UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA EN TELECOMUNICACIONES

IMPLEMENTACIÓN DE UN PROTOTIPO DE NUBE PARA BRINDAR INFRAESTRUCTURA COMO SERVICIO (IAAS) EN EL LABORATORIO DE TELEMÁTICA DE LA UCAB

TRABAJO DE GRADO

Presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

Como parte de los requisitos para optar al título de

INGENIERO EN TELECOMUNICACIONES

REALIZADO POR: Casique Pérez, Luis David. C.I. V-26.901.091

Piña Oliva, Isaac Gabriel. C.I. V-27.037.250

TUTOR Ing. José Pirrone Puma. C.I. V-5.533.711

FECHA Septiembre de 2022

AGRADECIMIENTOS

De Luis David:

A Dios, por su infinita misericordia y guía a lo largo de mi vida, por la oportunidad de estudiar esta maravillosa carrera, su apoyo incondicional durante mis años en la misma y compañía durante todas las etapas de mi vida. A mi madre Macxy, por su amor, constante apoyo, palabra de aliento, oración y clamor por mí en cada aspecto de mi vida, incluyendo mis estudios en esta carrera y este trabajo de grado. A mi padre Jhonny, por su buen consejo en la realización de este proyecto y sus buenas intenciones en enseñarme, y por su inagotable esfuerzo, apoyo y sustento a nivel económico en cada etapa de mi educación. A mi hermano Gustavo, por su apoyo, palabra de aliento, guía y ayuda en cada etapa de mi educación como actual ingeniero y egresado de la universidad, con el fin de conseguir ser su futuro colega y prosperar en nuestras vidas, incluyendo también un importante esfuerzo y sustento económico. A mi abuela Marta, por su incondicional apoyo en oración por mí para lograr conseguir mis objetivos a lo largo de mi vida, a pesar de los obstáculos y dificultades encontradas en el camino, siempre con la fe firme de superarlas, como han sido superadas. A mis tíos Jesús, Freddy y Milixy, de quienes recibí en todo momento apoyo incondicional y numerosos momentos de escape, liberación y despeje mental, que hicieron mi carga estudiantil mucho más llevadera. A mi primita Camila que, a pesar de su corta edad, me ha enseñado muchas cosas y me ha apoyado incondicionalmente como una hermana a lo largo de su vida. Y a todas las bonitas amistades y personas que Dios me presentó en este camino, quienes nos apoyamos mutuamente hicieron disfrutable mi carrera al punto de considerarla la etapa más bonita de mi formación profesional y de mi vida.

De Isaac Piña:

Agradezco a todas las personas que me acompañaron y apoyaron a lo largo de mi formación universitaria y de mi vida. En especial, doy gracias a mi familia, la cual me ha respaldado en los diferentes caminos que he decidido emprender a lo largo de mi vida. Asimismo, quiero agradecer a mis amigos, quienes estuvieron allí para mí, brindándome el apoyo necesario para hacer frente a los retos que se nos presentaron a lo largo de la carrera.

Deseamos agradecer de manera conjunta y especial al profesor José Pirrone, quien confió en nosotros para implementar este Trabajo de Grado, y de quien recibimos en todo momento especial atención, orientación, consejo y apoyo a lo largo de la realización del presente proyecto.

RESUMEN

El presente tomo recopila el conjunto de actividades y procesos realizados desde el mes de octubre del año 2021 hasta la fecha, para la consecución del proyecto de implementación de una nube prototipo, con la infraestructura computacional del Laboratorio de Telemática de la institución, que permita compartir y provisionar sus respectivos recursos de *hardware* en un formato de Infraestructura como Servicio (*Infrastructure as a Service,* IaaS).

Este trabajo de compartición de recursos fue logrado por medio del sistema operativo UBUNTU, la aplicación gestora y provisora de recursos de cómputo *MICROSTACK*, implementada en modalidad de nodos múltiples, computadoras del Laboratorio de Telemática, su infraestructura de red utilizada para la impartición de las asignaturas de laboratorios, un conjunto de cables *Ethernet* y un dispositivo conmutador de computadoras (*switch*) para dicha red.

Primero, se realizó un despliegue de una nube pequeña de forma local, utilizando computadoras pertenecientes a los autores del proyecto, con el fin de ensayar, aprender y familiarizarse con el uso de las herramientas del sistema UBUNTU y la aplicación *MICROSTACK*, así como las dificultades que su uso conllevó. Posteriormente, con la experiencia adquirida en el despliegue anterior, se implementó la nube en el laboratorio de telemática en la universidad, utilizando cinco de sus computadoras pertenecientes a su infraestructura conectadas como un macrobloque de cómputo, y se creó un despliegue completo que permite virtualizar un ambiente de redes para realizar pruebas de distintas capas, virtualizando el *hardware* de las computadoras que conforman la nube.

Adicionalmente, se encontraron actividades impartidas en las asignaturas del laboratorio y otros usos prácticos de la nube que le saquen el mayor provecho posible, dentro de las limitaciones de *hardware* de los equipos que conforman a la misma.

Palabras clave: Infraestructura como Servicio, *MICROSTACK*, nube, despliegue, nodos.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICOS

INTRODUCCIÓN

El crecimiento fugaz y exponencial de las tecnologías de *cloud computing* ha convertido en este tópico en una nueva competencia que el estudiante de una carrera ligada a las tecnologías de información debe adquirir. No obstante, hasta los momentos, este tópico no es abordado en ninguna rama académica ni administrativa perteneciente a la carrera de ingeniería en telecomunicaciones, con lo cual, surge el interés de la escuela de introducirse de lleno a este tópico a la brevedad posible. Una solución que permite introducirse de lleno en el tema y evaluar la factibilidad de aplicarla a un ámbito académico, es la implementación de un prototipo de nube que permita compartir los recursos de cómputo de las computadoras del laboratorio de telemática, de lo cual trata el presente Trabajo de Grado.

La estructura del cuerpo central del presente tomo documenta el proyecto en un total de seis capítulos, descritos y distribuidos de la siguiente manera:

En el Capítulo I, se describen todos los aspectos relacionados a los motivos que llevan a la realización del proyecto, sus objetivos, alcances, limitaciones y su respectiva justificación de realización. En el Capítulo II se encuentra el marco teórico, que recopila todos los temas y fundamentos que dan sustento y aportes teóricos al proyecto, incluyendo la descripción de las herramientas utilizadas en su implementación. El Capítulo III, envuelve la planificación metodológica del proyecto distribuida en fases de planificadas de desarrollo, que distribuyen la carga del proyecto en lapsos de tiempo (semanas).

El capítulo IV, expresa detalladamente el proceder de cada una de las actividades realizadas en la elaboración e implementación del proyecto en cada una de sus fases de desarrollo descritas en el capítulo anterior. El Capítulo V plasma de forma tangible los resultados conseguidos a lo largo de toda la realización del proyecto, identificados y descritos cada uno de ellos.

En el Capítulo VI, se recopila un conjunto de observaciones concluidas durante el desarrollo del proyecto, así como también, una serie de recomendaciones para el desarrollo de futuros proyectos similares. Y, por último, se presenta una lista con toda la bibliografía consultada y citada en el presente tomo y en el desarrollo del proyecto.

CAPÍTULO I. PLANTEAMIENTO DEL PROYECTO

En el presente capítulo se describen los elementos que constituyen la base del proyecto llevado a cabo. En primer lugar, se describe el problema que motivó la realización del proyecto; en segundo lugar, se definen los objetivos generales y específicos del mismo; en tercer lugar, se especifican los alcances y las limitaciones del proyecto, y finalmente, se desarrolla la justificación, donde se argumenta por qué debe atenderse la problemática que motivó a proponer el proyecto.

I.1. Planteamiento del problema

A lo largo de los últimos años, la computación en la nube se ha hecho presente en las diferentes implementaciones de telecomunicaciones y de tecnologías de la información en general, pues permite a las empresas tener acceso a recursos bajo demanda y en lapsos de tiempo reducidos, sin la necesidad de ocupar espacio adicional en sus instalaciones. Atendiendo el anterior orden de ideas, se debe mencionar que la implementación de infraestructuras de computación en la nube se ha establecido como una competencia de cada vez mayor importancia dentro del ejercicio profesional de la ingeniería en telecomunicaciones. Por ello, resulta de interés incorporar el estudio teórico/práctico de dichos tópicos dentro de las asignaturas de la línea telemática, dado que actualmente no son abordados en las mismas [1] [2] [3], en aras de robustecer el perfil de los egresados de la carrera.

Para introducirse de lleno al tema descrito, se ha implementado un prototipo de nube que permite brindar Infraestructura como Servicio (*Infrastructure as a service*, IaaS) como una herramienta que servirá de forma esencial para estudiar sus características de implementación y utilización, y para evaluar la eficacia de ejecutar sistemas operativos sobre instancias de la nube con un número reducido de nodos. Adicionalmente, la realización de este proyecto busca sentar las bases para aprovechar de manera eficiente los recursos del Laboratorio de Telemática, y en futuros proyectos, garantizar el acceso a los alumnos a los recursos, de cómputo y de red, que se ajusten a sus necesidades académicas.

El prototipo de nube para brindar IaaS está gestionado mediante la aplicación *MICROSTACK*, el cual es un compendio de módulos y paquetes de *OPENSTACK* [4]. La razón por la cual se escogió *MICROSTACK*, es porque permite desarrollar un conocimiento previo que

permite evaluar como dimensionar la nube de *OPENSTACK*, usando sus módulos completos. *OPENSTACK* es un *software* libre, con un desarrollo continuo, y una amplia comunidad, lo cual brinda acceso a manuales y sugerencias actualizadas. Contará con cinco nodos de cómputo, uno de los cuales será también el nodo de control.

I.2. Objetivos

I.2.1. Objetivo General

Implementar un prototipo de nube basada en *MICROSTACK*, para brindar Infraestructura como Servicio en el Laboratorio de Telemática de la Universidad Católica Andrés Bello.

I.2.2. Objetivos Específicos

- Investigar los recursos y servicios basados en la nube como la Infraestructura Como Servicio (IaaS) y la herramienta de cómputo en la nube *MICROSTACK*.
- Realizar la instalación local y configuración del sistema operativo UBUNTU y de *MICROSTACK*.
- Estudiar y experimentar con los comandos de ejecución de la consola de *MICROSTACK*.
- Realizar pruebas de habilitación y levantamiento de servicios basados en la nube en dispositivos locales. Creación de instancias para la solicitud de recursos como perfiles virtuales.
- Establecer conexión remota desde dispositivos externos los dispositivos locales en los que se implementó la nube.
- Instalar e implementar *MICROSTACK* en las computadoras del Laboratorio de Telemática.
- Realizar solicitudes de recursos de *hardware* a las computadoras del laboratorio desde un dispositivo externo a la nube.
- Determinar parámetros para evaluar rendimiento de las instancias de la nube.
- Evaluar el desempeño y rendimiento de las instancias.
- Estudiar la posibilidad de emplear la infraestructura de nube para ejecutar las plataformas de simulación empleadas en las cátedras de Laboratorio de Telemática I y Laboratorio de Telemática II.

I.3. Alcances y limitaciones

I.3.1. Alcances

- El usuario puede crear instancias a partir de configuraciones de memoria RAM, almacenamiento, y capacidad de procesamiento, predefinidas por el administrador de la nube.
- Se puede acceder a la interfaz de comandos del sistema operativo de cada una de las instancias de la nube.
- La plataforma cuenta con un sistema de autenticación que solicitará credenciales al usuario que desee hacer uso de la plataforma de computación en la nube.
- Se incluirá una guía de uso para el usuario, y una guía de creación y gestión para los administradores de la plataforma.
- Se presentará una tabla comparativa entre los recursos utilizados para la realización de las prácticas de Laboratorio de Telemática I y Laboratorio de Telemática II y los recursos que se pudieran proveer por la nube, indicando cuáles prácticas se pueden ejecutar sin mayores inconvenientes, y cuáles requieren una adaptación para emplear la infraestructura de la nube como medio para realizar actividades de Laboratorio.

I.3.2. Limitaciones

- Este Trabajo de Grado prevé la instauración de una nube de cinco nodos de cómputo, donde uno de ellos es el nodo de control.
- Los nodos ejecutarán *MICROSTACK*, y estarán ejecutándose en computadores que tengan la versión de LINUX UBUNTU más reciente, para el momento de montaje, que soporte *MICROSTACK*.
- Los recursos de *hardware* (memoria RAM, almacenamiento, ancho de banda y potencia de procesamiento), están limitados por los recursos de cada desktop encontrada en el Laboratorio de Telemática.
- El usuario de la nube podrá escoger entre un conjunto de configuraciones de *hardware* predefinido por el administrador para crear cada instancia, y la creación se llevará a cabo de acuerdo con los recursos disponibles.

- El acceso a la nube solo será desde una computadora que pertenezca a la misma LAN que las computadoras que conformen la nube, y será sobre protocolo SSH empleando la aplicación PuTTY.
- La velocidad de transmisión estará condicionada por los medios disponibles en el Laboratorio de Telemática.
- La realización de este proyecto no contempla la adaptación a la nube de las prácticas llevadas a cabo en Laboratorio de Telemática I y Laboratorio de Telemática II.
- No se adecuará *software* que no sea compatible de manera nativa con la plataforma de nube.

I.4. Justificación

En los últimos años, se ha desarrollado una aceleración masiva de los procesos de virtualización de los Sistemas y Tecnologías de Información (*Information Technologies*, IT) por varias razones que van desde la posibilidad de poder abrir camino a mayores posibilidades laborales, como los trabajos remotos o teletrabajos, hasta reducción de la emisión de CO_2 a la atmósfera, pasando por el aprovechamiento de servidores sin uso en los distintos operadores existentes. Entre los sistemas de información mencionados, la computación en la nube o *Cloud Computing* ha sido una de las soluciones que ha crecido como servicio de forma importante desde la llegada de la pandemia, permitiendo la continuidad de muchos empleos a distancia, por lo que esta irrupción se ha transformado en un "*camino sin vuelta atrás*" [5].

Adicionalmente, la computación en la nube permite a los usuarios acceder a recursos de acuerdo con sus necesidades, sin requerir tener *hardware* de última generación, pues la mayor demanda de recursos de cómputo recae sobre los servidores del proveedor del servicio informático. *Microsoft* resalta al respecto que: "*La Infraestructura como Servicio (IaaS) permite evitar el costo y la complejidad de comprar y administrar servidores físicos e infraestructura de centro de datos*" [6]. Asimismo, la nube brinda portabilidad al trabajo, pues el usuario puede acceder a la misma información desde numerosos dispositivos, y, adicionalmente, brinda un respaldo al usuario de su trabajo, lo cual reduce los riesgos de perder información relevante para el mismo.

La implementación del prototipo de nube propuesto le otorga mayor funcionalidad a la infraestructura del laboratorio de telemática de la universidad y permite a la institución introducirse

de lleno en el tópico del *cloud computing* utilizando su propia infraestructura de computadoras, para su próxima inclusión en el estudio teórico práctico en el área de telemática de la carrera, añadiendo una nueva competencia al perfil del estudiante de la misma. Además, esta solución le da otro tipo de usabilidad a dicha infraestructura, cuyas funcionalidades específicas son uno de los objetos de estudio del presente proyecto.

I.5. Antecedentes de la investigación

Previamente a la realización del presente proyecto, se ha realizado un trabajo de grado relacionado al *cloud computing*; la dirección de la escuela de telecomunicaciones tutorizó la implementación de una nube pequeña utilizando *OPENSTACK*, en el proyecto titulado "*Desarrollo de un sistema de red LAN virtualizada mediante el uso del software libre OPENSTACK*" llevada a cabo por los actualmente ingenieros Christian Ascanio y Fernando Simoes en el año 2018, con el fin de adentrarse a la implementación de infraestructuras de *cloud computing*, para brindar de igual forma Infraestructura como Servicio (IaaS), implementar el diseño de redes LAN virtualizadas y para servir fundamentalmente de base para la implementación del presente proyecto. Con las diferencias de haber implementado los módulos independientes de *OPENSTACK* fuera del paquete *snap* de *MICROSTACK*, tales como *Nova*, *Glance*, *Neutron*, *Keystone*, *Cinder* y *Horizon*; y también de haber utilizado una infraestructura física de cómputo propia, independiente de la infraestructura de algún espacio de trabajo perteneciente a los laboratorios de la institución.

El presente proyecto es el primer despliegue de una infraestructura de *cloud computing* realizado en un departamento académico de la institución.

CAPÍTULO II. MARCO TEÓRICO

El presente capítulo comprende un amplio compendio sobre los temas y aspectos teóricos que abarca el proyecto desde su planificación hasta su ejecución. La piedra angular sobre la que se fundamenta el mismo es la Infraestructura como Servicio (IaaS), no obstante, el proyecto abarca de forma directa la rama de telemática de la carrera de ingeniería en telecomunicaciones. En dicha rama, se tocan los temas de redes de computadoras, tipos de redes y diseño de redes en capas; adicionalmente, abarca temas de otras ramas como el *hardware* y *software*, uso de sistemas operativos, gestión de recursos de cómputo, virtualización y redes definidas por *software*.

Adicionalmente, ahondando en las herramientas que permiten el desarrollo del proyecto, se hace un repaso, y descripción del sistema operativo UBUNTU, la aplicación de gestión y provisión de *hardware OPENSTACK* y su implementación por medio del paquete *snap MICROSTACK*.

Previo al inicio del marco teórico, se muestra una esquematización de cada uno de los temas que se tocan por medio del siguiente mapa conceptual:

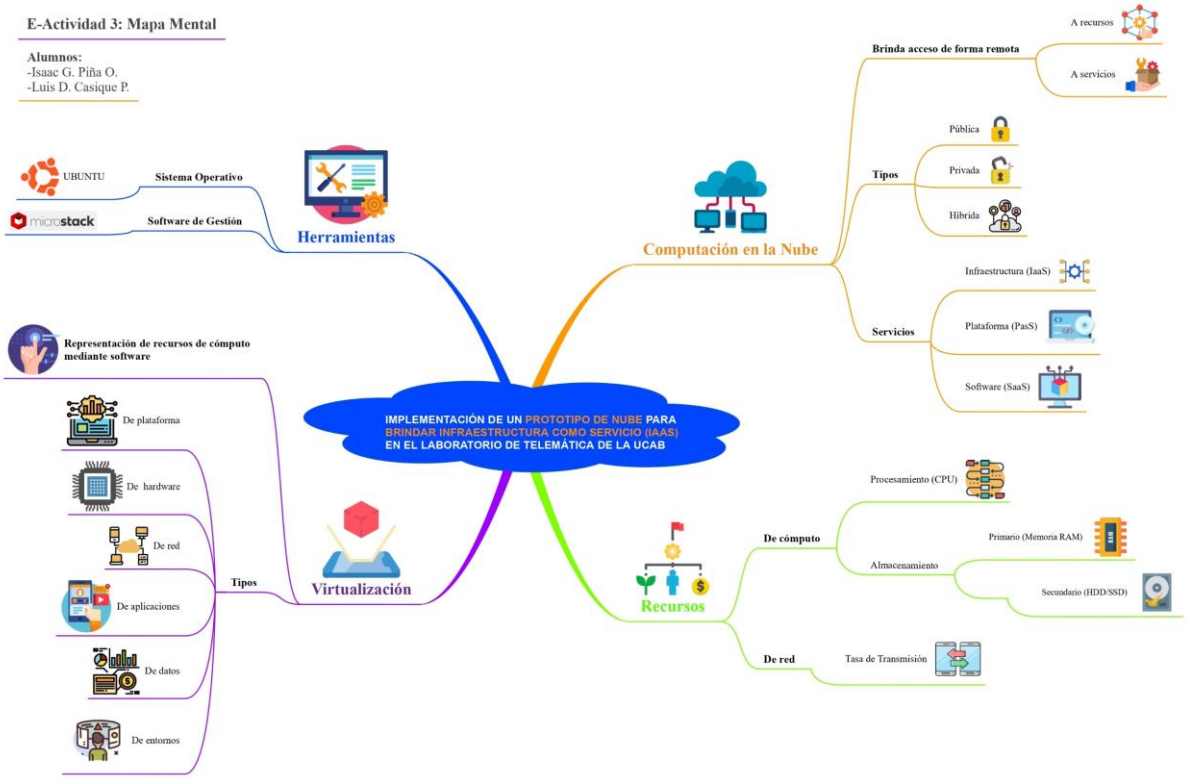


Figura 1. Mapa conceptual esquematizado de los temas asociados al proyecto.

II.1. Redes definidas por *software* (*Software Defined Networking*, SDN)

Es un enfoque de arquitecturas de red que busca separar de forma lógica la gestión y control de una red de su *hardware* físico. Esto permite que un amplio conjunto de aplicaciones y servicios que utilizan Interfaces de Programación de Aplicaciones (*Application Programming Interfaces*, API) abiertas programe el comportamiento de la red, lo cual brinda a los proveedores de servicios el control completo sobre las redes, y los equipos que las conforman, de forma consistente, dinámica y escalable, independientemente del *hardware* o de la tecnología de red subyacente. Mediante la implementación de estándares sobre las interfaces programables, SDN permite la gestión y control de las redes de forma unificada [7].

II.2. Computación en la Nube (*Cloud Computing*)

En el ámbito de las Tecnologías de Información y Comunicaciones (TIC), la nube se refiere a un conjunto de recursos de computadora (redes, servidores, almacenamiento, aplicaciones y servicios), que son brindados a usuarios, exigiéndoles el menor esfuerzo, y la mínima interacción con el proveedor. Se caracterizan por ser ubicuos, y porque el usuario accede solamente a los recursos que requiere, por lo que se le cataloga como un servicio bajo demanda. Supone un cambio de paradigma para la manera en la que se accede al contenido digital, pues antes se requería tenerlo almacenado localmente para consumirlo [8] [9] [10].

En la nube, es posible almacenar datos, y acceder a ellos cuando sea necesario, permite brindar servicio de correo electrónico, proveer *streaming* de contenido multimedia, y brindar acceso a poder de cómputo [8] [9] [10]. A continuación, se ilustran los componentes empleados en la computación basada en la nube:

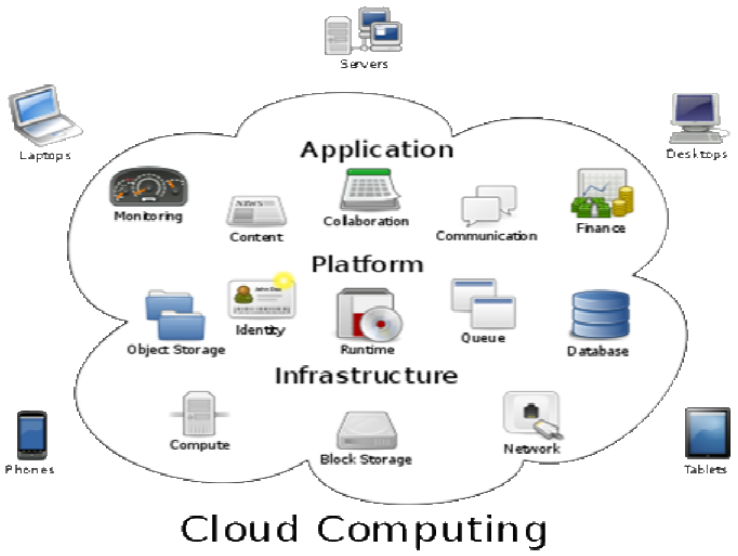


Figura 2. Capas de la Computación en la Nube [9].

II.2.1. Tipos de Nubes

Las nubes se pueden clasificar según el tipo de servicio o contenido que ofrece, según el tipo de identidad, grupo o empresa que las implementan y según la dimensión o sectorización del público hacia el que están dirigidas. Principalmente, distinguen tres tipos de nubes:

II.2.1.1. Nube Pública

Es quizás el tipo de nube más conocido por el usuario promedio. En este tipo de nube se comparten recursos y ofrece servicios al público general (o un importante público industrial [8]) a través de Internet [11]. La principal ventaja de este tipo de nube es su capacidad de procesamiento, y almacenamiento, sin tener que instalar *hardware* de forma local, por lo que no es necesario realizar inversiones iniciales, o gastos de mantenimiento, sino únicamente pagar por su uso [9].

II.2.1.2. Nube Privada

Es una nube implementada, gestionada y operada por una organización privada [9], donde se encuentran físicamente los servidores y equipos que la conforman. En general, son implementadas para ofrecer *hardware* e infraestructura de red (*Infrastructure as a Service*, IaaS), plataforma (*Platform as a Service*, PaaS), y aplicaciones (*Software as a Service*, SaaS). La localización física de los equipos dentro de la organización gestora tiene como ventaja una mayor seguridad de los datos y mayor facilidad para entregar los servicios; mientras que su principal desventaja son los costos periódicos para mantenimiento [9].

II.2.1.3. Nube Híbrida

Este tipo de nube cuenta con características tanto de nube pública como privada (o comunitaria), combinando las aplicaciones locales con las que se alojan en Internet. Se define también como una composición de dos o más nubes (privadas, comunitarias o públicas) que siguen siendo entidades independientes pero que funcionan en el mismo ecosistema estando unidas por tecnología estandarizada, o patentada, que permite la compatibilidad y portabilidad de los datos entre las nubes involucradas [8]. Generalmente las organizaciones utilizan la parte pública para servicios genéricos (como correo electrónico, o gestión de nóminas) y la parte privada es reservada para sus datos analíticos [10].

