de la aplicación.

6.1 Descripción del entorno socio-económico

Desde un punto de vista económico, adoptar OpenStack y MicroStack puede resultar en ahorros muy significativos para las empresas, al permitir implementar una infraestructura de nube sin incurrir a costo de licencias de software, al ser OpenSource. Además, al ser código abierto, permite adaptar y personalizarlo para la infraestructura de la empresa a sus necesidades específicas.

Tanto OpenStack como MicroStack tratan de promover la colaboración y el intercambio de conocimiento entre desarrolladores, para que así tanto usuarios como contribuyentes puedan influir en la dirección de la comunidad y del software con encuestas o ideas propias que son escuchadas.

A continuación, vamos a enumerar algunas de las ventajas económicas y ecológicas más importantes que podemos obtener al usar estas estructuras:

1. **Reducir los costes en equipamiento**, al necesitar menos equipo especializado y ser sustituido por servidores genéricos, reduciendo así también la huella de carbono generada.

2. **Se puede compartir la infraestructura**, al ser una de las principales ideas de los creadores del servicio.

3. **Mayor velocidad de despliegue de los nuevos servicios**, como hemos visto ampliar o disminuir la cantidad de recursos o instancias es prácticamente instantánea, con unos pequeños segundos de diferencia.

4. **Capacidad de cambios remotos**, sin necesidad de transporte para actualizar o implantar nuevo software.

5. **Reducción de consumo de energía,** reduciendo tanto la huella de carbono generada como los costes.

6. **Mayor capacidad** de utilizar la totalidad de los recursos del hardware.

En resumen, el uso de estas tecnologías aporta rentabilidad, mejoras en tiempo de comercialización, nuevos modelos de negocio, mejores infraestructuras, y una considerable reducción de nuestra huella de carbono.

6.2 Costes de recursos

Los costes relacionados con el uso/estudio de esta estructura lo vamos a dividir en diferentes partes, el hardware, software y tiempo necesario para estudiarlo en nivel de MicroStack y luego lo mismo a nivel de servidor más complejo.

6.2.1. Costes de MicroStack

En primer lugar, para no realizar mucho gasto en recursos en la empresa, lo mejor es hacer un estudio a nivel más pequeño con MicroStack. Para lograr esto necesitaremos un ordenador sobremesa con recursos similares a los siguientes:

- Procesador: AMD Ryzen 7 5700G 4.6GHz
- Tarjeta Gráfica: Modelo de gráficos Radeon™ Graphics
- Memoria RAM: 32 GB DDR4
- Almacenamiento: Kioxia Exceria G2 Unidad SSD 500GB M.2

Estos recursos se corresponden con el ordenador donde se ha realizado el estudio. Se podría realizar el estudio con un ordenador con menor potencia de procesamiento, aunque tanto la memoria como el almacenamiento debería de quedarse como mínimo con estos valores para asegurar un buen funcionamiento de todo el sistema, tanto el virtual como el sistema operativo que lo controla. Este hardware tiene un precio aproximado de unos 600€.

En cuanto a software, no tendríamos que contabilizar ningún precio, ya que tanto Ubuntu como MicroStack son totalmente gratuitos al ser open source, con lo cual no tendríamos que preocuparnos por este coste.

Por último, tendríamos que tener en cuenta el coste de recursos humanos, en el cual incluye la contratación de un programador aproximadamente unas 500 horas para estudiar y entender en profundidad el funcionamiento del sistema, así como para configurarlo debidamente. Si contamos que cobra unos 16€ la hora, nos sale un coste de 8.000€ aproximadamente.

Si sumamos ambas partes, realizar el estudio y tener una red completa en uso dentro de una empresa a nivel más pequeño, nos costaría unos 8600€, teniendo en cuenta que este estudio, a la hora de pasar a un sistema más complejo de servidor, haría muchísimo más sencillo para el desarrollador actualizar a este nuevo sistema.

6.2.2. Costes de Servidor

En este caso, la empresa debe de tener un servidor propio, en el cual se pueda incorporar todo el sistema. Estos servidores tiene un precio más elevado, que rondan entre los 3000 y los 6000€, dependiendo de las características que se les indique. Como ya sabemos, como mínimo deberán de cumplir estas dos características:

- Memoria RAM: 32 GB.
- Almacenamiento: 1TB.

En cuanto a software, no tendríamos que contabilizar ningún precio, al igual que sucede en el caso del coste de MicroStack.

Por último, deberíamos tener en cuenta el coste de recursos humanos, en el cual debería incluir un programador aproximadamente unas 300 horas para realizar todo el montaje y preparación como es debido, teniendo en cuenta que ya ha realizado el estudio y montaje previo de MicroStack. Si contamos que cobra unos 16€ la hora nos sale un coste de 5.000€, aproximadamente.

Si sumamos ambas partes, costaría unos 10.000€ en total, un precio mucho mayor a pagar, pero que nos terminaría ahorrando al cabo de unos años una gran cantidad de dinero y de recursos en un sistema convencional. No obstante, como podemos observar, el coste es bastante más elevado que con el estudio de MicroStack, el cual se puede usar igual, pero en menor escala.

6.3 Resumen

Recapitulando, desde un punto de vista económico para la empresa, es muy rentable utilizar este tipo de sistemas, al resultar en ahorros muy significativos para la misma, tanto en hardware como en recursos como electricidad, software, etc., aunque se deben de tener en cuenta múltiples factores antes de implementarlo, o antes de decidir cual versión utilizar, si la versión de MicroStack o OpenStack. En caso de que únicamente queramos realizar el estudio y implementarlo a nivel local o más pequeño dentro de la empresa, el uso de MicroStack e invertir el dinero en la investigación en muy buena idea, además de sencillo para los desarrolladores. En caso de tener pensado utilizarlo a gran escala, puede ser interesante centrarse directamente en el estudio del uso de OpenStack, por no hacer dos veces el mismo trabajo, aunque sea mucho más costoso.

Teniendo en cuenta todos los factores, es muy importante utilizar estos sistemas, que resultan muy útiles tanto nivel económico como ecológico, hecho que se detalla en el primer anexo.

CAPÍTULO 7
Conclusión

La estructura estudiada durante este trabajo tiene una dificultad relativamente elevada para un ingeniero junior, con poca experiencia, debido a la gran cantidad de conocimientos que se debe de tener en múltiples áreas, como redes, optimización de software, computación en la nube, uso de máquinas virtuales, etc.

Aunque la dificultad de despliegue y de realización de algunas tareas haya sido relativamente elevada, se han conseguido alcanzar todos los objetivos del proyecto, y se ha conseguido estudiar en detalle el uso de la plataforma, y comprender su funcionamiento en muchas partes de esta.

A continuación, expondremos las conclusiones que se ha extraído del estudio de la estructura de MicroStack, haciendo énfasis en virtudes y defectos del uso de esta plataforma, además de hablar sobre la relación que tiene este estudio con las asignaturas estudiadas en la carrera y con posibles ampliaciones que se pueden realizar.

7.1 Ventajas de uso

Como ya hemos estudiado anteriormente, esta estructura tiene miles de ventajas frente a la tradicional. Vamos a nombrar y comentar algunas de las que consideramos las más importantes.

- **Fácil instalación**: Microstack es muy sencillo de instalar y configurar, en comparación a la estructura de OpenStack de servidor. Esto hace posible que algunos usuarios sin experiencia previa y únicamente con conocimiento sean capaces de instalarlo y configurarlo.

- **Interfaz**: en el caso de MicroStack, puedes realizar todo desde una interfaz muy sencilla y simple de utilizar, sin obligarte a realizarlo todo por comandos.

- **Poca necesidad de recursos**: el sistema está planteado para que esté totalmente optimizado y dejar una huella de mínima, dejando así la posibilidad de que equipos con hardware más limitado puedan utilizarlo

- **Uso de arquitectura nativa de Kubernets**: al venir Microstack de una estructura nativa de Kubernetes, incluyendo así OCI para servicios de plano de control similares a OpenStack, para gestionar así de una manera más simplificada el ciclo de vida de las máquinas.