II.3. Servicios Brindados en la Nube

II.3.1. Software como Servicio (Software as a Service, SaaS*)*

Es el más limitado de los servicios basados en la nube. Permite la ejecución de aplicaciones y programas desarrolladas por el proveedor del servicio, sin necesidad de estar instaladas en los equipos de usuario. Cualquier dispositivo móvil, o fijo, que disponga de una conexión a internet, como celulares, tabletas o computadoras; se puede acceder a los servicios proporcionados por estas aplicaciones a través de una interfaz de cliente, como un navegador web. El usuario no interactúa con el *hardware*, el sistema operativo, la red, o incluso las capacidades individuales de las aplicaciones a las que accede; únicamente, controla los ajustes de la aplicación solicitada [8]. Ofrece dos tipos de servicio:

- **Servicios Empresariales:** Aplicaciones de apoyo empresarial, como gestión de flujos de trabajo, cadenas de suministro, firmas digitales, *software* de escritorio, correo electrónico, entre otros [8].
- **Aplicaciones *Web* 2.0:** Gestión de metadatos, redes sociales, blogs, servicios wiki, y servicios de portal [8].

II.3.2. Plataforma como Servicio (Platform as a Service, PaaS*)*

Permite la implementación de aplicaciones creadas, o adquiridas, por el consumidor, utilizando lenguajes de programación, y herramientas compatibles con el proveedor. En este nivel,

el usuario no controla, ni administra, la infraestructura local de la nube; tiene control sobre las aplicaciones implementadas, y de la configuración del entorno de alojamiento de las mismas [8].

Entre los servicios que se ofrecen están: Gestión de sesiones, integración de dispositivos, instrumentos y pruebas, gestión de contenidos, gestión de conocimientos, entre otros [8].

II.3.3. Infraestructura como Servicio (Infrastructure as a Service, **IaaS***)*

Es de los servicios basados en nube en los que se le da mayor control del *hardware* al usuario. Tiene la capacidad de proveer procesamiento, almacenamiento, redes, y otros servicios informáticos fundamentales. En este nivel de servicio, el consumidor puede implementar, y ejecutar *software* de cualquier tipo, incluyendo sistemas operativos y aplicaciones; no tiene control sobre la infraestructura de nube subyacente, pero sí tiene control sobre los sistemas operativos, almacenamiento, y algunos componentes de gestión de seguridad de red como cortafuegos [8].

Entre las posibilidades de servicio que se provisionan están: Alojamiento de servidores, servidores web, almacenamiento, *hardware*, sistemas operativos, instancias virtuales, balance de cargas, acceso a internet y aprovisionamiento de velocidad de conexión [8].

Este nivel de servicios basados en la nube le permite al usuario un mayor repertorio de opciones de servicio a solicitar con respecto a los niveles más bajos, al punto de que numerosas empresas, han comenzado a ofrecer Infraestructura como Servicio, como, por ejemplo: ABICLOUD, AMAZON WEBSERVICES EC2, GOGRID, entre otras [8].

La comparación entre los niveles de servicios basados en la nube, tanto como las ventajas y desventajas de cada uno de ellos, puede resumirse en la siguiente figura:

	IaaS	*PaaS*	*SaaS*
¿Qué ofrece?	Infraestructura virtual alquilada por uso (pay-as-you-go).	Plataforma capaz de ejecutar el código deseado a través de aplicaciones.	*Software* ready-to-use a través de la web.
Uso recomendado	Cargas de trabajo variables (balanceo de carga), gran cantidad de tareas en paralelo. *Quality Assurance* y testing (máquinas virtuales, servidores, alojamiento, red).	Ejecución de aplicaciones relativamente simples que no requieren control sobre la topología de red, el sistema operativo o la dirección de almacenamiento de los datos (bases de datos, servidores web, herramientas de desarrollo).	Herramientas ofimáticas (mail, procesador de textos, herramientas de colaboración, escritorio virtual, comunicaciones, juegos) y bases de datos de baja complejidad (CRM).
Contenido en el *Cloud*	Sistema operativo o máquina virtual.	Código fuente de las aplicaciones o herramientas.	Datos y procesos del negocio. Se traslada al *Cloud* una funcionalidad correspondiente al negocio.
Ventajas	El proveedor de servicios de cómputo en el *Cloud* es responsable de la administración de los equipos y de solventar los problemas relacionados con los mismos. Reducción de costes gracias a la modalidad (pay-as-you-go) y a la ausencia de responsabilidad sobre la instalación, administración y mantenimiento de los equipos. Escalabilidad prácticamente automática y transparente para los usuarios del servicio.	El proveedor de servicios de cómputo en el *Cloud* es responsable de la administración tanto del *hardware* como del *software* sobre el que se ejecutan las aplicaciones (sistema operativo). El cliente es un mero usuario de las soluciones ofrecidas en el *Cloud* sin necesidad de instalar, configurar ni dar mantenimiento a los sistemas. Escalabilidad prácticamente automática y transparente para los usuarios del servicio. Desarrollo de aplicaciones más sencillo ya que las tareas propensas a ser ineficientes (p. ej. manejo de datos) tienen APIs asociadas propuestas por el propio proveedor de servicios en el *Cloud*. Modularidad en el desarrollo.	Menor inversión inicial y por tanto menor riesgo de tipo económico ya que el proveedor de servicios de cómputo en el *Cloud* es responsable de la administración tanto del *hardware* como del *software*, así como de las aplicaciones y los datos. Reducción de costes de instalación y mantenimiento, pasando de un modelo de costes fijo (licencias) a uno de costes variable. Actualizaciones y nuevas funciones inmediatas. Soporte ágil y de mayor rapidez. El usuario/empresa centra sus esfuerzos en su actividad/negocio y no en la elección y mantenimiento de los sistemas. Mayor disponibilidad y seguridad de los datos. Facilidad en el acceso desde cualquier lugar.
Inconvenientes	Al realizarse todo aprovisionamiento del servicio a través de redes, la dependencia de la conexión a internet para acceder a los recursos es crítica. Dado que la gestión de las infraestructuras está en manos de terceros, se crean dependencias fuertes con los proveedores del servicio. Fallos en su gestión pueden dar lugar a problemas graves como la no operabilidad, la pérdida de datos, etc.	Limitación en cuanto a herramientas disponibles (lenguajes, operaciones, etc.), a favor de poder alcanzar un desarrollo sostenible. Alto grado de dependencia en el proveedor. Alta dificultad para migrar de un proveedor a otro.	Nivel de confianza bajo en la seguridad de los datos. Posible incumplimiento de los acuerdos sobre el nivel de servicio prestado. Integración con el resto de aplicaciones de los sistemas locales. La disponibilidad de los datos de la nube no es siempre posible. Alto grado de dependencia en el proveedor.

Figura 3. Tabla comparativa de los niveles de servicios basados en la nube [10].

II.4. Recursos de *Hardware*

El término *hardware* comprende todos los componentes físicos y electrónicos que conforman las diferentes partes que permiten el funcionamiento de un dispositivo tecnológico. Cualquier dispositivo que use electricidad para cumplir una función específica, requiere de un *hardware* básico para funcionar, bien sean elementos básicos como un transformador de corriente hasta equipos más sofisticados como un computador. Cada elemento o periférico que tenga un computador, dispositivos de comunicación, electrodomésticos, dispositivos multimedia, entre otros; consta de un *hardware* característico.

A nivel de computación, el *hardware* se suele definir y clasificar con respecto a grupos de recursos principales:

II.4.1. De cómputo

Es el grupo de recursos y componentes que permiten que un computador funcione, encienda, realice procesos, gestione archivos y documentos en su interior de forma dinámica. Están

constituidos por conjuntos de módulos construidos por subcomponentes electrónicos, mecánicos, fotoquímicos, magnéticos u ópticos. Se encuentran integrados a una placa principal de baquelita que recibe el nombre de placa madre, presente en todos los dispositivos electrónicos modernos. Cada uno tiene una función específica y están estrechamente relacionados entre sí.

El modelo del *hardware* de una computadora personal básica se puede dimensionar y abstraer en tres grupos principales: Procesador, Memorias y Periféricos de entrada y salida (E/S) [12]; conectados y comunicados entre sí por medio de un bus como en la siguiente figura:

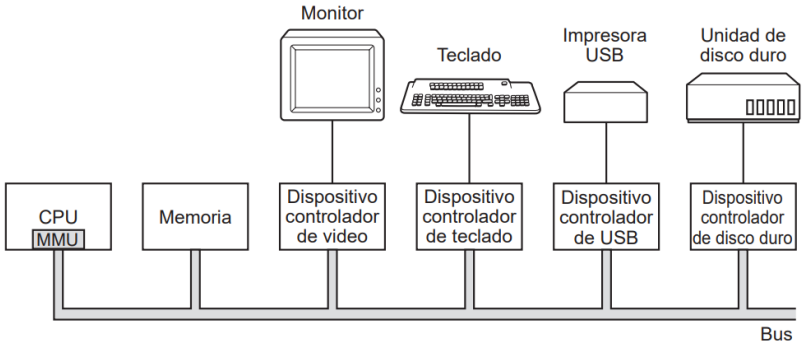


Figura 4. Algunos componentes de una computadora personal simple [12].

Los recursos de cómputo más importantes son:

II.4.1.1. Unidad Central de Procesamiento (CPU)

Se suele definir como el "cerebro de la computadora". La Unidad Central de Procesamiento (del inglés, *Central Processing Unit*, CPU) es el componente principal que obtiene las instrucciones de la memoria principal y las ejecuta a grandes velocidades. Todas las CPU cumplen un ciclo básico que consiste en obtener la primera instrucción de memoria, decodificarla para determinar su tipo y operandos, ejecutarla y después obtener, decodificar y ejecutar las instrucciones subsiguientes. El ciclo se repite hasta que el programa termina. De esta forma se ejecutan los programas [12].

II.4.1.2. Memorias

Son el segundo componente fundamental que debe tener toda computadora en su *hardware*, su función principal es guardar instrucciones, procesos y archivos que la CPU debe ejecutar a altas velocidades. Para optimizar su utilización y gestión, los sistemas de memoria son diseñados y construidos como una jerarquía de capas, en las que cada una de ellas cumple una función

específica. Las capas superiores tienen mayor velocidad, menor capacidad y mayor costo por bit, mientras que las capas inferiores son más lentas, pero de mayor capacidad [12].

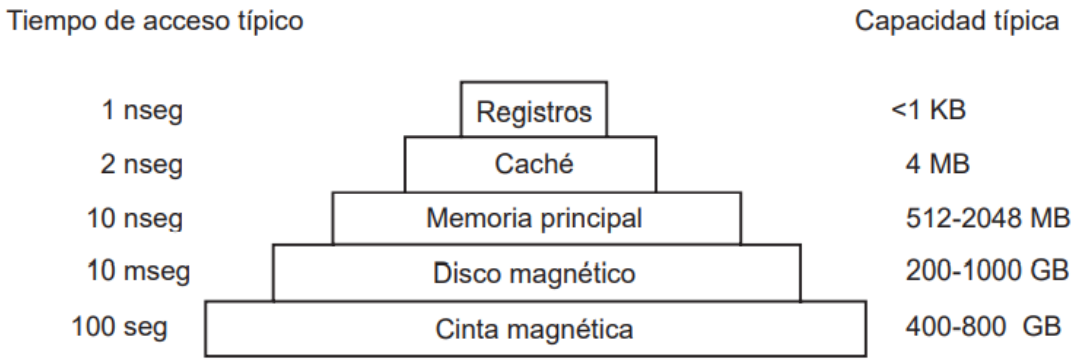


Figura 5. Una común jerarquía de memorias [12].

La capa superior corresponde a los registros internos de la CPU descritos anteriormente, están compuestos de su mismo material y tienen su misma rapidez, por lo tanto, no generan retrasos al momento de utilizarlos [12]. Su capacidad de almacenamiento por lo general es 32×32 bits y 64×64 bits, en CPU de 32 y 64 bits respectivamente.

La segunda capa corresponde a la memoria caché, que el *hardware* controla de forma parcial. Cuando el programa necesita leer una palabra de memoria, el *hardware* de caché comprueba si la línea que se requiere se encuentra en caché. Si es así, la petición de la caché se cumple y no se envía una petición de memoria a través del bus hacia la memoria principal, y se ahorra un tiempo considerable [12].

La tercera capa corresponde a la memoria principal de la computadora, generalmente conocida como Memoria de Acceso Aleatorio (*Random Access Memory* – RAM). Todas las peticiones de la CPU que no se pueden satisfacer desde la memoria caché, se guardan en la memoria principal, esta memoria pierde su contenido cuando no se le suministra energía [12].

Las dos últimas capas corresponden al espacio de almacenamiento de procesos fijos y datos. Dentro de estas capas, muchas computadoras tienen una cantidad de memoria de acceso aleatorio no volátil, que no pierde su contenido cuando no es provista de energía, a diferencia de la RAM.

II.4.2. Medios físicos de transmisión de datos

Son los medios que cumplen el propósito de transmitir bits de una computadora a otra. Cada uno tiene distintas características en cuanto a la velocidad de los datos, retardos, costo y facilidad de instalación y mantenimiento. En la abstracción más general, los medios de transmisión de datos se agrupan en medios guiados (como los cables de cobre, cable coaxial y fibra óptica) y en medios no guiados (transmisión inalámbrica, microondas, satélites).

II.4.2.1. Medios guiados

Son todos los medios que necesitan una guía física para dirigir la señal de información a través de una trayectoria. Esta guía puede ser un cable (como el par de cobre trenzado), una guía de ondas (como un cable coaxial o una fibra óptica) o una línea de transmisión. Por lo general, son de pocas pérdidas y cada uno permite una tasa de datos máxima correspondiente a sus características físicas.

Los medios físicos guiados utilizados en el proyecto fueron construidos utilizando cables de pares de cobre trenzados no blindado (*Unshielded Twisted Pairs*, UTP), que consiste un conjunto de pares de cobre aislados, generalmente de 1 mm de grosor, trenzados entre sí de forma helicoidal. El trenzado se debe a que dos cables paralelos constituyen una antena simple. Cuando se trenzan los cables, las ondas de distintos trenzados se cancelan y el cable irradia con menos efectividad. La señal se transmite como la diferencia en el voltaje entre los dos cables en el par [13].

II.4.3. Tasa de transmisión

Es el término que se utiliza para referirse a la velocidad de transmisión de datos en un canal, esta medida se toma en **bps** (bits por segundo), pudiéndose agrupar en miles, millones y billones (Kbps, Mbps y Gbps) según sea el caso. Los distintos medios de transmisión, tanto los guiados como los no guiados, permiten una tasa de transmisión de datos máxima, generalmente denominada como **capacidad del canal**. Esta capacidad está determinada por el **ancho de banda del canal** y la **relación señal a ruido** de la transmisión. Según Claude Shannon, la capacidad de un canal ruidoso, con ancho de banda *B* [Hz] y una relación señal a ruido de S/N está determinada con la siguiente ecuación [13]:

$$\text{Capacidad del canal [bps]} = \text{BLog}_2(1 + \text{S/N})$$

La tasa de transmisión de datos máxima es distinta por cada canal o medio de transmisión. Un cable de pares trenzados puede transmitir velocidades de hasta 10Gbps (por ejemplo, una interfaz 10 *Gigabit Ethernet*), una fibra óptica puede transmitir a velocidades superiores a los 100Gbps, y una red móvil inalámbrica de quinta generación (5G) podría alcanzar velocidades de hasta 10Gbps.

II.5. Sistemas Operativos

Las computadoras son dispositivos sofisticados que están compuestos por muchos módulos, una computadora moderna consta de uno o más procesadores, una memoria principal, discos, impresoras, un teclado, un ratón, una pantalla o monitor, interfaces de red y otros dispositivos de entrada/salida, es un sistema complejo. Los sistemas operativos son capas de *software* cuyo trabajo es administrar todos y cada uno de los recursos antes mencionados, además de proporcionar al usuario un modelo de computadora mejor, más simple, amigable y pulcro [12].

Los sistemas operativos se ejecutan directamente sobre el *hardware* del computador y proporciona la base para las demás aplicaciones de *software*. El programa con el que los usuarios generalmente interactúan se denomina ***shell***, cuando está basado en texto, y **GUI** (*Graphical User Interface*; Interfaz gráfica de usuario) cuando utiliza elementos gráficos o iconos. El último, en realidad no forma parte del sistema operativo, aunque lo utiliza para llevar a cabo su trabajo y ser más usable para el usuario [12]. El sistema operativo, como pieza fundamental del *software*, se ejecuta en modo ***kernel***, mientras que el resto del *software* del sistema se ejecuta en modo **usuario** [12].

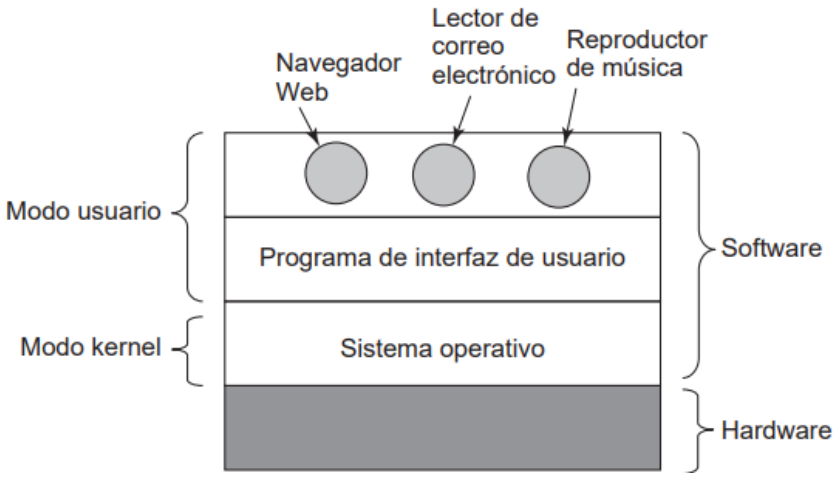


Figura 6. Ubicación y partes de un sistema operativo [12].

El programa de interfaz de usuario, *shell* o GUI, es el nivel más bajo del *software* en modo usuario y permite la ejecución de otros programas, como un navegador WEB, lector de correo electrónico o reproductor de música. Estos programas también utilizan en forma intensiva el sistema operativo [12].

II.5.1. Linux UBUNTU

UBUNTU es un moderno sistema operativo, de código abierto basado en LINUX, un proyecto de núcleo de sistema operativo que evolucionó del sistema operativo UNIX, desarrollado por el estudiante finlandés Linus Torvalds, del que deriva el nombre LINUX. UBUNTU puede ser instalado y utilizado en servidores empresariales, computadoras personales de escritorio o portátiles y para aplicaciones de computación en la nube e IoT (*Internet of Things*; Internet de las cosas). Está patrocinado por CANONICAL, una compañía británica de programación de ordenadores, su misión principal es "*llevar el software libre a la audiencia más amplia*" y su visión consiste en "permitir como plataforma que todos los que deseen consumir y participar de la innovación, puedan hacerlo". "*Permitir que una amplia diversidad de comunidades de código abierto colabore bajo el paraguas de UBUNTU*" [14].

La filosofía de los desarrolladores de UBUNTU se basa en que cada usuario de computadora "*Debe tener la libertad de descargar, ejecutar, copiar, distribuir, estudiar, compartir, cambiar y mejorar su software para cualquier propósito, sin pagar derechos de licencia. Debe poder utilizar su software en el idioma de su elección. Debería poder usar todo el software independientemente de la discapacidad*" [14]. "*Nuestra filosofía se refleja en el software*

que producimos, la forma en que lo distribuimos y nuestros términos de licencia también - Política de derechos de propiedad intelectual. Nuestro objetivo es ser la plataforma que lidere el logro de estos ideales. Trabajamos con el objetivo de que cada pieza de software que pueda necesitar esté disponible bajo una licencia que le brinde esas libertades" [14].

UBUNTU es el sistema operativo sobre el cual se desarrolló la el *snap* de la aplicación *MICROSTACK*, motivo por el cual fue escogido.

II.6. Virtualización de funciones de red

Es el método y tecnología que permite remplazar dispositivos de red físicos ejecutando funciones de red específicas con uno o más programas ejecutando las mismas funciones, mientras se corre sobre un *hardware* de computadora. Por ejemplo, remplazar un firewall físico con uno ejecutándose sobre una máquina virtual [15].

Debido a que la ejecución de aplicaciones puede sobrecargar los elementos de procesamiento de la red, es necesario contar con una infraestructura más robusta, lo cual puede resultar altamente costoso, y poco rentable. La Virtualización de Funciones de Red (*Network Functions Virtualization*, NFV) surgió para abstraer esas funcionalidades mediante *software*, y así reducir las inversiones necesarias para cumplir diferentes funciones de red [7].

II.7. *Software* de Gestión de Infraestructura de Nube

II.7.1. OPENSTACK

Es una plataforma basada en la nube, que controla grandes grupos de cómputo, almacenamiento y recursos de redes a lo largo de un *datacenter*, gestionado y provisionado mediante un conjunto de Interfaces de Programación de Aplicaciones (API). Cuenta con un tablero de gestión que brinda control a los administradores, al mismo tiempo que permite a los usuarios solicitar recursos a través de una interfaz gráfica web [16].

OPENSTACK es de código abierto, usa recursos virtualizados, agrupados, para diseñar, y gestionar nubes privadas y públicas que pueden ser de un solo nodo (*single node*) o de múltiples nodos (*multi-node*). Consta de un conjunto de módulos para cubrir requisitos de diferente índole, necesarios para la provisión de Infraestructura como Servicio [17].

II.7.1.1. Módulos de OPENSTACK

La aplicación de *OPENSTACK* permite desplegar un conjunto de módulos y componentes para brindar distintos tipos de servicios y gestión de recursos por medio de interfaces de programación de aplicaciones. Los módulos de *OPENSTACK* están clasificados en 10 grupos o categorías [18]:

- Cómputo (*Compute*).
- Ciclo de vida del *hardware* (*Hardware Lifecycle*).
- Almacenamiento (*Storage*).
- Redes (*Networking*).
- Servicios compartidos (*Shared Services*).
- Orquestación (*Orchestration*).
- Aprovisionamiento de carga de trabajo (*Workload Provisioning*).
- Ciclo de vida de aplicación (*Application Lifecycle*).
- Servidores Proxy API (*API Proxies*).
- Interfaces Web (*Web frontends*).

De estas categorías, se definen los módulos empleados para el desarrollo del presente proyecto:

- ***KeyStone:*** Es el servicio de gestión de identidades dentro del entorno de la infraestructura de la nube que se implementa. Permite la gestión de dominios, proyectos (que permiten la habilitación de uso multiusuario), cuentas de usuarios y roles [19]. Pertenece a la categoría *Shared Services* (Servicios Compartidos).
- **Glance:** Es el servicio de imágenes de sistemas operativos usados en las instancias. Se encarga de la gestión de catálogos de las imágenes que se usan como plantillas para el aprovisionamiento de las instancias [19]. Pertenece a la categoría *Shared Services* (Servicios Compartidos).
- **Neutron:** Es el servicio de redes TCP/IP. Este módulo se encarga del manejo de redes virtuales, subredes y routers virtuales, así como grupos de seguridad, y otros recursos de red. Proporciona una capa programable por encima de las soluciones comunes de redes definidas

por *software* (SDN), y se integra perfectamente con los demás servicios de *OPENSTACK* [19]. Pertenece a la categoría *Networking* (Redes).

- **Nova:** Es el servicio de cómputo de *OPENSTACK*. Es el responsable de la programación, aprovisionamiento y finalización de las instancias que conforman la nube. Proporciona una capa programable por encima de los hipervisores comunes, integrándose perfectamente con los otros servicios de *OPENSTACK* [19]. Pertenece a la categoría *Compute* (Cómputo).
- **Cinder:** Es el servicio de almacenamiento para las instancias. Se encarga de la programación, creación, asignación y finalización de volúmenes de almacenamiento persistentes. *Cinder* proporciona una capa programable por encima de las soluciones comunes de almacenamiento definido por *software* (*Software Defined Storage*, SDS), y es capaz de integrarse perfectamente con los demás servicios de *OPENSTACK* [19]. Pertenece a la categoría *Storage* (Almacenamiento).

II.7.1.2. Servicios de OPENSTACK

Aparte de los módulos descritos, OPENSTACK ofrece adicionalmente un conjunto de servicios y herramientas para su utilización, las cuales se describen a continuación:

- **Servicios:** Comprende las Interfaces de programación de aplicaciones (APIs) que manejan las funciones de la infraestructura de la nube, como la gestión del catálogo de imágenes, aprovisionamiento de instancias, entre otras [20].
- ***Dashboard* de *OPENSTACK* (*OPENSTACK Dashboard*):** Es una interfaz gráfica de usuario basada en la web, para los servicios de *OPENSTACK* [20].
- **Cliente de *OPENSTACK* (*OPENSTACK Client*):** Proporciona una interfaz de línea de comandos de consola, para los servicios de *OPENSTACK* [20].
- **Bases de datos SQL:** Permite almacenar grupos de registros creados por los servicios de *OPENSTACK* [20].
- **Colas de mensajes (*Message Queues*):** Facilitan la comunicación entre procesos entre los distintos componentes de los módulos de *OPENSTACK* [20].

II.7.2. MICROSTACK

Es una versión simplificada de *OPENSTACK* para el sistema operativo UBUNTU, de sencilla instalación a través de un paquete *snap*, que es la integración de una aplicación y sus

dependencias principales en un mismo paquete. Permite la implementación de nubes sencillas, que van desde aquellas de un nodo, hasta las de múltiples nodos [21], y cuenta con una completa documentación para la utilización mediante la consola de UBUNTU [4].

MICROSTACK realiza la instalación de integrada de los cinco módulos de *OPENSTACK* definidos anteriormente y de sus servicios de utilización, de forma rápida y simple en una máquina.

- **Configuración de Nodo Simple:** Consiste en un nodo de control que proporciona todo lo que el usuario necesite, incluida la función de nodo de cómputo.
- **Configuración Multi-Nodo o *Clustering*:** Permite al usuario instalar *MICROSTACK* en varias máquinas (nodos) y agruparlos para proporcionar un clúster. Un clúster de *MICROSTACK* puede constar de un único nodo de control y de varios nodos de cómputo, tomando en cuenta que un nodo de control puede actuar como un nodo de cómputo.

Una vez implementado uno de los dos tipos de configuraciones, el usuario podrá comenzar a realizar operaciones nativas de *OPENSTACK* creando pares de claves, redes, y tipos de nubes [4]. A efectos del presente trabajo, primero se realizaron pruebas con el modo de sólo nodo de forma local, y posteriormente, se implementó la nube en el laboratorio de telemática en modalidad de múltiples nodos.

II.7.3. Máquinas virtuales

Se les denomina máquinas virtuales a la virtualización completa de una computadora que proporcionan la misma funcionalidad que una computadora física, incluyendo la ejecución de aplicaciones y sistemas operativos. Consisten en archivos informáticos que se ejecutan sobre una computadora física, comportándose como computadoras independientes [22]. El proceso que corre en la computadora física que ejecuta la creación y ejecución de máquinas virtuales, se denomina hipervisor o monitor de máquinas virtuales [23].

II.7.4. Instancias

En el *cloud computing*, se le llama de esta forma a abstracciones de computadoras o máquinas virtuales, en referencia al significado informático de la instanciación, en programación orientada a objetos.

II.7.5. Flavors

Son especificaciones definidas de *hardware*, conformadas principalmente por la memoria principal (RAM), el almacenamiento y la potencia de cómputo, definidas por defecto al momento de implementar instancias en la nube, el usuario de la misma puede seleccionar entre un conjunto de *flavors* definidos o incluso crear propios de acuerdo a sus características personalizadas.

CAPÍTULO III. MARCO METODOLÓGICO

Este capítulo muestra la planificación metodológica de cada una de las actividades que se deben cumplir para la consecución satisfactoria de los objetivos del proyecto y obtener resultados satisfactorios. Busca resumir y agrupar los objetivos específicos del mismo en un conjunto de fases de desarrollo que van desde la investigación teórica de todos los temas relacionados al proyecto, hasta la implementación final en el producto final. Cada una de las fases repercute directamente en las fases anteriores y posteriores.

La planificación inicial ha sufrido algunas modificaciones en los plazos de duración de algunas de las fases, debido a que la misma se trató de una estimación inicial llevada a cabo al realizar la propuesta y aún no se había hecho frente a las diferentes dificultades que surgen al emplear herramientas de tecnología de vanguardia, como es el caso del *software MICROSTACK*, tal como la escasa disponibilidad de información sobre la misma.

El presente proyecto, tomando sus características en cuenta y consideración, entra en la categoría de **proyecto especial**, debido a que lleva a cabo la creación de una herramienta que sentará las bases para la solución de un problema, la cual busca dar respuesta a necesidades e intereses de tipo cultural [24].

El proyecto requiere de una elaboración estructurada dividida en cinco fases en desarrollo para cumplir bloques de objetivos específicos, las cuales son:

III.1. Fase I: Investigación teórica documental.

Esta fase consiste en la indagación en los conocimientos y fundamentos teóricos necesarios para la elaboración e implementación del proyecto. Es una fase continua y constante porque se requiere de fundamentos de cómputo en la nube, *OPENSTACK*, sistema operativo UBUNTU, *MICROSTACK*, *hardware*, e Infraestructura como Servicio (IaaS). Las acciones que se han llevado a cabo son las siguientes:

- Investigación sobre los conceptos asociados al levantamiento nubes para brindar Infraestructura como Servicio (IaaS).

- Indagación en antecedentes de levantamientos de nubes con *OPENSTACK* y *MICROSTACK*, en general, y sus aplicaciones en el área académica.
- Recopilación de guías de uso de *MICROSTACK*, y de sus comandos de consola.
- Determinación de las características de *hardware* de las computadoras del laboratorio de telemática.
- Definición de los parámetros a medir para evaluar rendimiento de plataformas de nube, y de computadora.

III.2. Fase II: Configuración de las Herramientas.

Esta fase inició la interacción con los entornos que hicieron posible la realización y desarrollo del proyecto. Se descargaron los programas necesarios para la implementación de la nube y se realizaron las configuraciones necesarias para brindar el servicio de infraestructura. Inicialmente, estas herramientas fueron instaladas en las computadoras de los autores del presente trabajo, con el fin de familiarizarse con el entorno de *MICROSTACK*, previo a la instalación en las computadoras del laboratorio. Las acciones realizadas durante esta fase fueron:

- Instalación y configuración de la versión 20.04.3 LTS del sistema operativo UBUNTU en computadoras locales, creación de particiones de disco duro, y asignación de recursos de memoria y procesamiento.
- Ajustes de configuración y rendimiento por medio de la consola de UBUNTU. Gestión de la conectividad a redes locales, y a redes externas como Internet.
- Descarga e instalación de la herramienta *MICROSTACK* desde la consola de UBUNTU. Visualización de tutoriales de configuración.

III.3. Fase III: Aprendizaje y experimentación local de *MICROSTACK*.

En esta fase se experimentó con las distintas opciones de servicio que se pueden ofrecer en la nube mediante la utilización del *software MICROSTACK*. Se ha estudiado de la documentación existente, comandos y módulos para cada servicio que ofrece, se han determinado cuáles de estos son los óptimos para el desarrollo del proyecto, y se instalaron en las computadoras de los autores del presente trabajo. Las actividades realizadas en el desarrollo de esta fase fueron:

- Lectura y estudio de la documentación de *OPENSTACK* y de *MICROSTACK* para el conocimiento de sus características, módulos y servicios ofrecidos.
- Estudio de los módulos de *MICROSTACK* necesarios para la implementación de la nube.
- Descarga e instalación los paquetes de *MICROSTACK* que permitieron ofrecer los recursos de *hardware* por medio de la nube.
- Habilitación las funciones de Infraestructura como Servicio, y realizar pruebas con otras computadoras de la misma red de área local. Medición el rendimiento de la conexión remota a la infraestructura con base en parámetros definidos.

III.4. Fase IV: Montaje de la nube en el Laboratorio de Telemática.

En esta fase, se realizó la instalación y levantamiento de la nube en las computadoras del laboratorio de telemática en la universidad. Se llevó a cabo el levantamiento de cinco nodos, correspondientes a cinco computadoras del laboratorio; cuatro de ellos, son nodos de cómputo, y, uno es el nodo de control encargado de gestionar los recursos las computadoras que conformen nube.

Dentro de esta fase, se desarrollaron las siguientes acciones:

- Instalación de *MICROSTACK* en las computadoras del laboratorio de telemática.
- Configuración de los módulos de *MICROSTACK* con los que se habrá experimentado en la fase III.
- Levantamiento de la infraestructura como servicio en las cinco computadoras que conformarán la nube y configuración del nodo de control.
- Creación de instancias de prueba.

III.5. Fase V: Realización de pruebas de rendimiento.

Ya estando implementada la nube en el laboratorio de telemática, en esta fase se definieron los parámetros para evaluar el rendimiento y la calidad del servicio. Se crearon instancias, se realizaron actividades solicitando el servicio durante sesiones prolongadas y se escogieron parámetros y criterios para la evaluación de la plataforma. Se evaluó el rendimiento, y se comparó con el rendimiento de máquinas virtuales.

Las actividades realizadas en el desarrollo de esta fase son:

- Determinación de tres parámetros de medición de rendimiento de instancias y máquinas virtuales de la nube.
- Comparación de rendimiento de instancias en la nube con respecto de las máquinas virtuales manejadas con *VirtualBox*, para determinar cuál de las dos opciones ofrece el mejor desempeño.

La duración de cada una de las fases realizadas, se presentan en el siguiente diagrama:

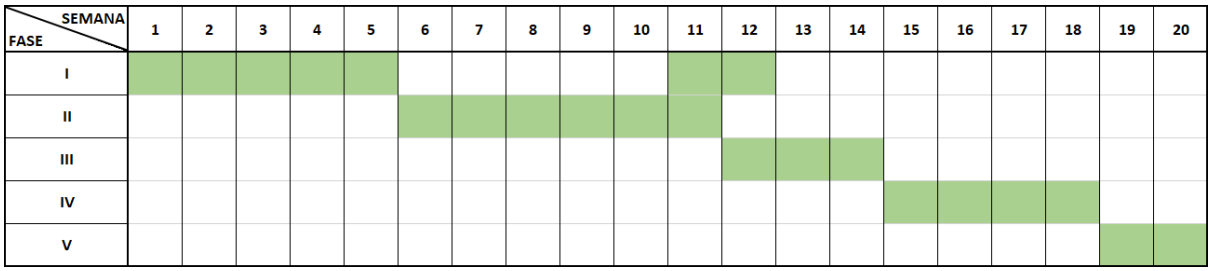


Figura 7. Duración de las fases del proyecto.

CAPÍTULO IV. DESARROLLO

El presente capítulo recopila la ejecución de las actividades que permitieron llevar a cabo este proyecto, desde su análisis teórico hasta su implementación. Las actividades descritas se presentan relacionadas y agrupadas en las mismas fases descritas en la planificación metodológica plasmada el capítulo anterior, pudiéndose visualizar su contribución al proyecto en cada uno de sus aspectos.

Dado que la estructura del proyecto no es estrictamente secuencial, algunas partes del desarrollo no contribuyen únicamente a sus respectivas fases, sino que contribuyen en otros aspectos presentes en toda la realización del proyecto, como es el caso de la investigación de fundamentos teóricos, en donde la extensión de cada uno de los temas fue descrita en marco teórico del presente tomo, en el capítulo II del mismo.

IV.1. Fase de investigación teórica documental

Esta fase comenzó con la búsqueda de los conceptos teóricos relacionados a la computación en la nube o *cloud computing*, como tema fundamental en el que se basa el proyecto. La infraestructura como servicio (IaaS) es la piedra angular del proyecto, siendo la principal bondad que ofrece el prototipo nube desplegado en el laboratorio de telemática. No obstante, la infraestructura como servicio comprende numerosos elementos que fueron estudiados con el propósito de comprender su estructura, y poder llevar a cabo el proyecto.

Los aspectos teóricos del presente proyecto se basan en numerosos macro grupos de temas y conceptos que engloban múltiples subpartes que se complementan entre sí, donde la infraestructura como servicio es el principal objeto de estudio.

Para conocer mejor el concepto de esta, se comenzó por el estudio global de la computación en la nube, comprendiéndolo como un cambio de paradigma que permite otorgar a los beneficiarios un conjunto de recursos tecnológicos de sistemas computacionales de forma remota utilizando una red que puede ser Internet. Estos recursos son compartidos desde una infraestructura diseñada y optimizada para ello, para que el usuario no deba poseer el *hardware* o el *software* físicamente en su misma ubicación, sino que pueda acceder a los mismos por medio de la red.

Posteriormente, se exploraron los principales modelos de servicio de la computación en la nube: El *Software* como servicio (SaaS), Plataforma como servicio (PaaS), e Infraestructura como Servicio (IaaS), estableciendo comparaciones y similitudes (ver Figura 2), donde la más completa y compleja es la última, dado que permite la compartición de recursos de *hardware* principal como: memorias principales, poder de cómputo y recursos de red como la tasa de datos; y recursos *software* como sistemas operativos y aplicaciones.

Para desplegar infraestructuras de computación basadas en la nube, es necesario utilizar múltiples herramientas y aplicaciones diseñadas exclusivamente para este fin. Se utilizó la aplicación *OPENSTACK*, un gestor y controlador de recursos de cómputo por medio de interfaces de programación de aplicaciones (API), el cual fue estudiado y desglosado a profundidad en el marco teórico del tomo.

Es posible realizar el despliegue de la nube en múltiples sistemas operativos; sin embargo, como *MICROSTACK* fue diseñado para el sistema operativo UBUNTU, se utilizó el mismo en su versión 20.04.3 LTS, requerida por *MICROSTACK*. El sistema operativo UBUNTU, de igual forma, fue estudiado a profundidad desde sus orígenes; en dicho estudio, se aprendió a interactuar con la computadora por medio de la consola de comandos, primordial para la implementación de *MICROSTACK*.

Se estudiaron las funcionalidades que permiten realizar despliegues de nubes utilizando múltiples computadoras pertenecientes a una misma red de área local, como es el caso del laboratorio de telemática de la universidad, se recopilaron distintos tutoriales y guías para el desarrollador de la nube [4]. Entre las que se destacan los dos tipos de despliegue, de un solo nodo (*sigle-node*) [25], y de múltiples nodos (*multi-node*) [26], ambos implementados en los ensayos de prueba y la implementación en el laboratorio respectivamente.

IV.2. Fase de configuración de las herramientas

En esta fase se recopilaron, adquirieron y configuraron los distintos elementos y herramientas con las cuales se llevó a cabo el proyecto, con el fin de realizar un ensayo de prueba desplegando una “micro nube” en una computadora local, perteneciente a los autores del presente

trabajo, en el que se logró comprender a profundidad el funcionamiento de UBUNTU, *MICROSTACK* y sus componentes.

Previo a la instalación de UBUNTU en su versión 20.04.3 LTS (*Long Term Support*), se realizó una partición del espacio de almacenamiento en el disco duro principal de la computadora, para dar lugar a todos los archivos y programas que conforman al sistema operativo, que cumple con los requisitos mínimos de *hardware* para su funcionamiento, establecidos en su página web oficial [4], los cuales se listan a continuación:

- Procesador de múltiples núcleos.
- 8GB de memoria principal (RAM).
- 100GB de memoria secundaria (almacenamiento).

Posteriormente, se realizaron pruebas de conectividad y funcionamiento a través de la interfaz de comandos, empleando aquellos recopilados y estudiados en la fase I del proyecto, los cuales permitieron verificar la correcta operatividad del sistema antes de llevar a cabo el despliegue de la nube. Dichas pruebas serán descritas a continuación:

IV.2.1. Pruebas de conectividad en UBUNTU

En primer lugar, se comprobó la operatividad de las interfaces de red y la conectividad a redes locales y a Internet; para tal fin, se ejecutaron comandos de red en la consola (*Shell*) de Interfaz de Líneas de Comandos (*Command-Line Interface*, CLI) de UBUNTU. Se ejecutaron comandos de visualización de estatus de las interfaces de red, información de redes conectadas, y envío y recepción de paquetes de red por medio del protocolo ICMP, pruebas correspondientes a los comandos "*ip address*" (o "*ip link*") y *ping*, respectivamente.

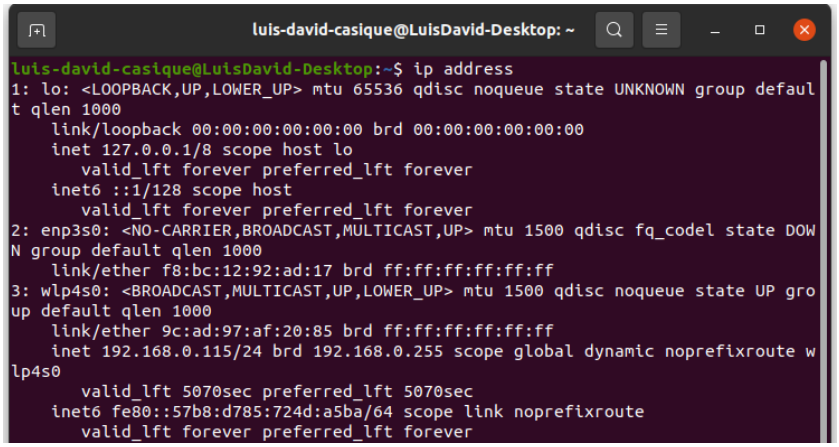


Figura 8. Visualización de Interfaces de red con el comando "ip address".

Al ejecutar el comando "*ip address*", se puede visualizar que UBUNTU devuelve el estatus de las interfaces de red de la computadora local en la que se instaló el S.O. La cantidad de interfaces varía dependiendo del equipo; en este caso, se cuenta con tres.

Como puede observarse en la consola, se muestra en todo momento tanto el nombre del usuario o *user name* que opera el nodo ("luis-david-casique") y también el nombre dado al equipo o *hostname* al momento de la instalación del sistema UBUNTU ("LuisDavid-Desktop"). Todas las capturas que tengan el *hostname* LuisDavid-Desktop, fueron tomadas desde el nodo de control.

Posteriormente, se comprobó la conectividad mediante protocolo ICMP, por medio de los comandos que, por cuestiones de comodidad visual del presente tomo, serán mostrados en su fuente tipográfica original de la propia consola de UBUNTU.

`$ ping 192.168.0.1`. Puerta de enlace de la red del nodo de la nube.

`$ ping www.google.com`. Dirección web de Google, para comprobar conexión a Internet.

Ambas conexiones resultaron satisfactorias, por lo que se continuó con la instalación de *MICROSTACK*.

IV.2.2. Instalación de MICROSTACK

En este paso, se llevó a cabo la instalación de *MICROSTACK* en su última versión disponible para el momento (242). El proceso de instalación está descrito a continuación:

IV.2.2.1. Descarga del paquete snap

Para iniciar la descarga del paquete *snap* de *MICROSTACK*, se ejecutó el comando:

```
~$ sudo snap install microstack --beta
```

Nótese que es un comando que exige el permiso de super usuario, indicado con el prefijo "sudo" del comando de instalación.

Una vez que la descarga e instalación finalizan de manera exitosa, aparece el mensaje "`Se ha instalado microstack (beta) ussuri por Canonical ✓`". La instalación se puede comprobar si se intenta ejecutar el mismo comando nuevamente, donde aparece el siguiente mensaje ilustrado en la **Figura 9**.

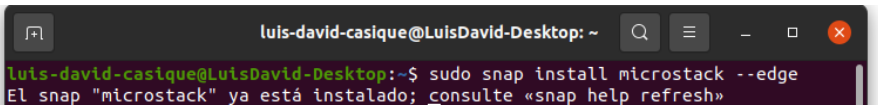


Figura 9. Comprobación de instalación del *snap MICROSTACK*.

En la siguiente figura, se pueden observar los servicios de *MICROSTACK* inactivos.

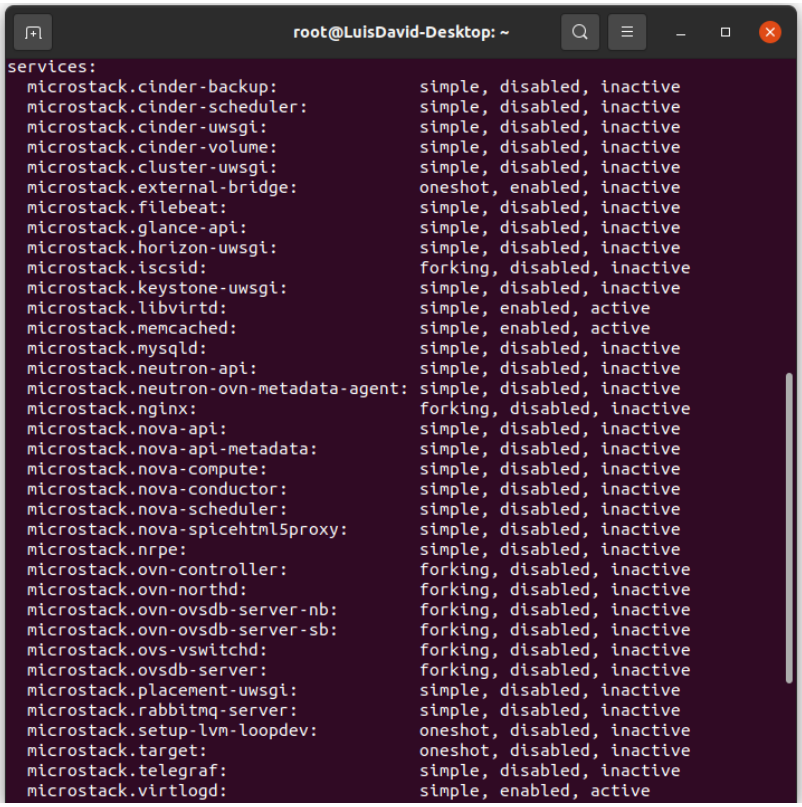


Figura 10. Módulos y servicios de *MICROSTACK*, desactivados.

IV.3. Fase de aprendizaje y experimentación local con *MICROSTACK*

En la presente fase, comenzó con la realización de análisis sobre la documentación de *MICROSTACK* y *OPENSTACK*, con el propósito de tener una comprensión de su lógica operativa, a través de la ejecución de comandos en la consola y con la interfaz gráfica de usuario web. Se aplicaron las nociones adquiridas en los procesos de implementación, configuración y gestión de la plataforma de *MICROSTACK* para el despliegue de la "micro nube" local de prueba descrita e iniciada en la fase anterior.

Posteriormente, se continuó el proceso de creación de la "micro nube" de prueba, continuando en el punto en el que culminó la fase II del proyecto. Se procedió a realizar la inicialización de la aplicación *MICROSTACK*, inicialmente en la modalidad de un solo nodo (*single-node*), debido a que, para efectos de esta prueba, se utilizó en primer lugar la infraestructura de una computadora local, y una vez finalizadas las pruebas, se incorporó una segunda computadora a la nube para desempeñar la función de nodo de cómputo, la implementación en el laboratorio. Este proceso será descrito a continuación:

IV.3.1. Inicialización de MICROSTACK

Cuando se adquirió el *snap* de *MICROSTACK*, hizo falta su activación e inicialización para poder utilizarlo como gestor de recursos de la computadora, para ello, se ejecutó el comando:

```
$ microstack init --auto –control
```

Este comando comenzó la habilitación de sus módulos, ajustes de variables de entorno, configuración de ajustes de clústeres, generación de certificados, entre otros procesos:

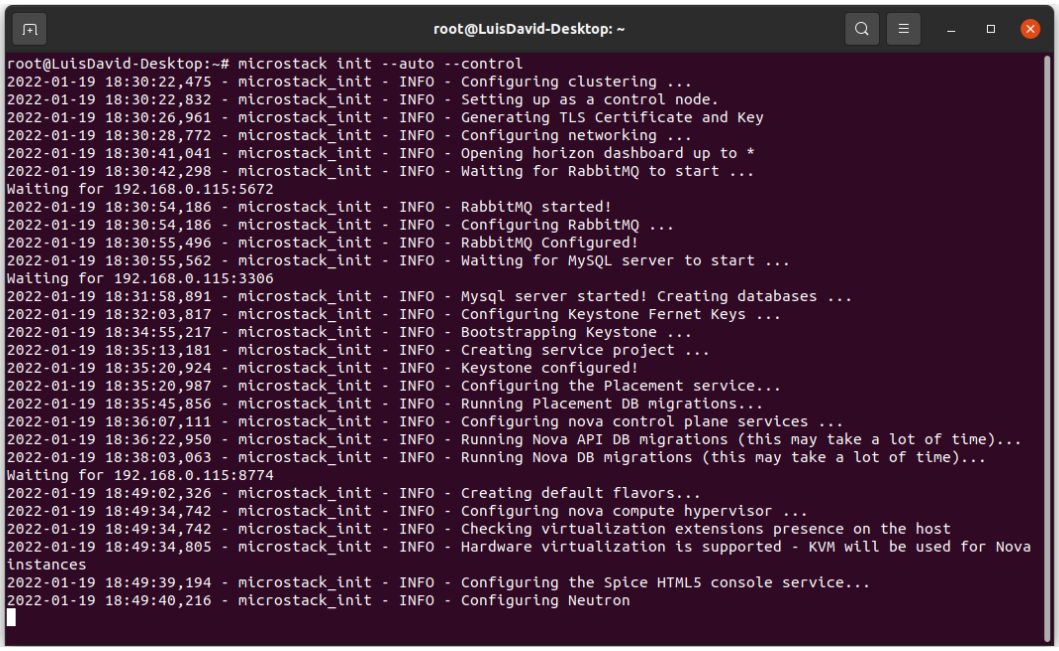


Figura 11. Inicialización de *MICROSTACK*.

Una vez que ha finalizado la inicialización, el *shell* ha devuelto el mensaje "`Complete. Marked microstack as initialized!`":

IV.3.2. Interacción con OPENSTACK

Una vez que la inicialización de *MICROSTACK* ha finalizado, se han obtenido los módulos de *OPENSTACK* que permiten gestionar el *hardware* del equipo por medio de su aplicación web. Para ello, fue necesario solicitar las credenciales para el perfil de administrador del gestor de *OPENSTACK*, se realizó esta solicitud con la ejecución del siguiente comando:

```
# snap get microstack config.credentials.keystone-password
```

Nótese que este último comando, viene precedido por un símbolo de asterisco (#) en vez de un dólar ($), esto indica que fue ejecutado posteriormente a entrar al modo de super usuario global, por medio del comando "`sudo -i`", que permite que todas los procesos y comandos se ejecuten bajo ese modo, sin la necesidad de tipear "`sudo`". Nótese también que el comando tiene entre sus caracteres el término "`keystone`", el cual es el módulo de gestión de identidades de *MICROSTACK*. Este comando anterior, devolvió como salida la contraseña alfanumérica (puede ser cambiada posteriormente), para el usuario administrador denominado "*admin*", que es creado por defecto por *MICROSTACK*:

Luego de obtener estas credenciales, ya es posible interactuar con *OPENSTACK* por medio de su aplicación web, a la cual se accedió ingresando la dirección IP 10.20.20.1 en el navegador web del sistema.