7.2 Desventajas de uso

Aunque esta estructura tenga muchas ventajas, también podemos encontrar algunas desventajas importantes. Vamos a nombrar y comentar algunas de las que consideramos las más importantes.

- **Limitaciones de escala**: como hemos comentado múltiples veces, MicroStack está pensado para entornos de nube en pequeña escala y no puede ser adecuado para implementar a gran escala. Para niveles más grandes es necesario un sistema más complejo.
- **Limitación de servicio**: al igual que en la escala, MicroStack solo viene con servidores centrales de OpenStack, con muchas de las opciones de estos eliminadas.
- **Requisitos de hardware**: Aunque como hemos dicho antes necesita pocos recursos en comparación a otros, sigue necesitando una cantidad de recursos considerable para funcionar a niveles más altos.
- **Dependencia de Snap**: dependemos totalmente de Snap, si trabajas en cualquier distribución que no pueda usar Snap no podrás utilizar el sistema.

7.3 Conclusión final

En cualquier caso, pese a las desventajas que plantea este sistema, cuenta con muchas más virtudes que defectos; además, como es sabido, sus desarrolladores pretenden seguir mejorando y sacando nuevas versiones mucho más sencillas de implementar y utilizar.

Para ser una alternativa más sencilla de utilizar que OpenStack, es una buena manera de aprender su funcionamiento, y poder prepararse para un futuro cambio en caso de ser necesario. No obstante, el uso de MicroStack está haciendo que, tanto este sistema como OpenStack, evolucionen con mayor velocidad al tener más usuarios que comprenden su uso.

7.4 Relación con las asignaturas cursadas

En relación con las asignaturas cursadas durante la carrera, este estudio tiene mucho que ver con todo lo que sería redes o gestión de servidores. Al tratarse de una investigación de una tecnología para estas estructuras, está relacionado principalmente con múltiples asignaturas de los últimos años de la carrera. Algunas de estas asignaturas serian:

- Redes corporativas
- Diseño y configuración de redes de área local
- Sistemas y servicios en red
- Administración de sistemas
- Integración de aplicaciones

Un buen ejemplo seria tanto con administración de sistemas o integración de aplicaciones, ya que tanto en MicroStack como en Openstack, se debería de hacer una buena administración del sistema, y definir correctamente los permisos que tiene cada usuario. Por otro lado, la integración de aplicaciones es muy útil, ya que se deben de adaptar parcialmente las aplicaciones para que sean capaces de sacar el máximo provecho dentro de nuestro sistema.

Otro claro ejemplo seria diseño y configuración de redes de área local, asignatura que nos ayuda bastante en el estudio de MicroStack y sus usos, ya que está tecnología esta centrada en el uso a nivel local de una empresa.

Aparte de estas asignaturas, también se acerca a otras asignaturas, pero estas son menos relevantes en el estudio realizado.

7.5 Posibles ampliaciones

Después de las pruebas y estudio realizado, podemos proponer nuevas líneas de estudio para posibles proyectos o futuros estudios en el uso de MicroStack.

Un posible estudio que se podría realizar es el uso conjunto de múltiples despliegues dentro de la arquitectura de MicroStack, para tratar de alcanzar el límite teórico que posee la estructura. En concreto se trataría de utilizar la estructura completa y todo lo estudiado (sistemas, uso de redes, grupos, etc.) para conseguir desplegar múltiples máquinas virtuales que hagan cada una la función de una parte de una aplicación: una máquina de base de datos, otra de hosteando la aplicación, etc. De esta manera se podría comprobar hasta qué nivel MicroStack puede gestionar una estructura más compleja, además de conseguir ver las estadísticas o resultados más precisos.

Otra posible idea sería buscar la manera de que, en el caso de que las máquinas estén al máximo de carga, tratar de automatizar la división de la carga en dos máquinas virtuales dentro del sistema, es decir, generar otra máquina de las mismas características para que se dividan la carga. De esta manera se podría estudiar una funcionalidad parecida a la creada por la empresa VMware, que divide la carga en máquinas de manera automática y a gran velocidad, con la idea de mejorar el sistema.