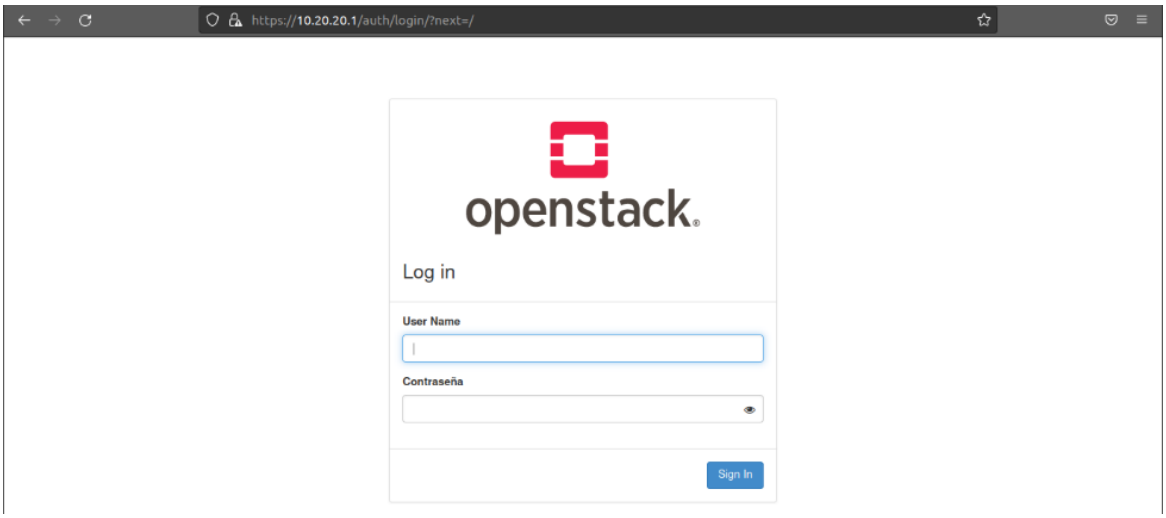


Figura 12. Inicio de sesión en el *dashboard OPENSTACK*.

Se ingresaron las credenciales del perfil de administrador, obtenidas en el paso anterior, con ello, se concedió acceso al gestor de los recursos de *hardware* destinados de la computadora, un *dashboard* (tablero de mandos) que muestra en conjunto de categorías a las instancias virtuales creadas con la proporción de *hardware* correspondiente a cada una:

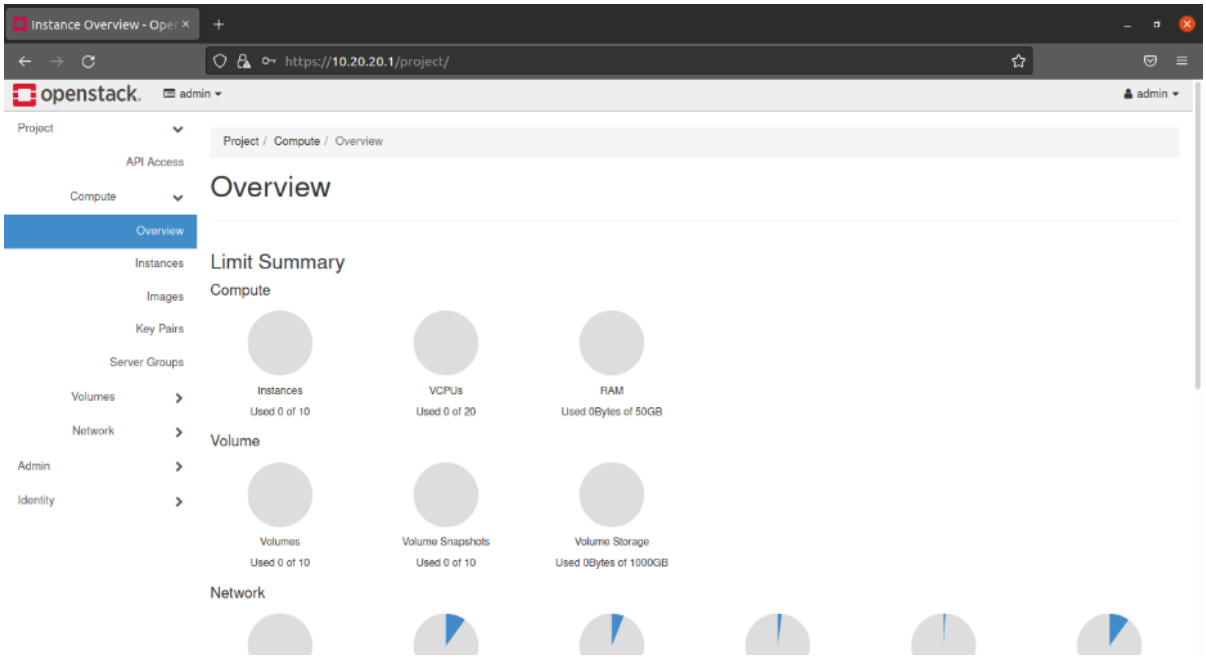


Figura 13. *Dashboard* inicial de *OPENSTACK*.

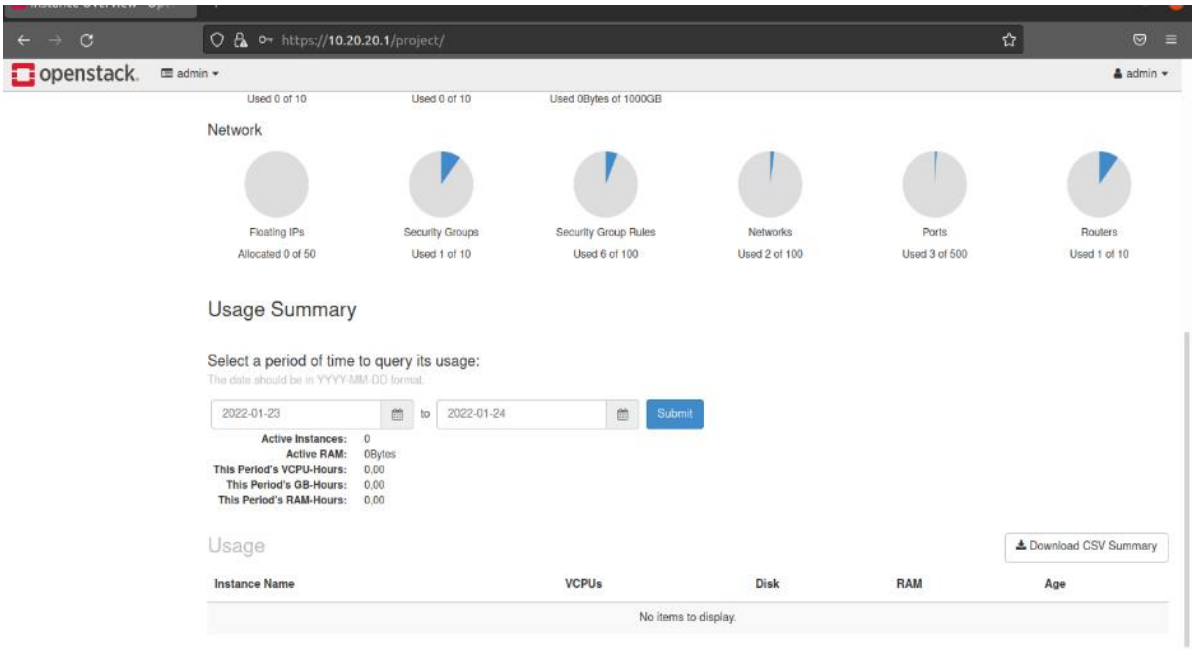


Figura 14. Resumen de uso del *hardware* compartido.

IV.3.2.1. Creación de una instancia rápida de prueba

Por defecto, *MICROSTACK* tiene disponible un conjunto limitado de funciones a través de la consola; adicionalmente, también cuenta con una imagen inicial del sistema operativo **CirrOS**, una versión ligera de UBUNTU sin interfaz gráfica y con opciones limitadas. Una vez que se inicializa *MICROSTACK*, se pueden crear instancias con la imagen de CirrOS a través del grupo de comandos `microstack launch`:

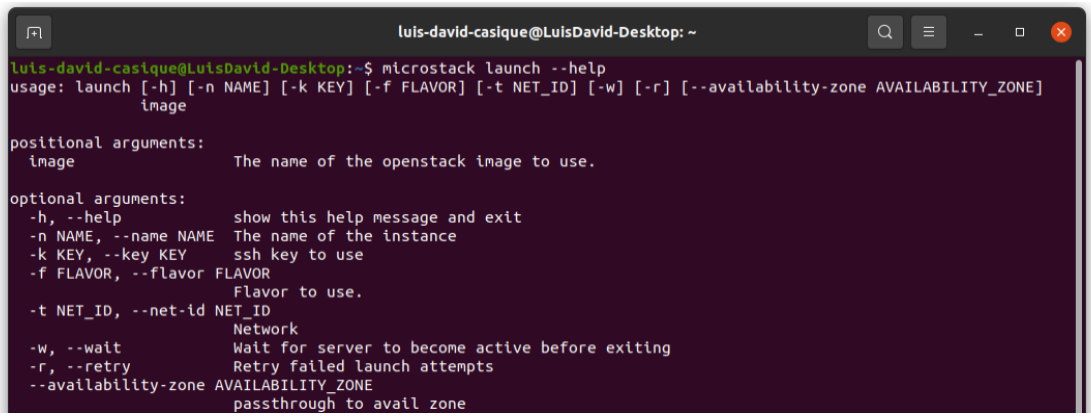


Figura 15. Opciones del comando de lanzamiento de instancias y creación de la instancia "*test*".

El comando específico para la creación de una instancia con CirrOS, es:

```
~$ microstack launch cirros --name test
```

Este comando creó una instancia simple de CirrOS cuyo nombre es "*test*". La ejecución del mismo proporciona como resultado el comando que se necesita para acceder a la instancia vía SSH (*Secure Shell*), por medio de la llave de acceso, un archivo de extensión ".pem" (Correo de Privacidad Mejorada, del inglés *Privacy-Enhanced Electronic Mail*).

```
/home/luis-david-casique/snap/microstack/common/.ssh/id_microstack
cirros@10.20.20.162
```

Para que la infraestructura de la nube de *OPENSTACK* pueda comunicarse con las instancias virtuales y con el nodo de control, reserva un grupo de direcciones IP flotantes del bloque 10.20.20.0/24, que luego son traducidas por medio del protocolo NAT (*Network Address Translation*) y asignadas a cada una de las instancias para su acceso remoto fuera de la nube. En el caso de la instancia "test", su dirección IP flotante es la 10.20.20.162. Para el acceso SSH a la instancia, se ejecutó el siguiente comando, que incluye la llave de acceso SSH devuelta:

```
$ ssh -i /home/luis-david-casique/snap/microstack/common/.ssh/id_microstack
cirros@10.20.20.162
```

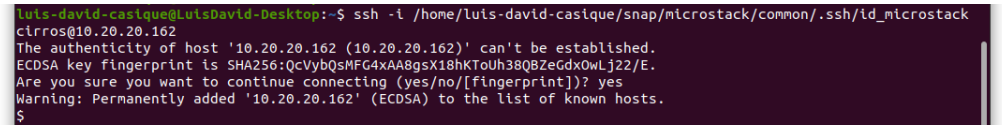


Figura 16. Acceso remoto a la instancia de prueba "*test*".

El acceso a la instancia fue exitoso. No obstante, como se mencionó anteriormente, cirrOS es muy limitado en cuanto a funcionalidades, por lo que las pruebas que se pudieron realizar fueron pobres, como, por ejemplo, visualizar la hora y el tiempo de actividad de la instancia, así como el estatus de sus interfaces de red:

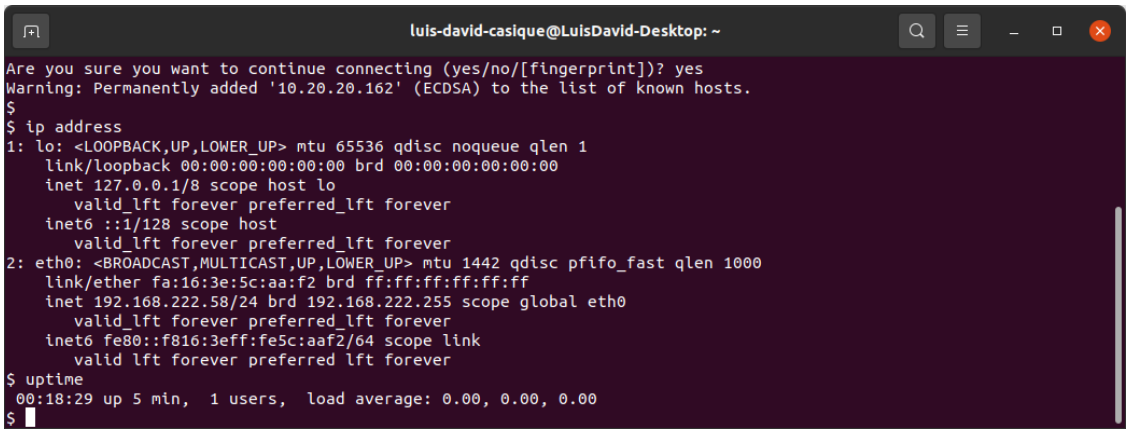


Figura 17. Interfaces de red, tiempo de uso y carga de la instancia "*test*".

Para efectos de la presente prueba, se creó otra instancia adicional denominada “segundo-test”. Ambas, se pueden visualizar por medio del comando:

```
$ microstack.openstack server list --all-projects
```

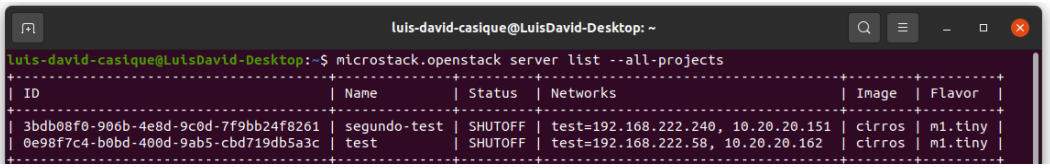


Figura 18. Visualización de instancias iniciales “*test*” y “segundo-test” de prueba.

No obstante, con los ajustes por defecto de *MICROSTACK*, se puede hacer poco más de lo hecho hasta el momento. Para ampliar el abanico de opciones para mejorar la infraestructura de las nubes que se deseen crear y gestionar, se ajustó y acondicionó el entorno de *OPENSTACK*.

Lo primero que se realizó en este proceso fue la instalación del cliente completo de *OPENSTACK* (*OPENSTACK Clients*), descrito en el marco teórico del presente avance. Para ello, se ejecutó el siguiente comando:

```
$ sudo snap install openstackclients
```

Luego, se descargó un archivo de guión de recursos (*Resource script*, RC) de extensión “.sh” a través del *dashboard* principal de *OPENSTACK*. Se ingresó al menú desplegable del administrador en la parte superior derecha del *dashboard*, y desde ahí se descargó el *script* con el nombre **“`admin-openrc.sh`”**, correspondiente al usuario administrador [20].

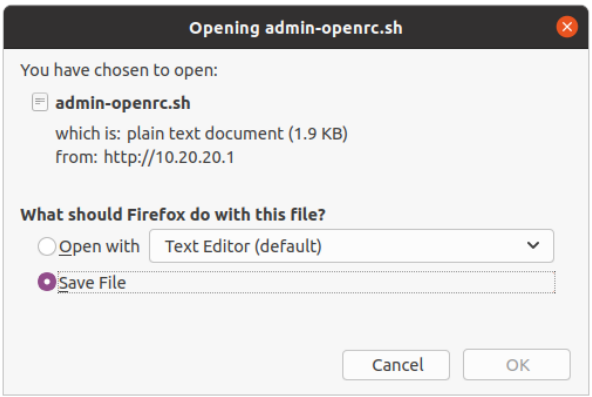


Figura 19. Descarga del archivo con las variables de entorno del usuario *admin* para cliente de *OPENSTACK*.

Antes de utilizar los comandos del *OPENSTACK clients*, se cargaron un conjunto de parámetros y variables de entorno que acceden a los recursos de la nube, definidos en el archivo

script RC descargado en el paso anterior. Para establecer y dichas variables se ejecutó el siguiente comando en la consola:

```
$ source ~/Descargas/admin-openrc.sh
```

Al cargarse los parámetros del usuario administrador, el comando introducido, solicita la contraseña de acceso a *OPENSTACK* bajo el perfil del usuario administrador, la cual fue ingresada. Al realizar este paso, se puede utilizar el cliente de *OPENSTACK* en conjunto con el *dashboard* para gestionar la nube.

IV.3.3. Acondicionamiento del entorno

Luego de configurar las herramientas anteriores, se procedió a crear un entorno personalizado para el desarrollo de las pruebas. Se creó un dominio propio, cuentas de usuario y roles para cada uno [27]; por medio de los pasos descritos a continuación:

IV.3.3.1. Creación de dominio

Se creó un dominio aparte del dominio *default*, denominado "*mydomain*", con un usuario administrador; por medio de los siguientes comandos:

```
$ openstack --insecure domain create --description "My domain" mydomain
```

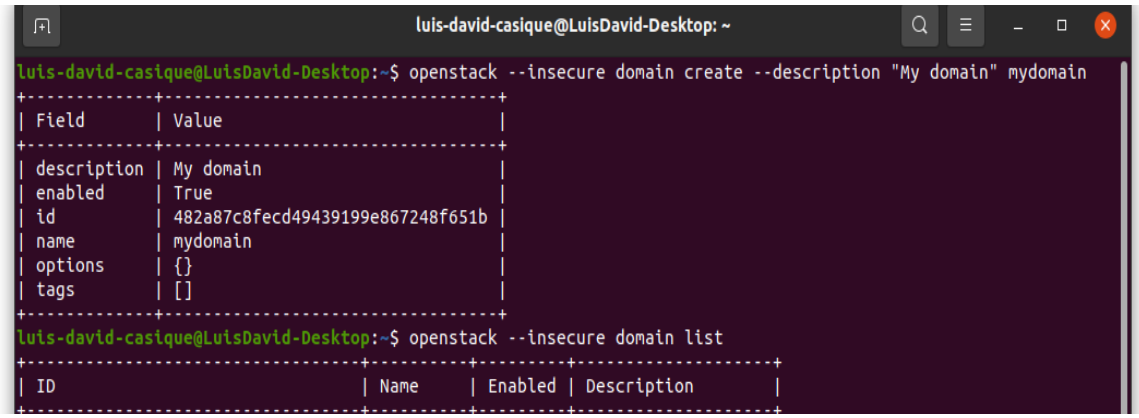


Figura 20. Creación y visualización del dominio "*mydomain*".

Para este dominio, se creó un usuario administrador de nombre "*admin*", se le asignó el rol de administrador y una contraseña de acceso distinta a la anterior, se recomienda que la misma sea suficientemente extensa y robusta. Por medio de los siguientes comandos:

```
$ openstack --insecure user create --domain mydomain --password admin admin
```

```
$ openstack --insecure role add --domain mydomain --user-domain mydomain --
user admin admin
```

IV.3.3.2. Habilitación de soporte multi dominio

Por defecto, *OPENSTACK* tiene deshabilitado el soporte multidominio, el mismo fue habilitado por medio de los siguientes comandos:

```
$ sudo bash -c 'cat >
/var/snap/microstack/common/etc/horizon/local_settings.d/_10_enable_multidomai
n_support.py' << EOF

OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True

EOF

$ sudo snap restart MICROSTACK.horizon-uwsgi
```

Una vez hecho esto, se cerraron las sesiones en el *dashboard* de *OPENSTACK*, al momento de volver a iniciar, se pudo observar un tercer campo para la introducción de credenciales, correspondiente al ingreso del dominio en el que se desea trabajar.

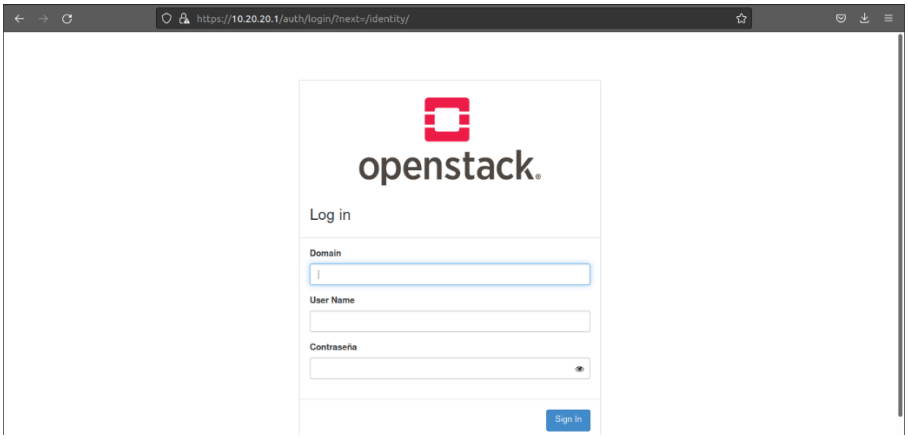


Figura 21. Inicio de sesión en el *dashboard* de *OPENSTACK*, nótese el campo adicional para especificar el dominio.

IV.3.3.3. Creación de proyecto

Luego, aparte del proyecto por defecto (*default*), se creó un proyecto personalizado de nombre "myproject" para el despliegue de la nube y las pruebas correspondientes; esta acción se puede realizar por medio del cliente de *OPENSTACK* y también por medio del *dashboard*, por cuestión de comodidad y bondades de la interfaz gráfica de usuario, se decidió realizarla en el *dashboard*, a través de las siguientes opciones:

Identity → Projects → Create Project

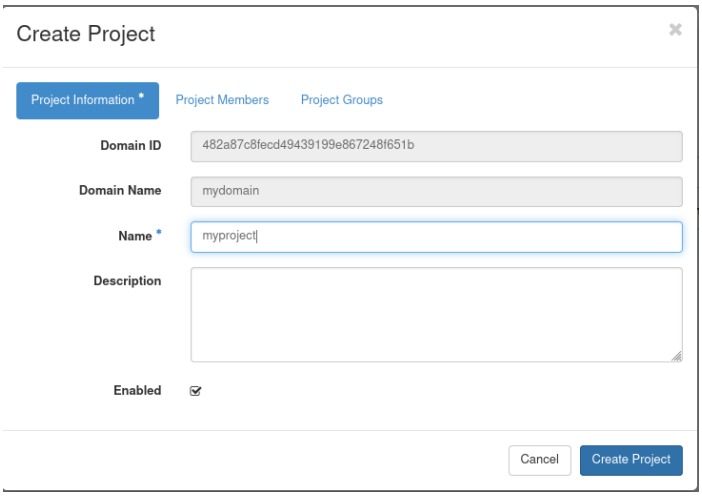


Figura 22. Creación del proyecto "*myproject*".

Luego de que fueron creados, fue posible visualizar el nuevo proyecto en el *dashboard* y en el cliente de *OPENSTACK*, por medio del comando:

```
$ openstack --insecure project list --domain mydomain
```

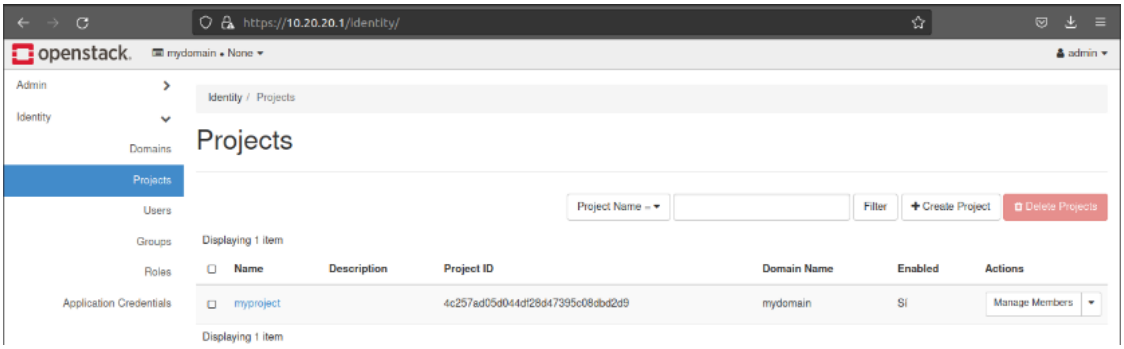


Figura 23. Visualización de "*myproject*" en el *dashboard*.

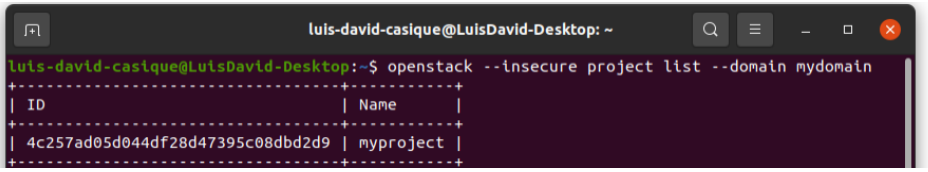


Figura 24. Visualización de "*myproject*" en el cliente de *OPENSTACK*.

IV.3.3.4. Creación de nuevo usuario y grupo

Para complementar la identidad del entorno que se ha estado desarrollado, se creó una segunda cuenta de usuario denominada "*myuser*", dentro de "*mydomain*" y "*myproject*". Adicionalmente, se creó un grupo de usuarios denominado "*mygroup*", en el que "*myuser*" fue

incluido con el rol de miembro general. Este proceso se realizó en el *dashboard* de *OPENSTACK*, a través de los siguientes pasos [27]:

Identity → *Users* → *Create User*

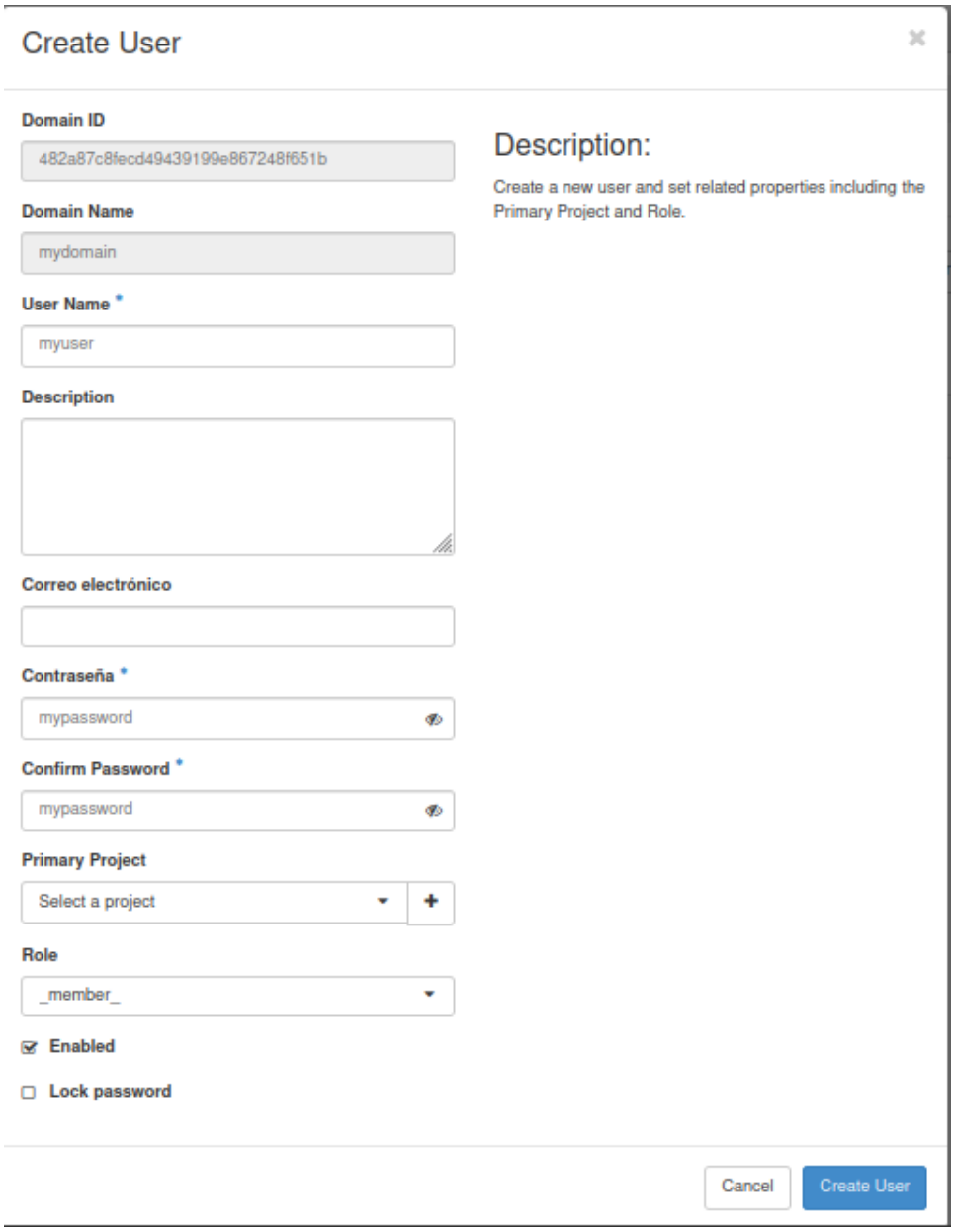


Figura 25. Creación de usuario "*myuser*", del dominio "*mydomain*". Con su respectiva contraseña y rol de miembro.

Es posible visualizar el nuevo usuario desde el *dashboard* de *OPENSTACK*, en la dirección:

Identity → *Users*

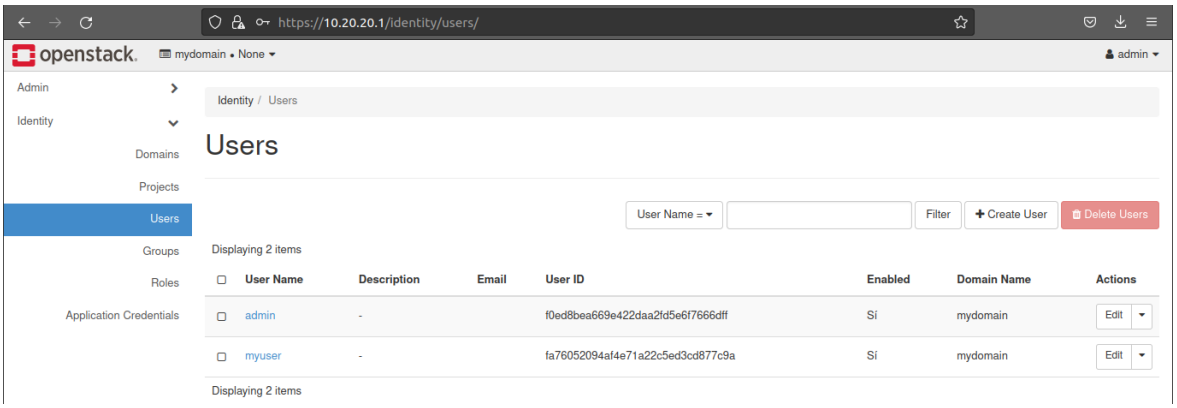


Figura 26. Visualización de "*myuser*" en el *dashboard* de *OPENSTACK*.

Y de igual forma, en el cliente de consola, por medio del comando:

```
$ openstack --insecure user list --domain mydomain
```

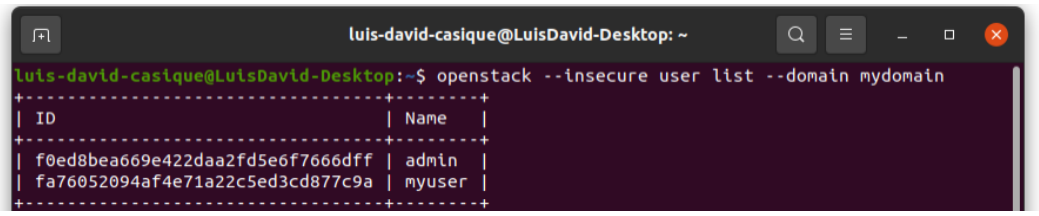


Figura 27. Visualización de "*myuser*" en el cliente de consola de *OPENSTACK*.

La creación del grupo "*mygroup*", se realizó de igual forma, en el *dashboard* de *OPENSTACK*, a través de los siguientes pasos:

Identity → *Groups* → *Create Group*

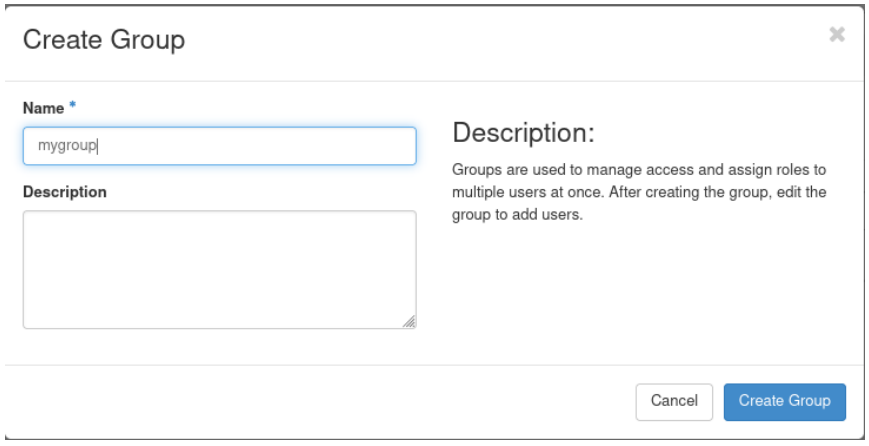


Figura 28. Creación de "*mygroup*", dentro de "*mydomain*".

Igual que el usuario creado anteriormente, el grupo "*mygroup*" se puede visualizar en el *dashboard* de *OPENSTACK*, en la dirección:

Identity → Groups

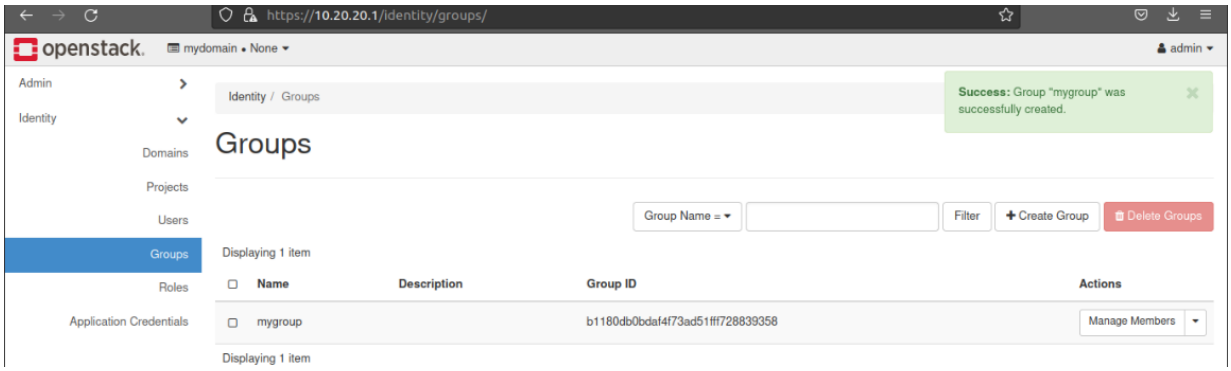


Figura 29. Visualización de "*mygroup*" en el *dashboard* de *OPENSTACK*.

De igual forma, en el cliente de consola, por medio del comando:

```
$ openstack --insecure group list --domain mydomain
```

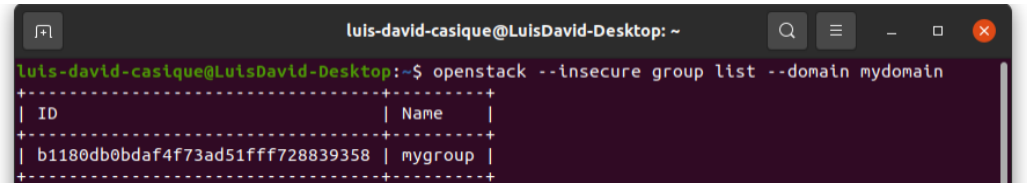


Figura 30. Visualización de "*mygroup*" mediante *OPENSTACK client*.

Para finalizar, se gestionaron las membresías de las identidades creadas. Se añadió el usuario "*myuser*" al grupo de usuarios "*mygroup*", no se incluyó el perfil administrador por ser un usuario global. A su vez, "*mygroup*" fue añadido al proyecto "*myproject*". A través del *dashboard* de *OPENSTACK*:

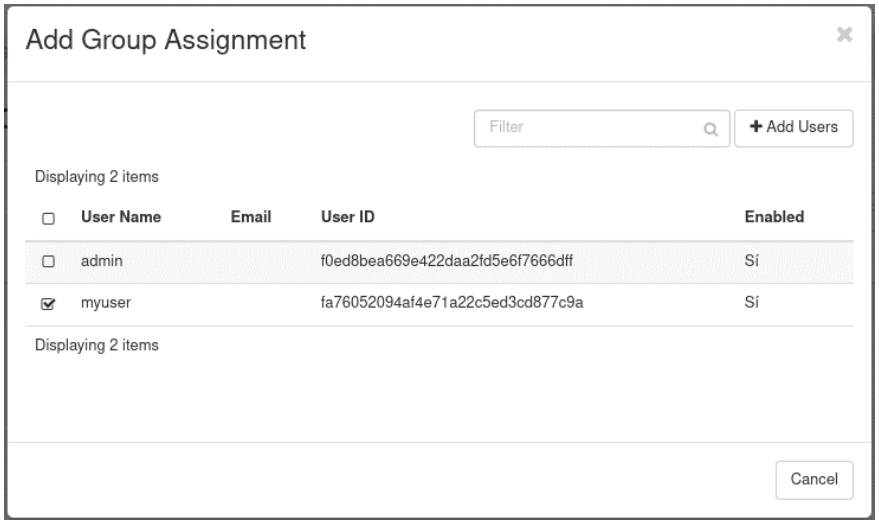


Figura 31. Inclusión de "*myuser*" dentro de "*mygroup*".

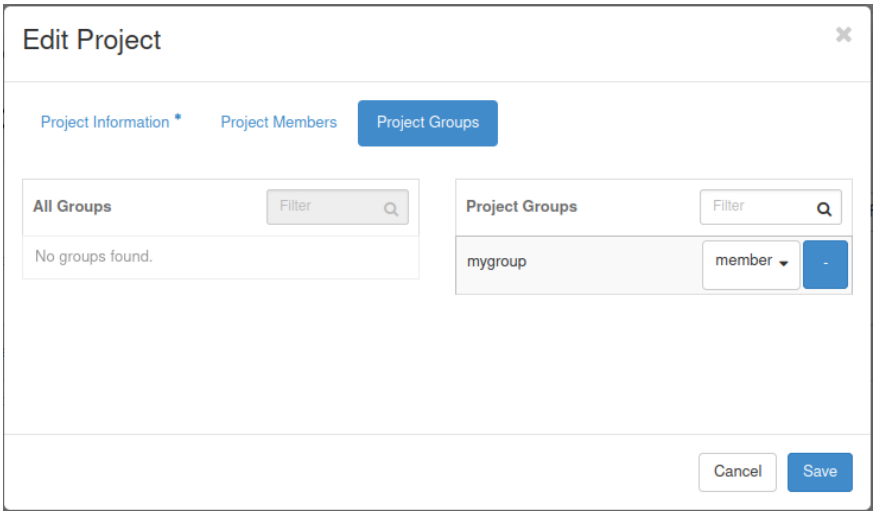


Figura 32. Inclusión de "*mygroup*" dentro de "*myproject*".

Con este paso, se finalizó el acondicionamiento del entorno de *OPENSTACK* de forma satisfactoria. Con la presente configuración, se realizó el resto de las pruebas descritas a continuación, dentro del dominio "*mydomain*", el proyecto "*myproject*" y el usuario "*myuser*".

IV.3.4. Configuración de parámetros de instancias

En este punto, se realizó un conjunto de configuraciones adicionales fundamental para la creación de las instancias. Se crearon y gestionaron imágenes de distintos sistemas operativos, volúmenes de memoria principal y secundaria (denominados *flavors*), grupos de seguridad, direcciones IP flotantes, llaves de acceso (denominadas *key pairs*), redes y *routers*.

IV.3.4.1. Gestión y creación de flavors

Se puede realizar por medio del *dashboard* y del cliente de *OPENSTACK*; por comodidad, se decidió crear por medio del *dashboard*, un *flavor* de características de "gama baja", denominado "Flavor-Prueba-1", a través de las siguientes opciones y características:

Admin → Compute → Flavors → Create Flavor

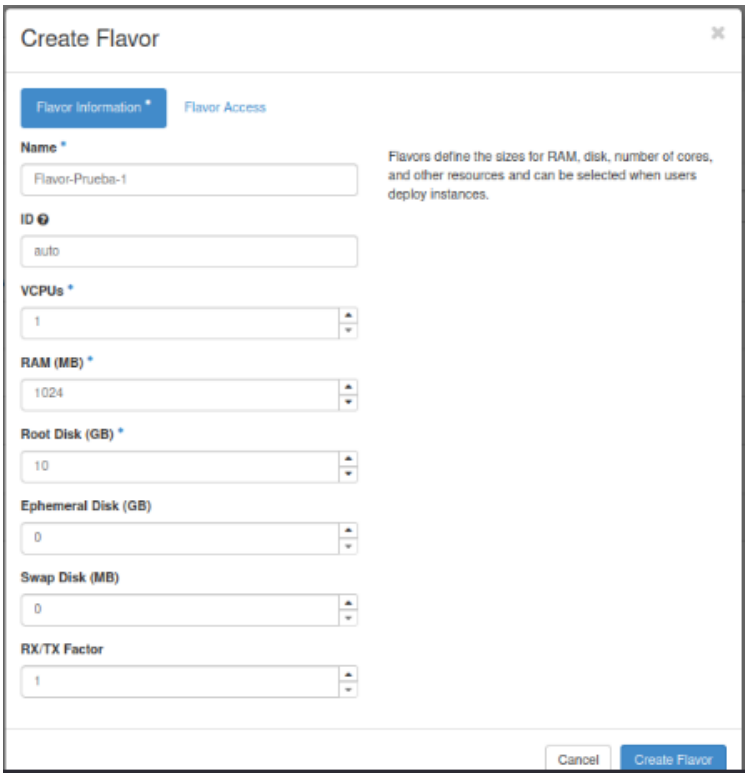


Figura 33. Creación del "Flavor-Prueba-1" en el *dashboard* de *OPENSTACK*.

Se puede visualizar el *flavor* en el *dashboard* (Figura 34) y en el cliente de consola (Figura 35):

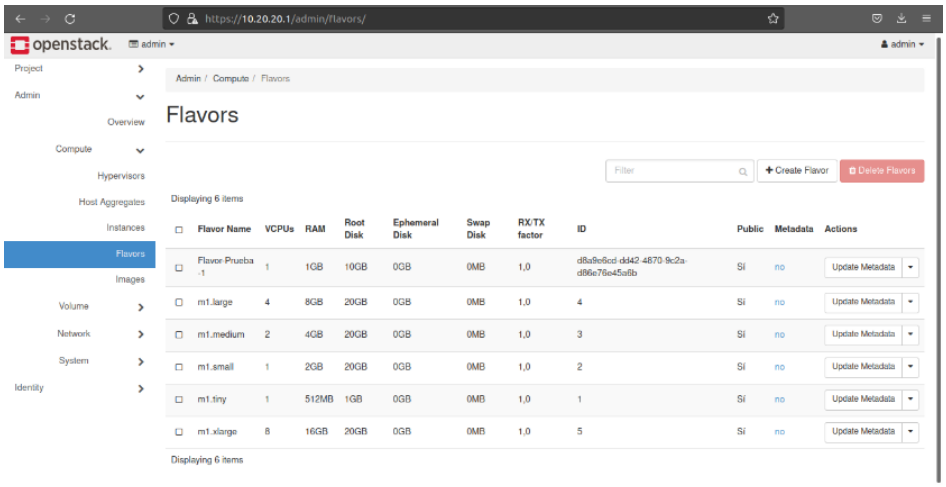


Figura 34. Visualización del "Flavor-Prueba-1" en el *dashboard*.

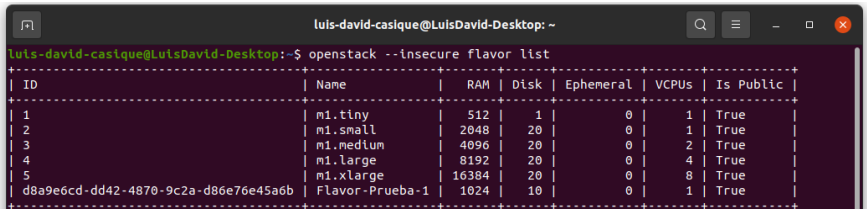


Figura 35. Visualización del "Flavor-Prueba-1" en el cliente de consola.

IV.3.4.2. Creación de imágenes

Para efectos de estas pruebas, se crearon imágenes de dos versiones del sistema operativo UBUNTU, una de la versión 20.04 LTS y otra con la versión 18.04 LTS, utilizando los servicios del módulo *Glance* de *OPENSTACK* [28]; ambas imágenes, fueron descargadas desde un repositorio de imágenes de UBUNTU preparadas y optimizadas para nube, disponible en la web [29]. Este proceso, también se puede realizar por medio del *dashboard* y por medio del cliente; para experimentar también con el cliente de *OPENSTACK*, se decidió crear las imágenes utilizando el mismo, a través de los siguientes comandos:

```
$ openstack --insecure image create --disk-format qcow2 --min-disk 8 --min-ram
512 --file ~/Descargas/focal-server-cloudimg-amd64-disk-kvm.img --public 20.04

$ openstack --insecure image create --disk-format qcow2 --min-disk 8 --min-ram
512 --file ~/Descargas/bionic-server-cloudimg-amd64.img --private 18.04
```

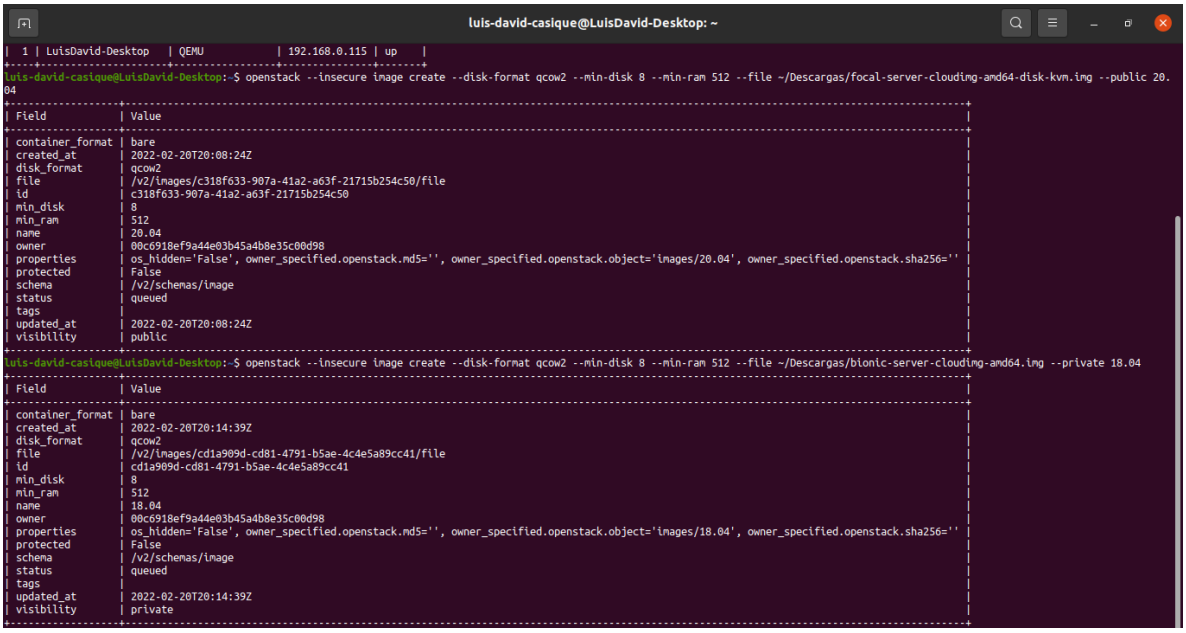


Figura 36. Creación de imágenes de UBUNTU 20.04 LTS y 18.04 LTS por medio del cliente de *OPENSTACK*.

IV.3.4.3. Reserva de dirección IP flotante

Para comunicarse con dispositivos externos a la nube, *OPENSTACK* reserva un grupo de direcciones IP flotantes del bloque 10.20.20.0/24, de la cual, la dirección 10.20.20.1 corresponde al nodo de control. Cada instancia, debe tener una dirección IP flotante del bloque descrito, la cual se puede reservar para el proyecto por medio del *dashboard* de *OPENSTACK*. Se hizo por medio de los siguientes pasos [30].

Project → Network → Floating IPs → Allocate IP to a Project

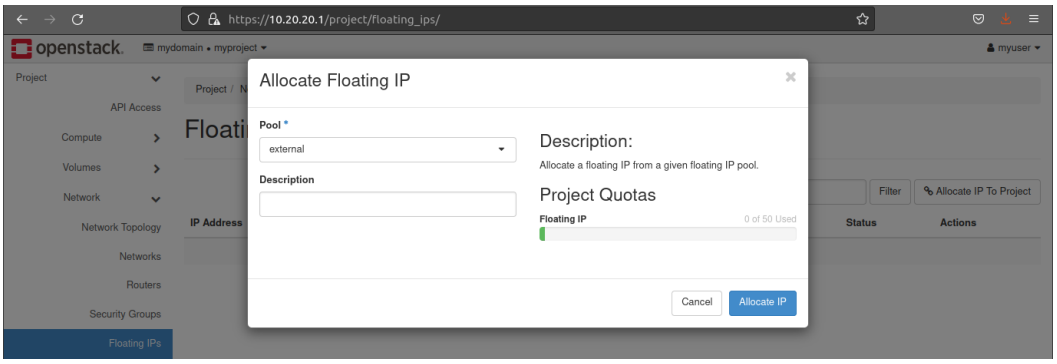


Figura 37. Asignación de Dirección IP flotante al proyecto "*myproject*".

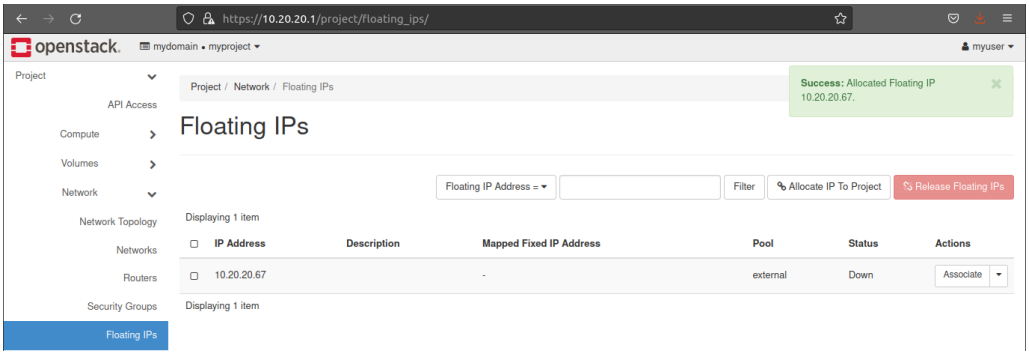


Figura 38. Visualización de Dirección 10.20.20.67, reservada para "*myproject*", en el *dashboard*.

De igual forma, se puede observar la dirección IP por medio del cliente de *OPENSTACK*, por medio del comando:

```
$ openstack --insecure floating ip list
```

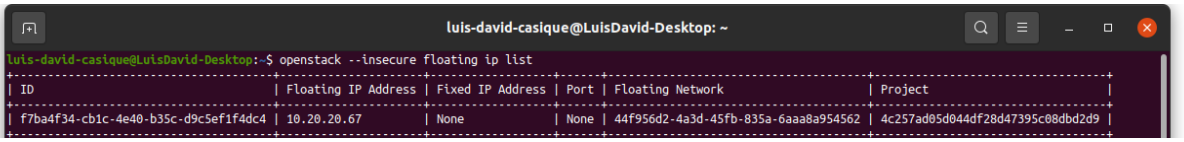


Figura 39. Visualización de Dirección 10.20.20.67, reservada para "*myproject*", en el cliente de *OPENSTACK*.