Estas serían dos de las posibles ampliaciones que se podrían realizar, además de realizar pruebas con algún sistema con características más limitadas, para probar la adaptabilidad de este.

Bibliografía

[1] ¿Qué es la NFV?, Publicado 16 de Agosto de 2019. Consultado en https://www.redhat.com/es/topics/virtualization/what-is-nfv.

[2] Network Functions Virtualization Overview, Actualizado el 31/05/2019. Consultado en https://docs.vmware.com/en/VMware-vCloud-NFV/2.0/vmware-vcloud-nfv-openstack-edition-ra20/GUID-FBEA6C6B-54D8-4A37-87B1-D825F9E0DBC7.html.

[3] NFVI Components Overview, Actualizado el 03/08/2020. Consultado en https://docs.vmware.com/en/VMware-vCloud-NFV/2.0/vmware-vcloud-nfv-reference-architecture-20/GUID-4EEE8C63-E3BE-41D1-B0FA-D14D09616770.html.

[4] Logical Architecture and Components, Actualizado el 10/07/2020. Consultado en https://docs.vmware.com/en/VMware-vCloud-NFV/3.2.1/vmware-vcloud-nfv-reference-architecture-321/GUID-57F38988-4546-41DD-A133-9FEFEF04D499.html.

[5] ETSI NFV Management and Orchestration, Publicado en "Noticias SG38.en el año 2014. Consultado en https://www.ietf.org/proceedings/88/slides/slides-88-opsawg-6.pdf.

[6] Tear down your OpenStack lab environment. Consultado en https://ubuntu.com/tutorials/tear-down-your-openstack-lab-environment#1-overview.

[7] OpenStack, Publicado el 2/1/14. Consultado en https://sg.com.mx/revista/38/openstack.

[8] OpenStack on Kubernetes. Consultado en https://microstack.run.

[9] Install OpenStack yourself. Consultado en https://ubuntu.com/openstack/install.

[10] ¿Qué es la seguridad de red?. Consultado en https://www.ibm.com/mx-es/topics/network-security.

[11] Guía de administración y utilidades del controlador OpenStack Nova 1.0 en Oracle® VM Server for SPARC, Publicado en septiembre de 2016. Consultado en https://docs.oracle.com/cd/E71843_01/pdf/E79751.pdf.

[12] Install and configure controller node for Ubuntu, Actualizado el 2022-05-24. Consultado en https://docs.openstack.org/nova/rocky/install/controller-install-ubuntu.html.

[13] WHAT IS OSM?. Consultado en https://osm.etsi.org.

[14] Flavors. Actualizado en 2021-01-29 . Consultado en https://docs.openstack.org/nova/latest/user/flavors.html.

[15] Guía del administrador de VMware Integrated OpenStack.Actualizado el 31/05/2019 Consultado en https://docs.vmware.com/es/VMware-Integrated-OpenStack/4.0/com.vmware.openstack.admin.doc/GUID-D0F1EB56-6CC0-4C09-9B41-F9845524F04A.html.

[16] Las claves de OpenStack Cloud. Publicado el 25.11.2021 Consultado en https://www.ionos.mx/digitalguide/servidores/herramientas/que-es-openstack/.

[17] Trabajar con grupos de seguridad. Actualizado el 31/05/2019 Consultado en https://docs.vmware.com/es/VMware-Integrated-OpenStack/4.0/com.vmware.openstack.admin.doc/GUID-D0F1EB56-6CC0-4C09-9B41-F9845524F04A.html.

[18] Creating and Managing Instances . Actualizado el 2023-04-26. Consultado en https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.2/html-single/creating_and_managing_instances/index.

[19] Create and access a database instance. Actualizado el 2021-08-02. Consultado en https://docs.openstack.org/trove/latest/user/create-db.html.

[20] Host networking . Actualizado el 2023-08-17. Consultado en https://docs.openstack.org/install-guide/environment-networking.html.