IV.3.4.4. Creación de llave de acceso (key pair)

Para el acceso remoto a la instancia, vía SSH, se creó una llave de acceso denominada "*mykeypair*", en formato clave de acceso en un archivo. pem, denominado "*mykeypair.pem*", se seleccionó el tipo de clave SSH, a través del *dashboard* de *OPENSTACK*.

Project → Compute → Key Pairs → Create Key Pair

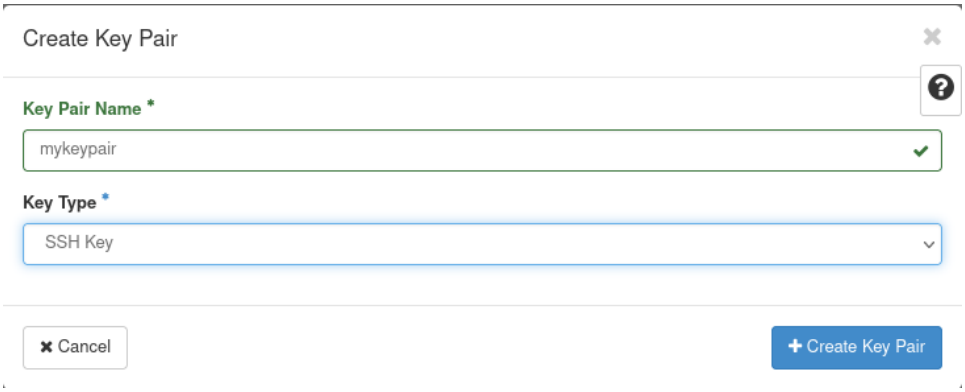


Figura 40. Creación de clave de acceso "*mykeypair*", del tipo SSH.

IV.3.4.5. Creación de red

Para el despliegue de entornos de redes virtuales, se deben crear redes con sus respectivas direcciones IP según el tipo de topología que se desee probar. A efectos de la presente prueba, se creó una red denominada "*mynetwork*", con direcciones IP del bloque 192.168.0.0/24, con el subgrupo de asignación del Protocolo de Configuración Dinámica de Host (*Dynamic Host Configuration Protocol*, DHCP) desde la dirección 192.168.0.101 hasta la 192.168.0.200, denominado "*mysubnet*", y DNS 8.8.8.8. Por medio del *dashboard* de *OPENSTACK* [30].

Project → Networks → Create Network

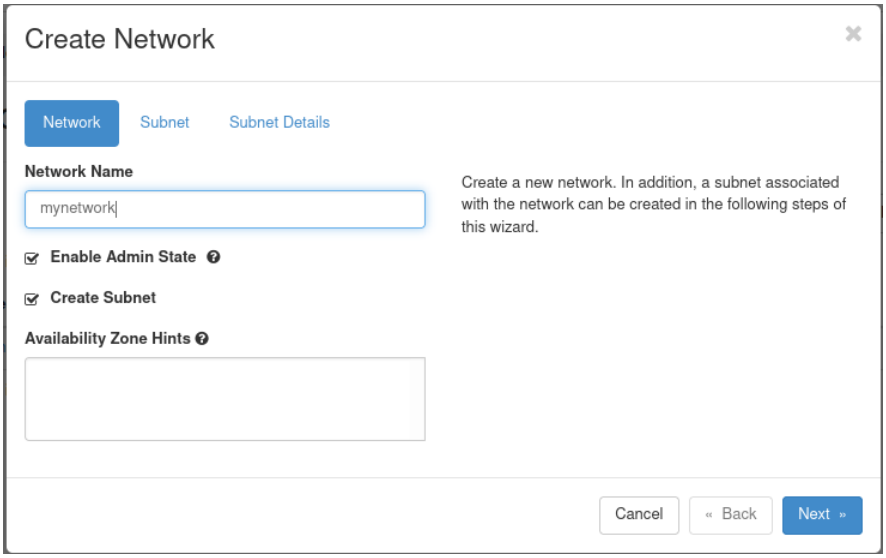


Figura 41. Creación de "*mynetwork*".

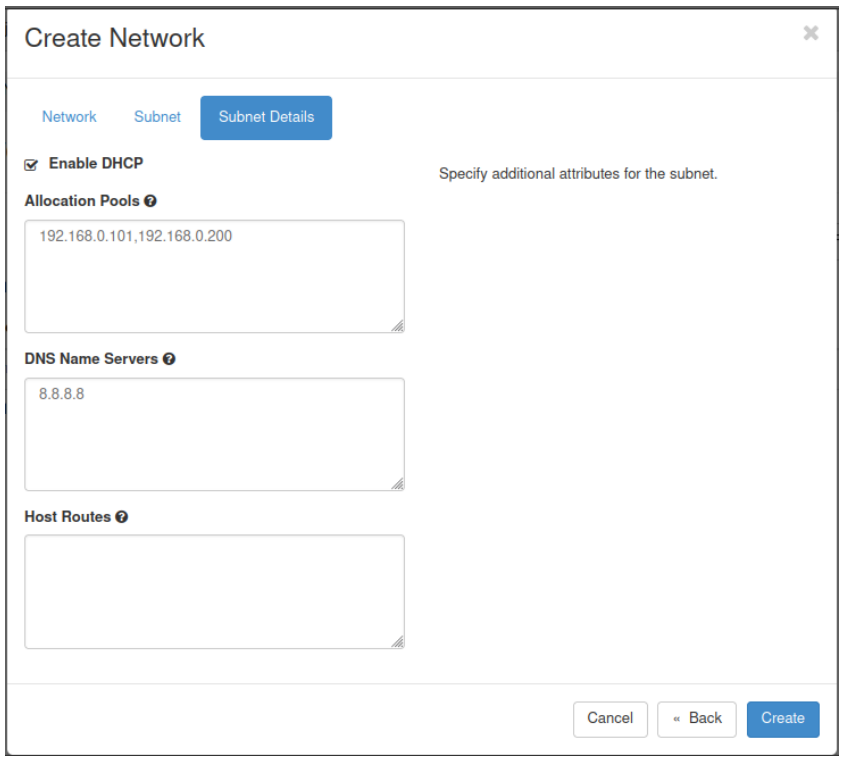


Figura 42. Creación de "*mysubnet*", con sus respectivos bloques de direcciones IP.

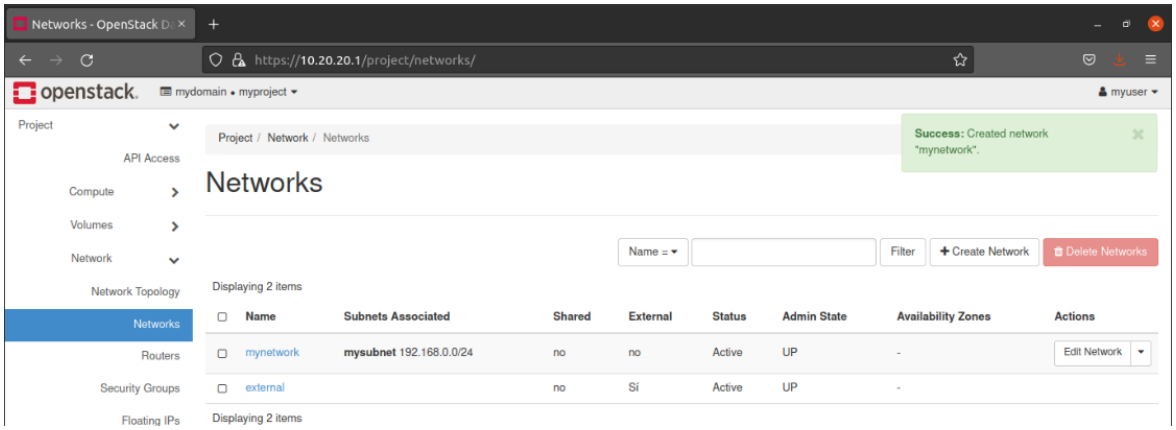


Figura 43. Visualización de "*mynetwork*" y "*mysubnet*" en el *dashboard*.

IV.3.4.6. Creación de router

Para la comunicación entre las distintas redes virtuales de la nube y el nodo de control, se debe crear al menos un enrutador virtual para el direccionamiento de los paquetes entrantes y salientes de cada una de ellas. Se creó un *router* denominado "*myrouter*" con dos interfaces de red, una para la red "*mynetwork*" y la otra para la red "*external*", que se encarga de la comunicación con el nodo de control; por medio del dashboard de *OPENSTACK*, a través de los siguientes pasos [30]:

Projects → Networks → Routers → Create Router

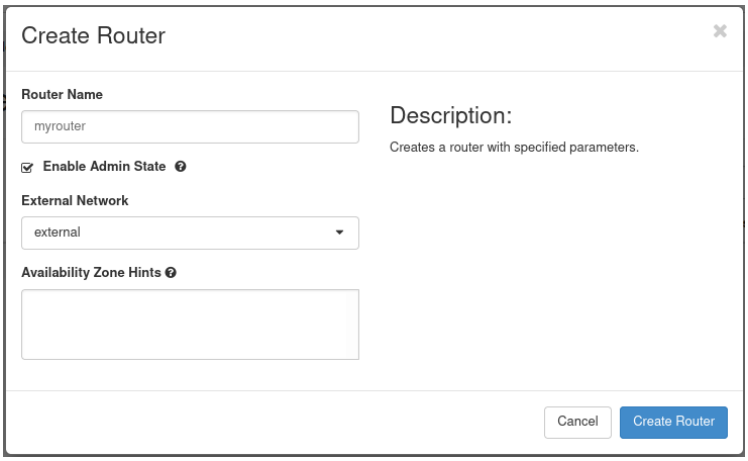


Figura 44. Creación del enrutador "*myrouter*".

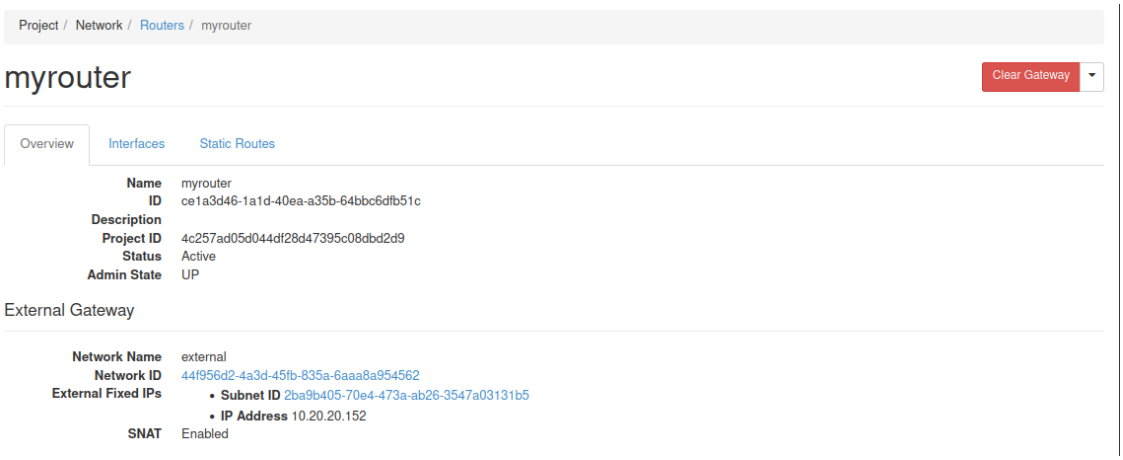


Figura 45. Características de "*myrouter*".

Como puede observarse, por defecto, "*myrouter*" se creó con una interfaz, y se le asoció una dirección IP flotante, la 10.20.20.152 como puerta de enlace (*Gateway*) hacia el nodo de control. La segunda interfaz para la red "*mynetwork*" fue añadida de forma manual:

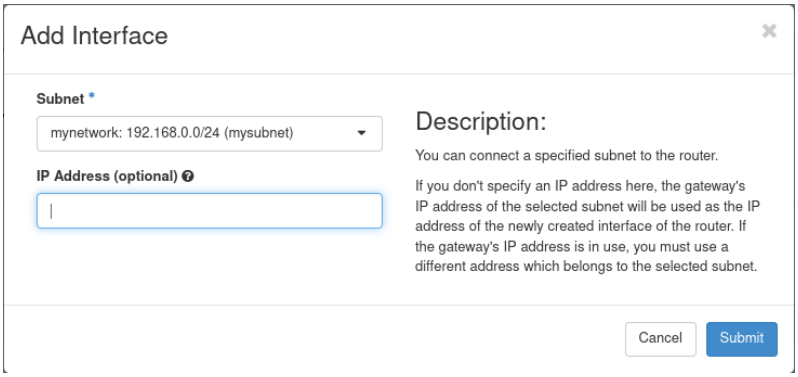


Figura 46. Adición de interfaz a "*myrouter*", para conectar con "*mynetwork*".

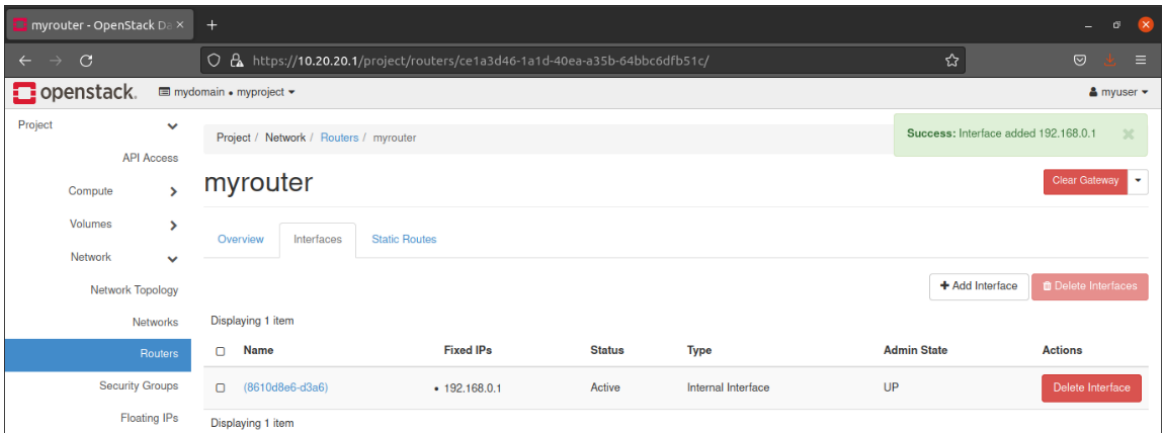


Figura 47. Visualización de la puerta de enlace de la red "*mynetwork*" en "*myrouter*".

IV.3.4.7. Gestión de grupos de seguridad

Se creó un nuevo grupo de seguridad denominado "*mysecuritygroup*", en el que se añadió una regla adicional para habilitar las conexiones SSH de ingreso, por medio del puerto 22 de TCP [30]. Se realizó por medio del *dashboard* de *OPENSTACK*, a través de los siguientes pasos:

Project → Network → Security Groups →Create Security Group

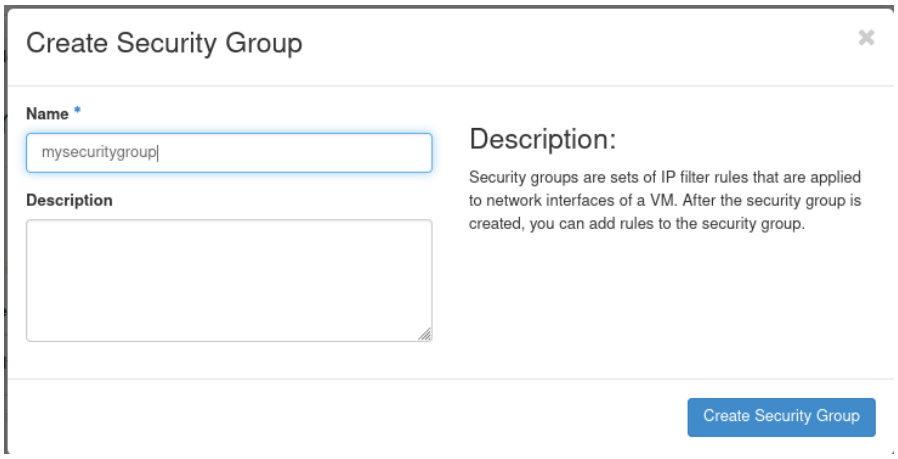


Figura 48. Creación de "*mysecuritygroup*".

Luego, se le añadió la regla de habilitación de conexión SSH:

Project → Network → Security Groups → Manage Security Group Rules: mysecuritygroup → Add Rule

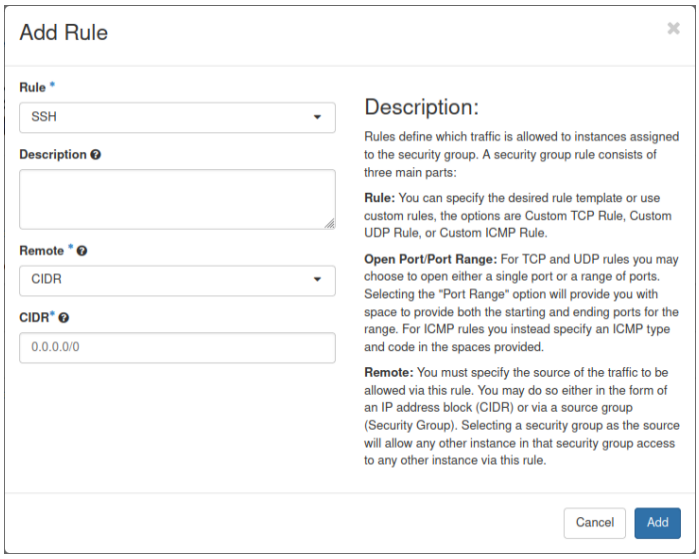


Figura 49. Adición de regla de conexión SSH en "*mysecuritygroup*".

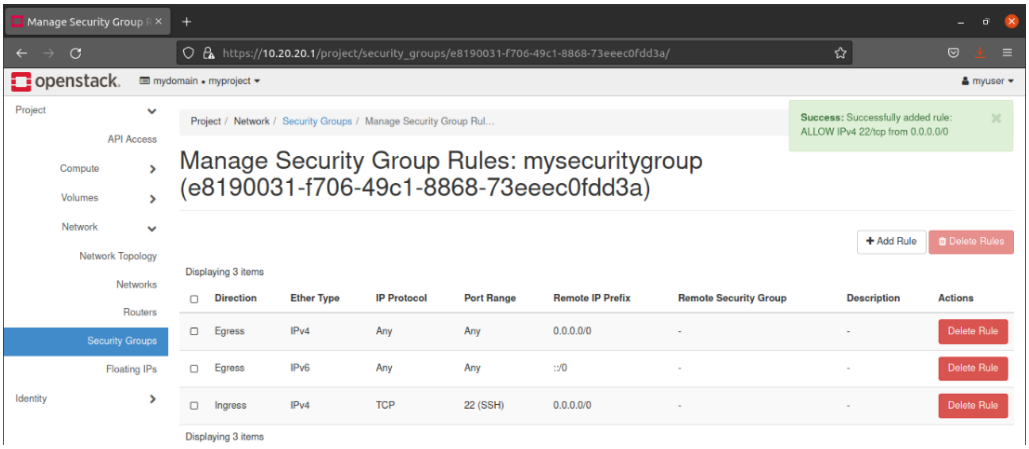


Figura 50. Visualización de las reglas del grupo "*mysecuritygroup*".

IV.3.4.8. Creación de nuevas instancias

Luego de que fueron preparados los *flavors*, imágenes de SOs, direcciones IP flotantes, redes, subredes y al menos un *router*, se pudo armar una infraestructura de red de instancias en la nube de *OPENSTACK*. El siguiente paso fue la creación de dos nuevas instancias con los parámetros descritos, denominadas "*myinstance1*" y "*myinstance2*". A través del *dashboard* de *OPENSTACK*, se siguieron los siguientes pasos y especificaciones obligatorias:

Project → Compute → Instances → Launch Instance

1. **Detalles:** Se especificó el nombre "*myinstance*", descripción "*My instance*", Nova como zona de disponibilidad (única disponible) y en cantidad, 2 (*myinstance1* y *myinstance2*). Lo anteriormente descrito se ilustra en la **Figura 51**.

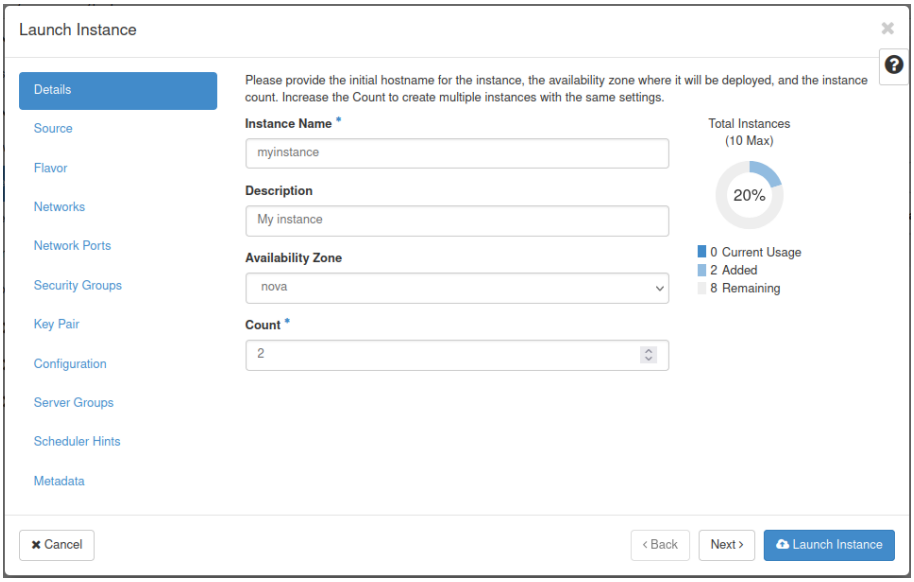


Figura 51. Lanzamiento de "*myinstance1*" y "*myinstance2*", sección "*Details*".

2. **Fuente (*Source*):** Se especificó la fuente de arranque para la instancia del tipo "imagen", y como imagen se seleccionó la del SO UBUNTU 20.04 LTS. No se creó un nuevo volumen para el almacenamiento, sino que fue gestionado por medio del *flavor* asignado.

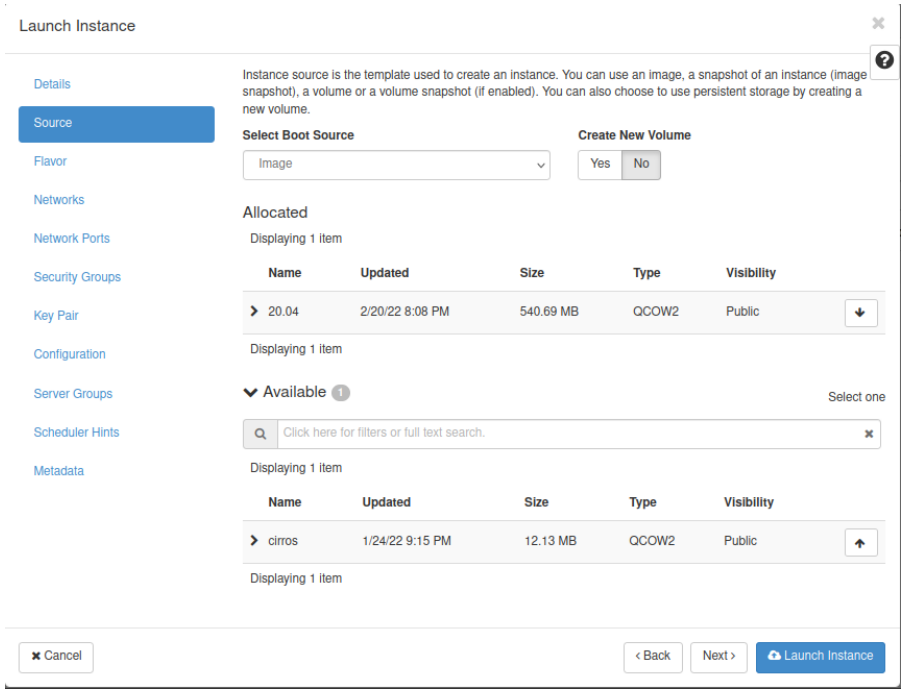


Figura 52. Lanzamiento de "*myinstance1*" y "*myinstance2*", sección "*Source*".

3. ***Flavor*:** Se seleccionó "*Flavor-Prueba-1*", creado en el respectivo paso anterior, porque es el más reducido en cuanto a cantidades de sus características, dado el *hardware* limitado del nodo de control (*LuisDavid-Desktop*).

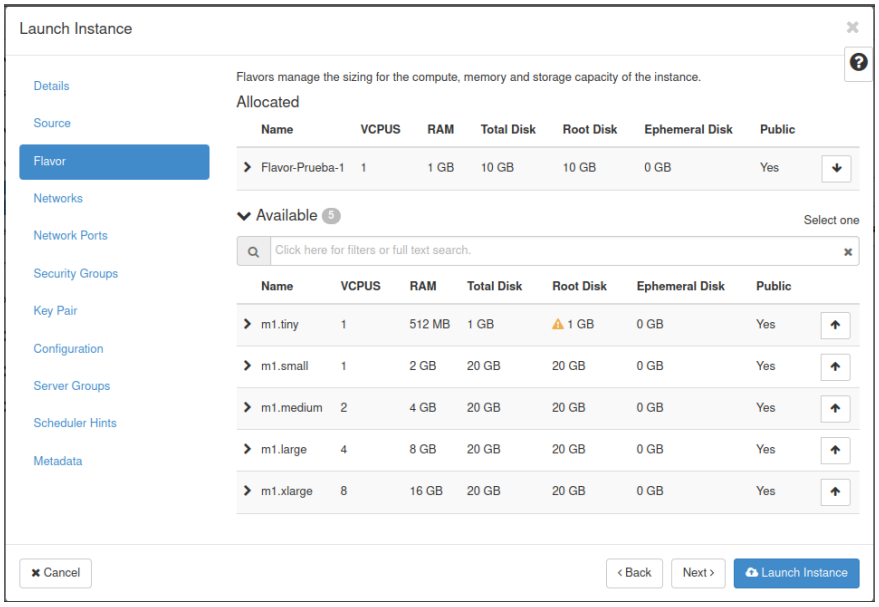


Figura 53. Asignación de "Flavor-Prueba-1" a "*myinstance*".

1. ***Networks*:** Se seleccionó la red "*mynetwork*", y fueron reservadas de forma automática las direcciones IP 192.168.0.170 y la 192.168.0.188 de "*mysubnet*", para "*myinstance1*" y "*myinstance2*" respectivamente.

2. ***Security Groups*:** Se seleccionó el grupo "*mysecuritygroup*", con la regla SSH incorporada anteriormente, la cual ilustra la **Figura 49**.

3. ***Key Pair*:** Se seleccionó la clave "*mykeypair*", creada en el respectivo paso anterior, para el acceso remoto vía SSH.

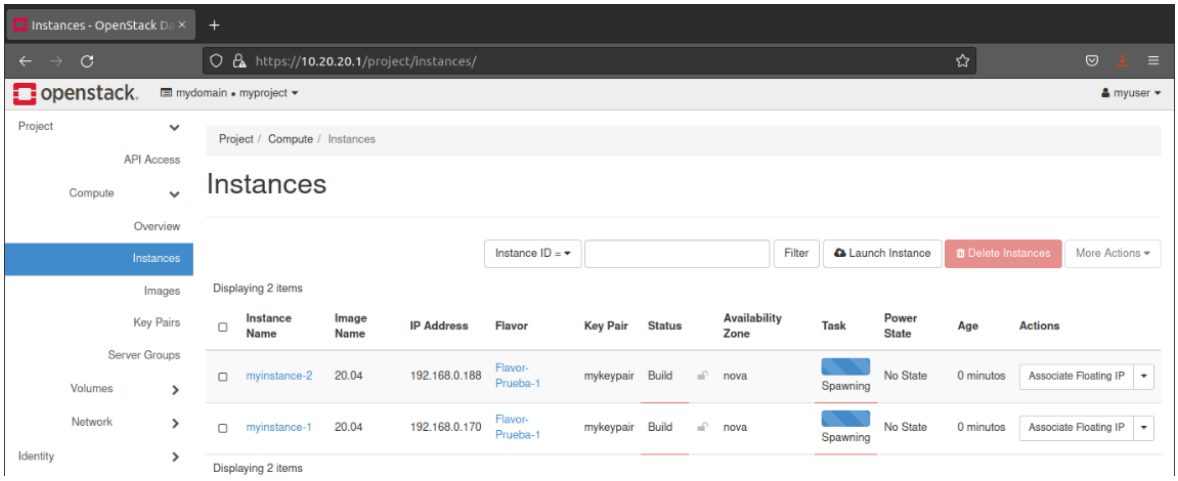


Figura 54. Visualización de "*myinstance1*" y "*myinstance2*" en el *dashboard* de *OPENSTACK*.

Una vez creadas las instancias, se realizó un grupo de ajustes adicionales:

Asociación de IP flotante: Se reservó y asignó una dirección IP flotante, inicialmente para "*myinstance1*", en este caso, la 10.20.20.67.

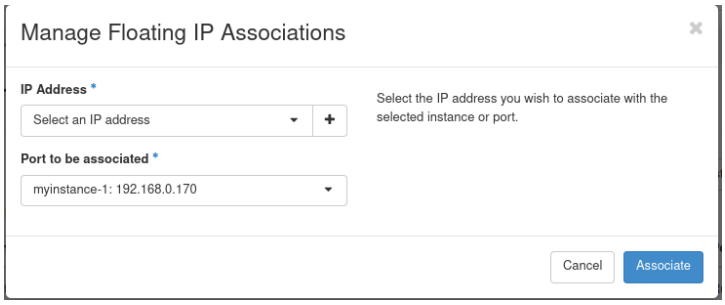


Figura 55. Asignación de dirección IP flotante para "*myinstance1*".

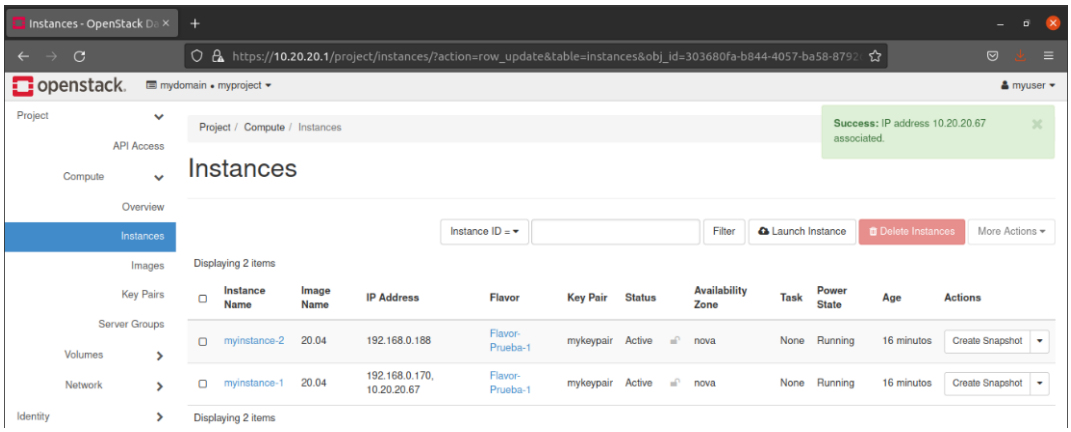


Figura 56. Actualización de las instancias, con la dirección 10.20.20.67 asociada a "*myinstance1*".

Adjuntado de grupo de seguridad: Posteriormente, se adjuntó "*myinstance1*"al grupo de seguridad "*mysecuritygroup*" creado anteriormente, al cual se le añadió la regla para la habilitación de conexión remota vía SSH. Se realizó por medio del *dashboard*, siguiendo la ruta:

Project → Instances → myinstance1 → Edit Instance.

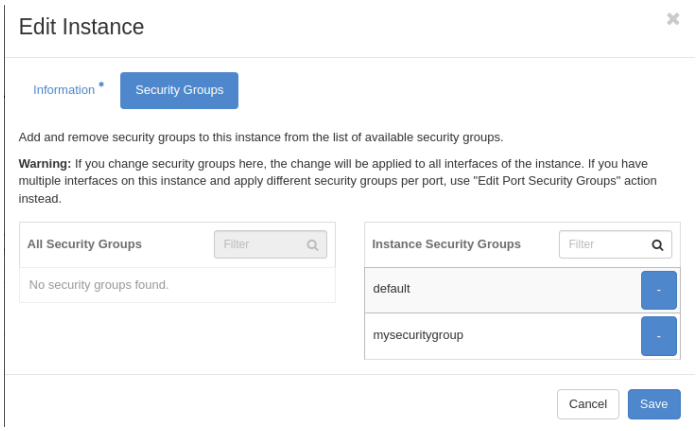


Figura 57. Adjuntado de "*mysecuritygroup*" a "*myinstance1*".

IV.3.5. Realización de pruebas con las instancias

Luego de haber creado las instancias y haber modificado sus parámetros, se procedió a realizar un conjunto de pruebas de funcionalidad con las mismas, que consistieron en el acceso e interacción remota por medio del protocolo SSH y la comunicación entre ellas y el nodo de control de la nube por medio del envío de paquetes ICMP.

IV.3.5.1. Acceso SSH

Se accedió a "*myinstance1*", utilizando la llave de acceso "*mykeypair*" creada en el respectivo paso anterior. Previo, se modificaron los permisos de lectura y escritura del archivo de la llave de acceso (*mykeypair.pem*), para que únicamente el propietario de este pueda leer y escribir. Se realizó por medio de la consola, a través del siguiente comando:

```
$ chmod 600 /home/luis-david-casique/Descargas/mykeypair.pem
```

Luego, se ingresó a la instancia "myinstance1" vía SSH, con el comando:

```
$ ssh -i Descargas/mykeypair.pem ubuntu@10.20.20.67
```

Antes de acceder, se debe continuar de forma manual el proceso de autenticación de la firma de la llave de acceso correspondiente al Algoritmo de Firma Digital de Curva Elíptica (*Elliptic Curve Digital Signature Algorithm*, ECDSA), se realizó tipeando "yes" a la pregunta que planteaba el s*hell*:

Luego de este paso previo, la conexión se logró de forma satisfactoria (Figura 58).

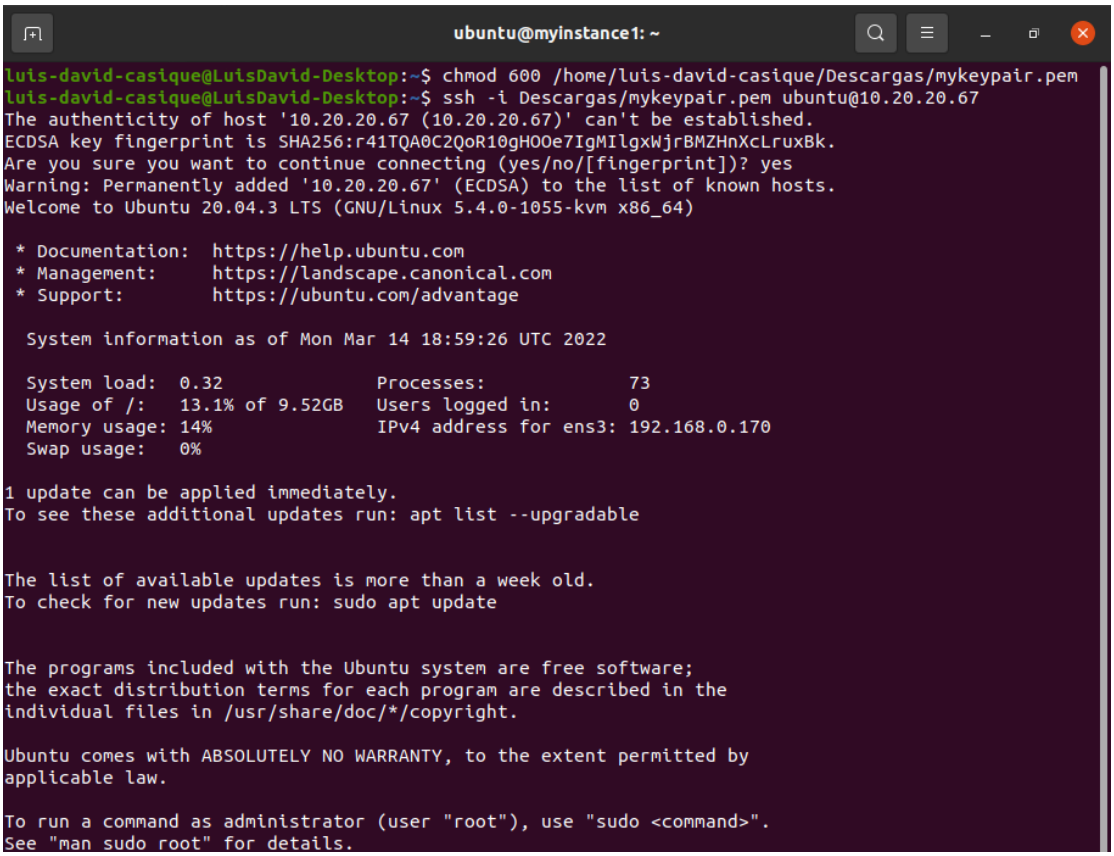


Figura 58. Acceso remoto a "*myinstance1*" vía SSH.

Una vez que se logró el acceso a “myinstance2”, se ejecutaron algunos comandos simples para comprobar la funcionalidad de la instancia y observar algunos parámetros como:

Visualización de las interfaces de red: `$ ip address`

Visualización la hora y el tiempo de actividad de la instancia: `$ uptime`

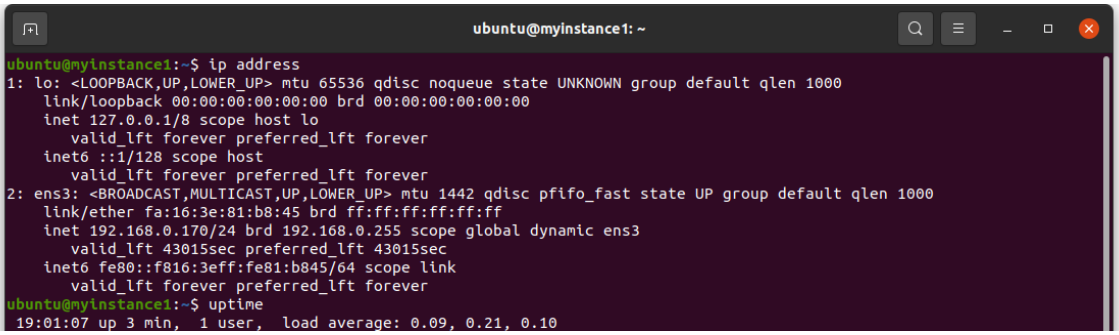


Figura 59. Visualización de las interfaces de red y del tiempo de ejecución de "*myinstance1*", desde su propia consola.

Se verificó el envío de paquetes hacia el nodo de control: `$ ping 10.20.20.1`

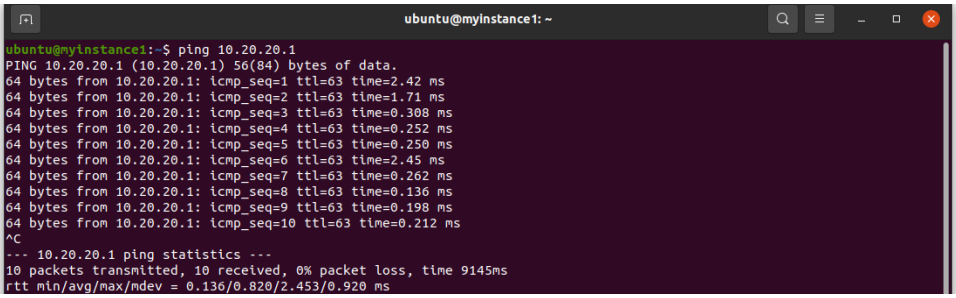


Figura 60. Comunicación ICMP directa entre "*myinstance1*" y el nodo de control de la nube.

IV.3.5.2. Comunicación entre instancias

Como las instancias creadas pertenecen a la misma red, "*mynetwork*", con direcciones IP del mismo bloque CIDR (*Classless Inter-Domain Routing*), pueden comunicarse entre sí de forma directa intercambiando paquetes ICMP, de la manera ilustrada en la **Figura 61**.

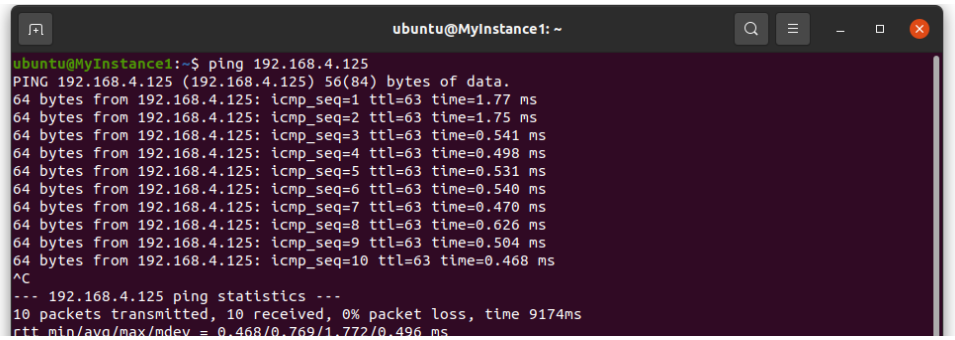


Figura 61. Envío de paquetes exitoso desde "*Myinstance1*" hasta "*MyInstance2*".

El envío de paquetes en sentido opuesto entre ambas instancias, se muestra a continuación (Figura 62):

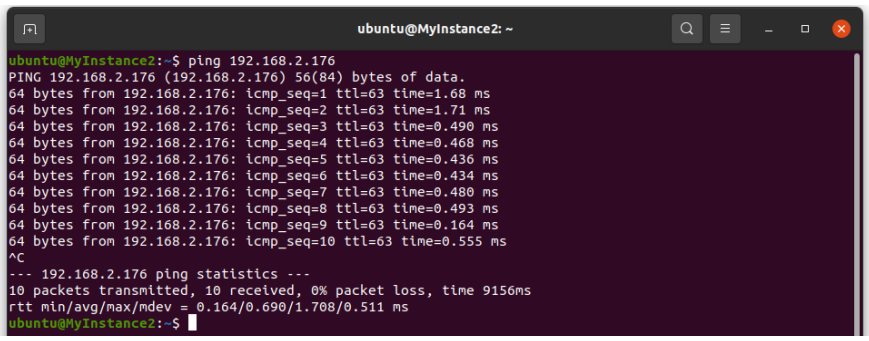


Figura 62. Envío de paquetes exitoso desde "*Myinstance1*" hasta "*MyInstance2*".

IV.3.5.3. Topología de red desplegada

Durante las pruebas locales, se implementó una topología de red que permitió crear instancias en redes de área local o LANs distintas, para verificar que puedan comunicarse entre sí

utilizando a *myrouter* como enrutador de la red. A *myrouter* se le crearon tres interfaces de red que funcionaron como las puertas de enlace a las subredes respectivas. La topología de red desplegada final es la siguiente (Figura 63 y Figura 64):

Para mostrar las capturas a partir de esta sección, se decidió cambiar el tema personalizado de colores del *dashboard* de *OPENSTACK* por defecto, al tema UBUNTU, debido a que, visualmente, resulta más cómodo a la vista y más fácil de leer e interpretar los datos, a criterio de los autores del presente tomo.

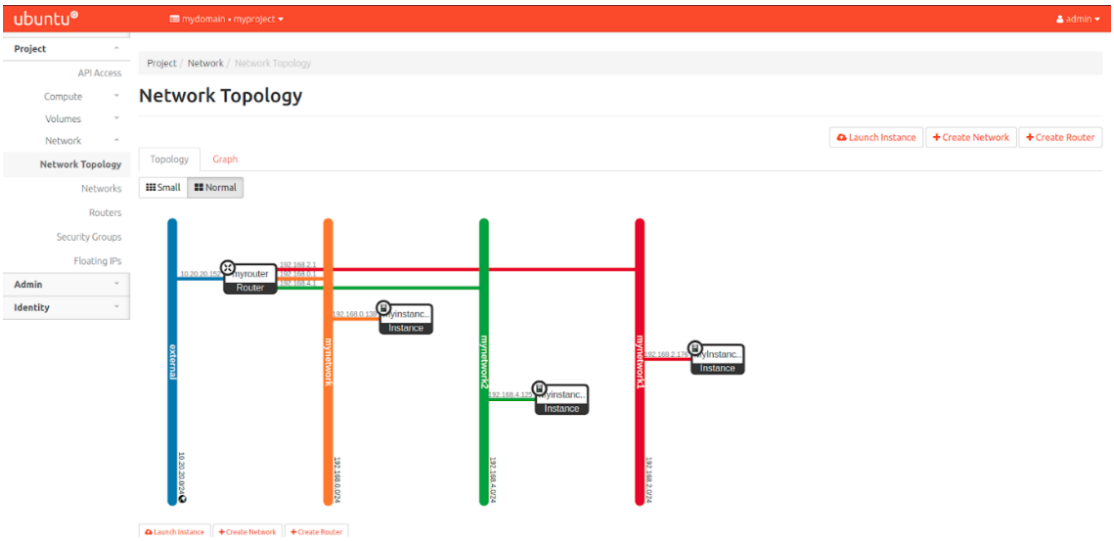


Figura 63. Diagrama N°1 de la topología resultante.

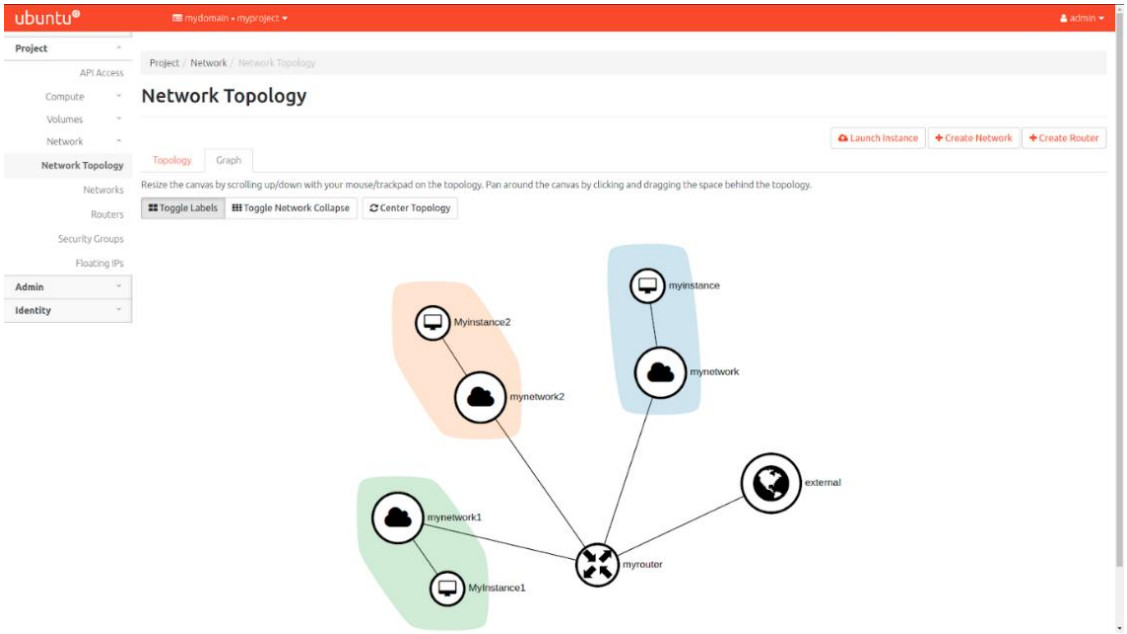


Figura 64. Diagrama N°2 de la topología resultante.

IV.3.6. Conexión de nodo de cómputo a la micro nube

Con el fin de practicar la adición de nodos de cómputo a la nube previo a la instalación en el laboratorio de telemática, se añadió un nodo de cómputo a la "micro nube" local y se experimentó con su funcionamiento. Para ello, se siguió la documentación de *MICROSTACK* para el despliegue de una nube en el esquema de múltiples nodos (*multi-node*) [26].

El segundo nodo es otro computador personal local, perteneciente a los autores del presente tomo, al cual le fue instalado el SO UBUNTU en su versión 22.04. LTS, con el fin de, analizar adicionalmente el comportamiento de *MICROSTACK* entre nodos con distintas versiones de UBUNTU. El nombre de *host* del dispositivo incorporado es "macxy-desktop", con el que se identifica en las futuras capturas de pantalla.

Previo a la instalación de *MICROSTACK*, se instaló el paquete *net-tools* en el segundo nodo, para mejorar la interacción con las interfaces de red de este con la familia de comandos `ifconfig`, tal como se ilustra en la **Figura 65**.

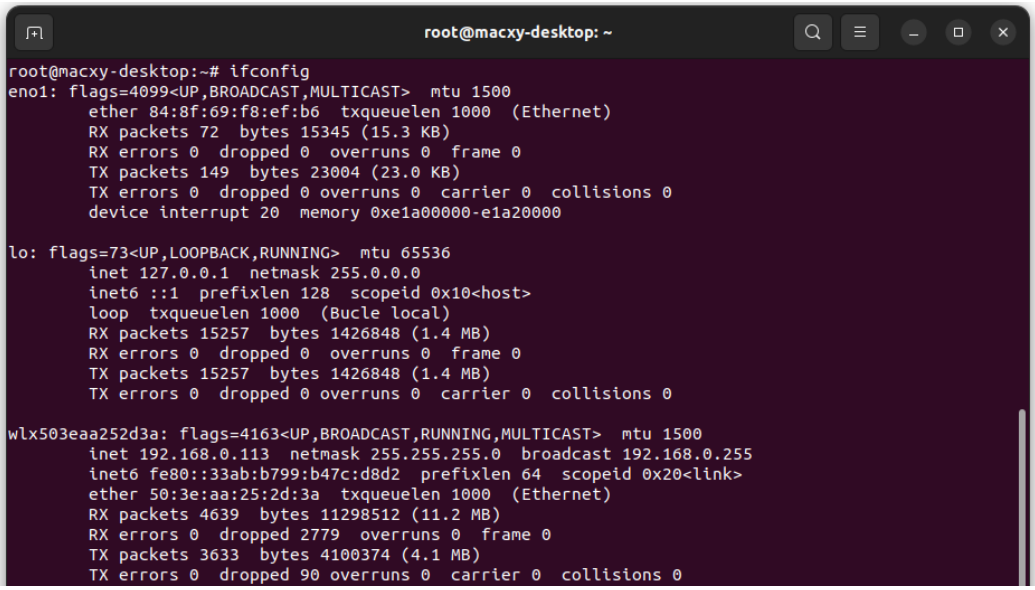


Figura 65. Interfaces de red del segundo nodo.