[21] Desplegando infraestructura en Openstack con Terraform . Publicado el 29/03/2023. Consultado en https://blog.sysdual.com/desplegando-infraestructura-en-openstack-con-terraform/.

[22] Kevin Jackson, Cody Bunch, Egle Sigler, *OpenStack Cloud Computing Cookbook - Fourth Edition*, Packt Publishing, 2016. Disponible en: https://www.packtpub.com/product/openstack-cloud-computing-cookbook/9781782174783

[23] Manoj Hirway, *Hybrid Cloud for Developers*, Packt Publishing, 2018. Disponible en: https://www.packtpub.com/product/hybrid-cloud-for-developers/9781788830874

[24] Omar Khedher, Chandan Dutta Chowdhury, *Mastering OpenStack - Second Edition*, Packt Publishing, 2017. Disponible en: https://www.packtpub.com/product/mastering-openstack-second-edition/9781786463982

APÉNDICE A
OBJETIVOS DE DESARROLLO SOSTENIBLE

Tabla A.1: Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. Fin de la pobreza.				X
ODS 2. Hambre cero.				X
ODS 3. Salud y bienestar.				X
ODS 4. Educación de calidad.				X
ODS 5. Igualdad de género.				X
ODS 6. Agua limpia y saneamiento.				X
ODS 7. Energía asequible y no contaminante.	X			
ODS 8. Trabajo decente y crecimiento económico.	X			
ODS 9. Industria, innovación e infraestructuras.	X			
ODS 10. Reducción de las desigualdades.				X
ODS 11. Ciudades y comunidades sostenibles.			X	
ODS 12. Producción y consumo responsables.	X			
ODS 13. Acción por el clima.		X		
ODS 14. Vida submarina.				X
ODS 15. Vida de ecosistemas terrestres.				X
ODS 16. Paz, justicia e instituciones sólidas.				X
ODS 17. Alianzas para lograr objetivos.	X			

De los objetivos de desarrollo sostenible señalados en la tabla anterior, el uso de estructuras como OpenStack o MicroStack está relacionado con:

- **ODS 7. Energía asequible y no contaminante:** El uso de estructuras y tecnologías en la nube, como OpenStack, MicroStack u OSM, tiene múltiples ventajas a nivel energético y contaminante, ya que esta tecnología está pensada principalmente para maximizar la capacidad energética de los sistemas convencionales, y evitar la necesidad de utilizar hardware extra.
- **ODS 8. Trabajo decente y crecimiento económico:** Esto es posible, ya que, al usar esta tecnología, las empresas gastarán mucho menos recursos económicos, provocando así un crecimiento económico mayor dentro de esas empresas.
- **ODS 9. Industria, innovación e infraestructuras**: Esta tecnología es muy novedosa y potente para las infraestructuras actuales hasta el punto de que, conforme va

pasando el tiempo y se va conociendo más, va cambiando el uso de servidores tal y como se conocen, trasladando la mayoría a este nuevo formato.

- **ODS 11. Ciudades y comunidades sostenibles**: Aunque sea en menor medida, esto permite que los servidores centrales de las ciudades como Valencia, encargados de mantener gran parte de la cuidad, contaminen menos, y así sean mucho más sostenibles.
- **ODS 12. Producción y consumo responsables:** En este apartado, nuestro estudio es muy importante ya que hace que las empresas no gasten tanto ni cambien tanto el hardware, y además es capaz de crear un ecosistema circular, donde los recursos vayan pasando de empresa a empresa dependiendo de sus necesidades.
- **ODS 13. Acción por el clima:** Como ya es sabido, todo el hardware genera mucho calor que necesita ser disipado; al eliminar gran parte del hardware necesario en los servidores con esta tecnología, bajaríamos mucho la temperatura de acción de estos sistemas.
- **ODS 17. Alianzas para lograr objetivos:** Por último, esta tecnología es OpenSource, con lo cual está pensado para que cualquier usuario la estudie, la mejore, y conseguir entre una gran cantidad de usuarios que este sistema sea mucho más potente y útil.