Los pasos para integrar el nodo de cómputo adicional a la infraestructura de nube se describen en las siguientes secciones:

IV.3.6.1. Descarga de MICROSTACK

Al igual que en el nodo de control, se necesita la aplicación *MICROSTACK* en los nodos de cómputo de la nube, la cual se descargó e instaló con el comando:

```
$ sudo snap install microstack --beta
```

IV.3.6.2. Generación e introducción de cadena de conexión

Para añadir y sincronizar un nodo de cómputo a un clúster de *MICROSTACK*, se necesita generar una cadena de conexión (*connection string*) en un nodo de control previamente inicializado con *MICROSTACK*, perteneciente a la misma LAN que el nodo de cómputo. En este caso, se generó el *string* desde el nodo de control de la "*micro nube*", representado por el host "LuisDavid-Desktop", por medio del comando:

```
$ sudo microstack add-compute
```

Este comando, arrojó como salida en pantalla, el siguiente *string* o cadena de conexión:

```
hKhob3N0bmFtZa0xOTIuMTY4LjAuMTE1q2ZpbmdlcnByaW50xCABHbQoEeo7tn96QJogPgEa
+GmjxpoM4p9EFliH8R8bVqJpZNkgMzQ3NjczOWM5YzFiNDNmZWJiYTRiZjRmY2VlM2I1ZWamc2Vjcm
V02SBOWXBlZWN5aTRCTHlzQ2tnUlFhNmcwNEpQZ1MyWkNfZw==
```

Luego, en el nodo de cómputo, se ejecutó el comando que permite iniciar la aplicación y adjuntarse al nodo de control conformando el clúster, el comando es:

```
$ sudo microstack init --auto --compute –join <Cadena de conexión>
```

Se introdujo el comando con la cadena correspondiente y el nodo de cómputo procedió a deshabilitar los servicios que no utilizará, porque al ser nodo de cómputo, únicamente tiene la función de proveer su potencia de cómputo y capacidades de almacenamiento a la nube, mientras que la gestión de dichos recursos sigue siendo responsabilidad del nodo de control.

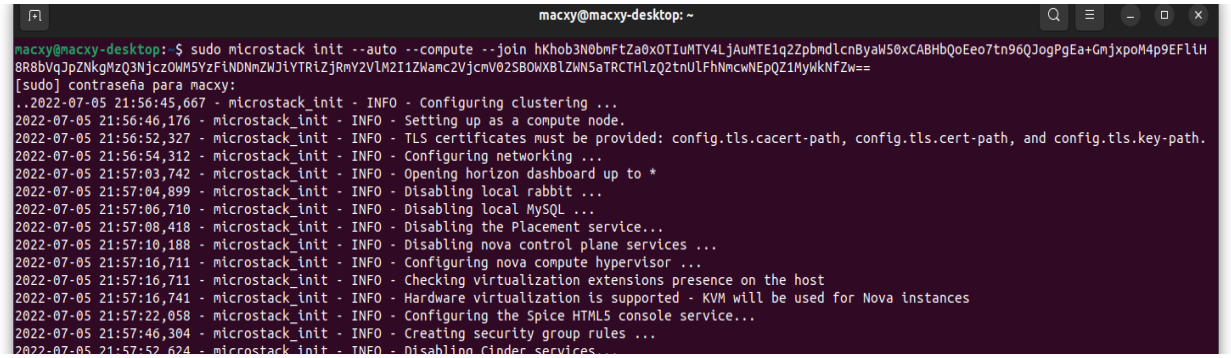


Figura 66. *MICROSTACK* inicializado en segundo nodo.

IV.3.6.3. Comprobación de creación de clúster

Existen varias formas de comprobar la inicialización de *MICROSTACK* en el nodo de cómputo y la creación del clúster de la nube; una de ellas, es la introducción de comandos de *OPENSTACK* que involucren a ambos nodos, en este caso, se solicitó la lista de hipervisores de la nube, por medio de los comandos:

```
$ openstack --insecure hypervisor list $ microstack.openstack hypervisor list
```

Este comando muestra los datos de los hipervisores provistos por cada nodo que conforma el clúster, tal como se refleja en la **Figura 67**.

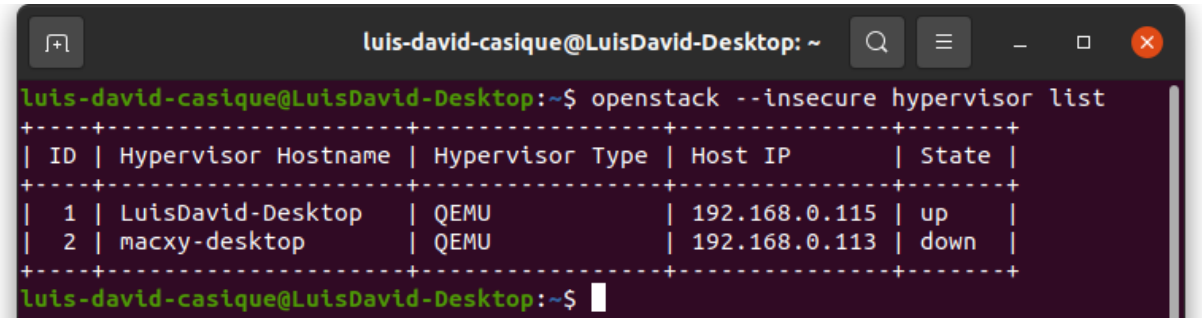


Figura 67. Visualización de hipervisores en el clúster local.

IV.3.6.3.1. Instancias en el nodo de cómputo

Siguiendo la documentación oficial *MICROSTACK*, para comprobar la inicialización, se intentó crear una instancia con los ajustes por defecto (imagen CirrOS), sobre el *hardware* del nodo de cómputo, a través del comando con la siguiente sintaxis:

```
$ microstack launch cirros -name <nombre> --availability-zone nova:<hostname>
```

Se creó una instancia denominada “test”, dentro del nodo de cómputo de la nube, tal como se ilustra en la **Figura 68**. Para tal fin, se empleó el comando:

```
$ microstack launch cirros –name test --availability-zone nova:macxy-desktop
```

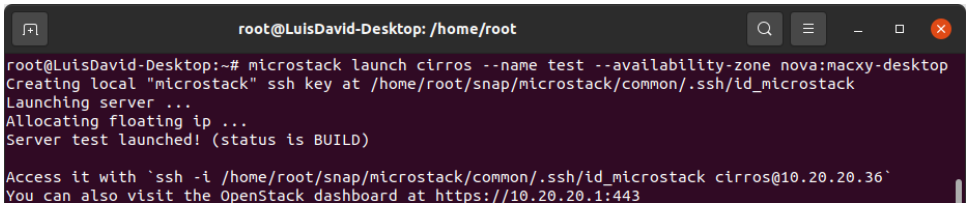


Figura 68. Instancia "*test*" creada en segundo nodo.

IV.3.6.3.2. Comprobación a través del dashboard

Otra forma de comprobar la creación del clúster con el nodo de cómputo es por medio del *dashboard* de *OPENSTACK*, accediendo desde el nodo de control, e ingresando las credenciales de un perfil con privilegios de administrador, ya que esta clase de perfiles permiten realizar una gestión global de los recursos de cómputo, red y almacenamiento disponibles en la nube. Tras acceder al *dashboard*, se ingresó al menú de hipervisores siguiendo la ruta: *Admin* / *Compute* / *Hypervisors*, tal como se ilustra en la **Figura 69**.

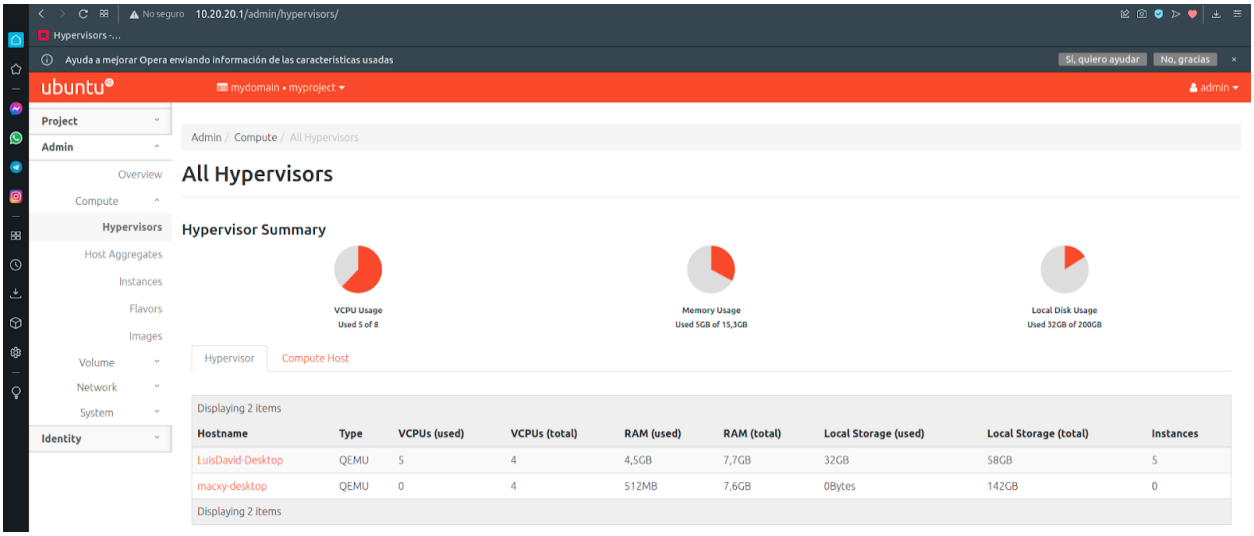


Figura 69. Visualización del nodo de cómputo “macxy-desktop” como hipervisor integrado al clúster.

Esta pantalla, en la pestaña “*Hypervisor*” permite observar los nodos pertenecientes al clúster y sus características físicas, como las vCPUs (CPUs virtuales) utilizadas, porcentaje de uso de la memoria RAM, y porcentaje de ocupación del espacio de almacenamiento disponible en la nube. Lo previamente descrito se ilustra en la **Figura 70**.

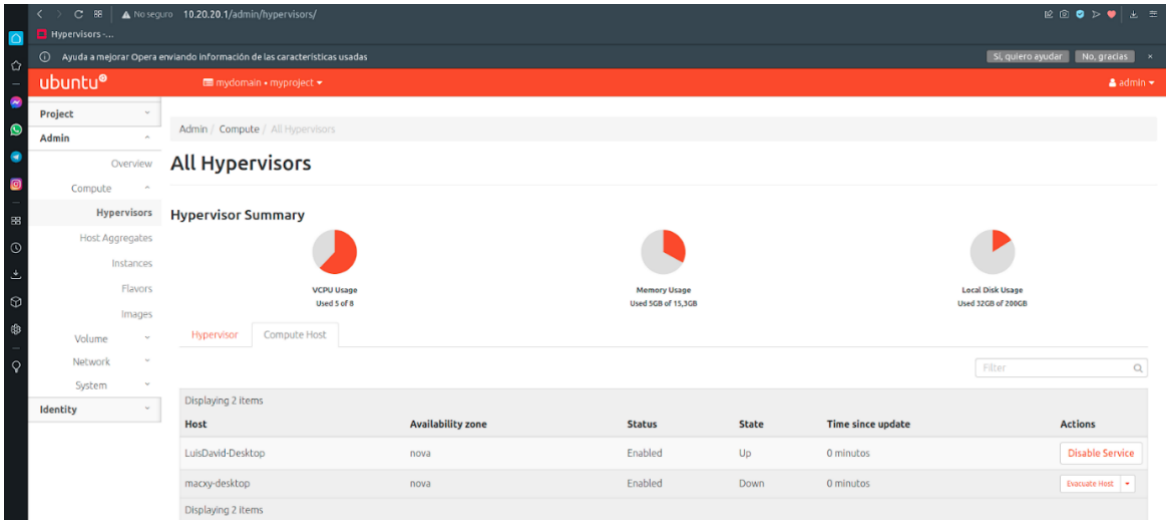


Figura 70. Visualización de los nodos de cómputo y control y su estatus dentro del clúster.

En la pestaña *Compute Host* de esta pantalla, se puede observar el estatus de los nodos conectados al clúster, y el tiempo desde su última actualización de estado. Adicionalmente, en los gráficos de uso, se observa que se agrupan y unen las características de cómputo de ambos nodos, mostrándose como un grupo de cómputo con las características combinadas, mientras que más abajo, se detalla la distinción de ambos nodos.

Adicionalmente, en la ruta *Admin*/*Compute*/*Host Aggregates* también puede observarse las "Zonas de Disponibilidad" (*Availability Zones*) de los servicios de *MICROSTACK*, en la que se puede distinguir que la función de cómputo está disponible por medio del módulo Nova de *OPENSTACK*, que es el responsable de la provisión de recursos de cómputo. Adicionalmente, se muestra una "zona" denominada "*internal*", la que indica que estos recursos son gestionados por el nodo de control.

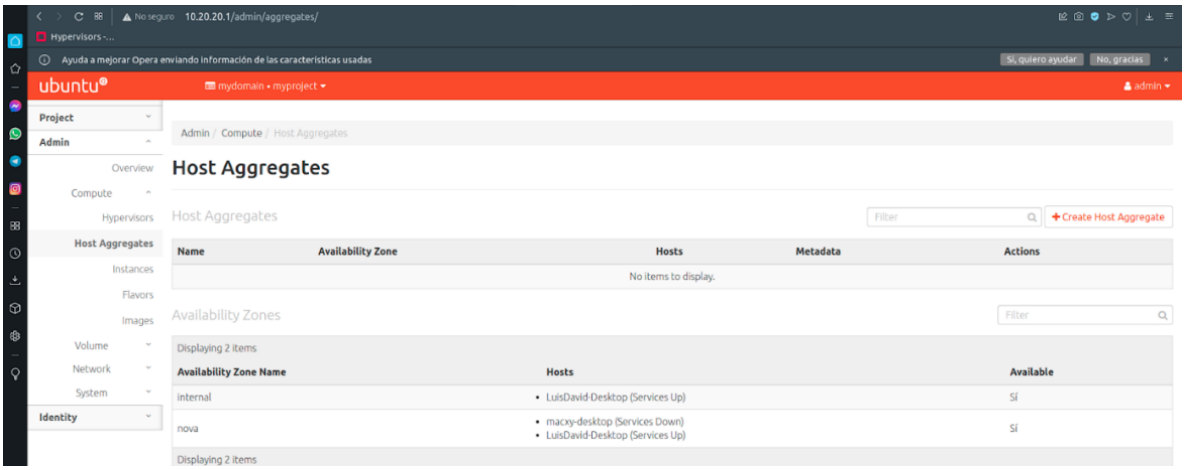


Figura 71. Pestaña de los hosts pertenecientes al clúster y los módulos respectivos de *OPENSTACK* que estos utilizan.

IV.4. Fase de Montaje de la nube en el Laboratorio de Telemática.

Luego de haber concluido exitosamente el ensayo de la "micro nube" local de prueba, ejecutada y documentada en la fase anterior, se procedió a implementar la nube directamente en la infraestructura del Laboratorio de Telemática de la institución, replicando el despliegue desarrollado en la fase II, e incrementando la cantidad de nodos de cómputo.

La implementación fue realizada bajo el esquema de múltiples nodos de *MICROSTACK* [21], empleando cinco computadoras del laboratorio, de las cuales, cuatro son nodos de cómputo, y una es el nodo de control, la cual a su vez pone su *hardware* a disposición de la nube, por lo tanto, constituye un quinto nodo de cómputo.

Para llevar a cabo el despliegue, se instaló UBUNTU 20.04.3 LTS en cinco máquinas del laboratorio, por ser la versión más reciente con soporte a largo plazo disponible, y posteriormente, se realizó la respectiva descarga e instalación de *MICROSTACK* en cada una de ellas.

IV.4.1. Operatividad de los nodos del laboratorio

La infraestructura como servicio, al funcionar como un servicio bajo demanda, requiere que sus recursos estén siempre disponibles para los consumidores, lo que hace necesario que los equipos que conforman dicha infraestructura estén operativos en todo momento funcionando como servidores de red. Para cumplir con esta condición, la infraestructura del laboratorio de telemática,

se llevaron a cabo un conjunto de medidas y acciones que permiten garantizar lo descrito anteriormente.

IV.4.1.1. Conexión a Internet

Los nodos que conforman la nube requieren de conexión a internet para la descarga de la aplicación *MICROSTACK*, para ello, el laboratorio de telemática cuenta con una infraestructura de cableado estructurado desde cada una de las computadoras, hasta un conmutador (o *switch*) gestionado por el Departamento de Tecnologías de Información (DTI). Este dispositivo, a su vez, permite la conexión de los equipos con un servidor del Protocolo de Configuración Dinámica de Host (*Dynamic Host Configuration Protocol*, DHCP) que le asigna direcciones IP a las computadoras del laboratorio, conectividad con la red de la universidad e Internet.

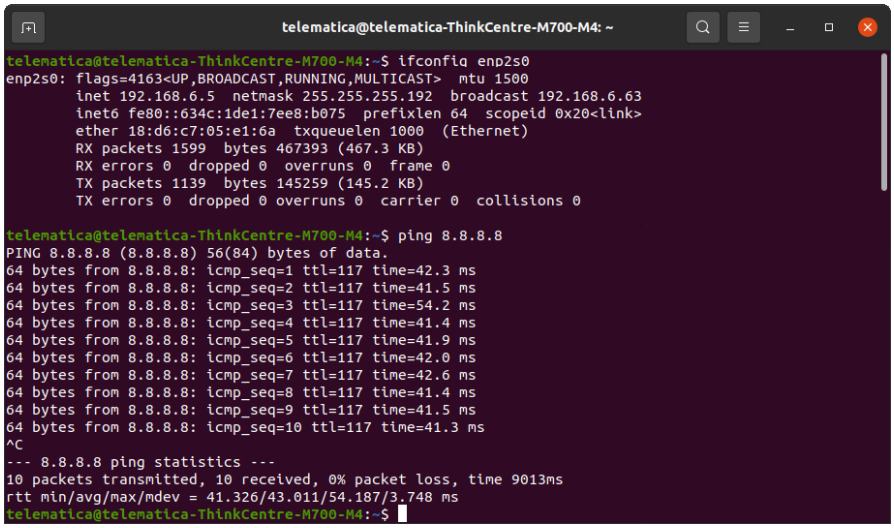


Figura 72. Dirección IP y verificación de conectividad a Internet en el nodo de control.

IV.4.1.2. Diseño de red independiente

No obstante, es conocido que las redes de Internet empresariales como la de la universidad, debido al gran número de elementos que las conforman y la dependencia de un proveedor de servicio de Internet (*Internet Service Provider*, ISP) externo, hace que esta red tenga fluctuaciones en la conectividad de cada elemento que la conforma, y la aplicación *MICROSTACK* requiere que los equipos estén conectados y activos para funcionar correctamente y no generar fallas.

Adicionalmente de la red de DTI descrita, el laboratorio cuenta con una infraestructura de red interna más simple y totalmente independiente del DTI, conformada por un cableado

estructurado interno del laboratorio que desemboca en un *patch panel* ubicado en un *rack* que, a su vez, alberga los elementos de red utilizados en las prácticas, tales como *hubs*, *switches* y *routers*.

Para garantizar la independencia de la nube de Internet y las intermitencias descritas de la red universitaria, se conectaron las computadoras del laboratorio a ambas infraestructuras de red; la de DTI para proveer conectividad a Internet a la red independiente del laboratorio para el montaje de la nube sobre la misma. La conexión de las computadoras de la nube se realizó por medio de un *switch* CISCO *Catalyst* 3750 *Series* y diez cables *Ethernet* (*patch cords*), de los que, cinco de ellos, conectan las computadoras al cableado estructurado hasta el *patch panel*, y los otros cinco, conectan el *patch panel* con el *switch*, tal como se diagrama y se muestra a continuación (Figura 73 y Figura 74):

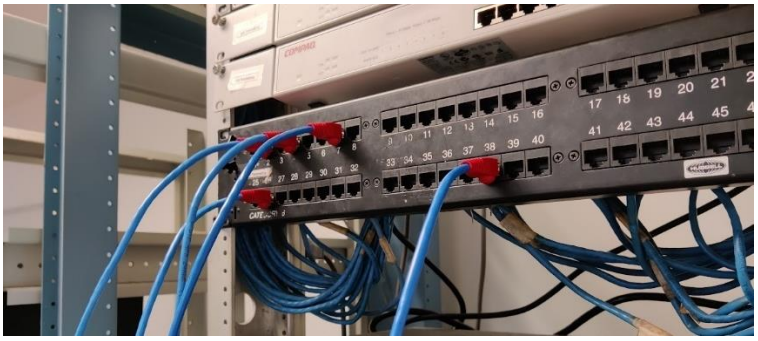


Figura 73. Conexión de los nodos al *patch panel*.

Figura 74. Conexión de los nodos al *switch*.

IV.4.1.2.1. Direccionamiento IP

Para lograr la comunicación entre las computadoras de la nube en la red interna, se realizó un breve direccionamiento IP en los cinco nodos que conforman el clúster. Todas las computadoras cuentan con dos interfaces de red denominadas “eno1” y “enp2s0”, y fueron utilizadas para la conexión interna de la nube y para la conexión a Internet respectivamente. El direccionamiento IP para la conexión a Internet se realizó de forma automática por medio del servidor DHCP

perteneciente al DTI, mientras que el direccionamiento de la red interna se realizó de forma estática relacionando cada dirección con el número con el que cada máquina se identifica en el laboratorio.

Se utilizó el bloque 192.168.0.0/24 para la red de la nube, en la que cada máquina tenía su respectiva dirección, con el último octeto igual al número de máquina multiplicado por diez, como se muestra en la siguiente tabla, donde el nodo de control de la nube es la máquina #4, identificada con el nombre de equipo o *hostname* "telemática-ThinkCentre-M700-M4".

Nombre del equipo	Dirección IP	Tipo de nodo
Telematica-ThinkCentre-M700-M1	192.168.0.10	Cómputo
Telematica-ThinkCentre-M700-M2	192.168.0.20	Cómputo
Telematica-ThinkCentre-M700-M4	192.168.0.40	Control
Telematica-ThinkCentre-M700-M5	192.168.0.50	Cómputo
Telematica-ThinkCentre-M700-M6	192.168.0.60	Cómputo

Tabla 1. Nombres de los nodos, sus direcciones IP y su rol.

IV.4.2. Descarga e inicialización de MICROSTACK

De igual forma que en la prueba de la "micro nube", el primer paso fue la instalación de *MICROSTACK* en cada una de las computadoras que conforman la nube, tanto en el nodo de control como en los nodos de cómputo. Se instaló con el mismo comando empleado en la "micro nube":

```
$ sudo snap install microstack --beta
```

IV.4.2.1. Nodo de control

En el nodo de control, luego de instalar la aplicación, se procedió a inicializarla en modalidad de nodo de control. A diferencia del nodo de control empleado en la "micro nube" (LuisDavid-Desktop), las computadoras del laboratorio fueron conectadas a dos redes, por lo que la inicialización de *MICROSTACK* debió hacerse de forma manual por medio del comando:

```
$ sudo microstack init
```

Luego, el *Shell* pregunta si el rol de la computadora es de control o cómputo y solicita la dirección IP de la misma, datos respondidos con la información correspondiente en cada máquina, como se muestra a continuación.

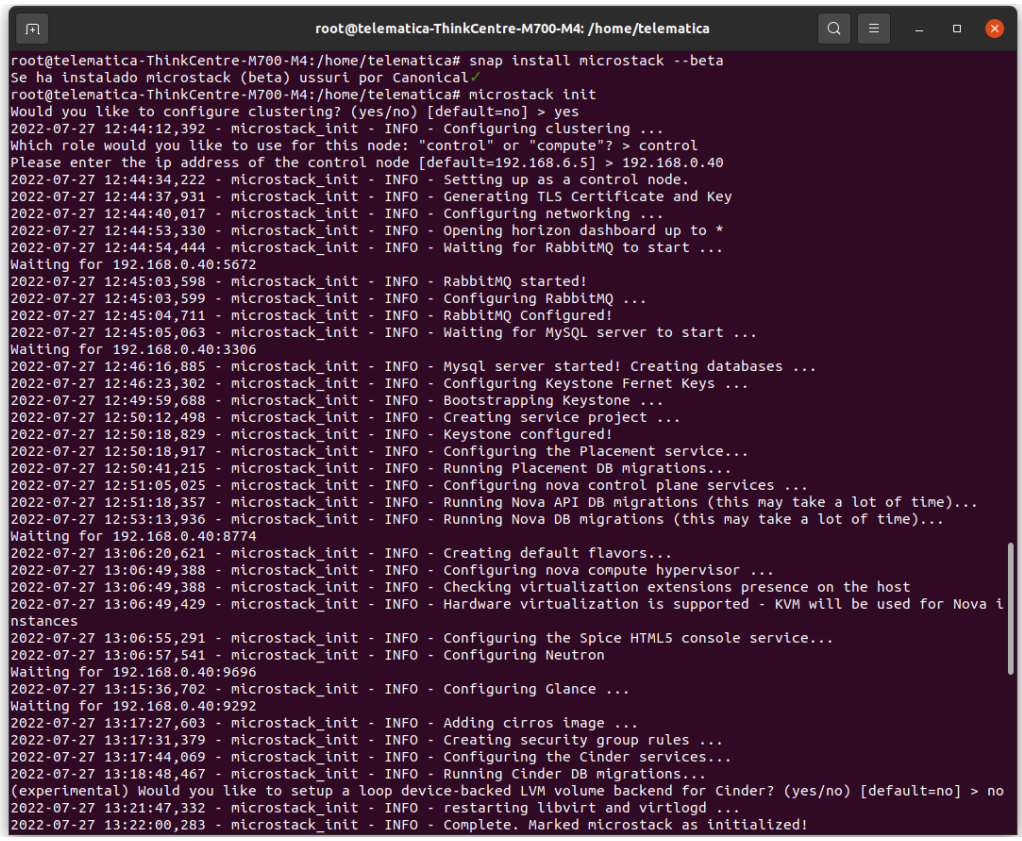


Figura 75. Nodo de control de la nube inicializado.

IV.4.2.2. Nodos de cómputo

Los nodos de cómputo fueron inicializados de forma manual, a diferencia del nodo de cómputo empleado en la “micro nube” (macxy-desktop); debido a que, al igual que el nodo de control de la red del laboratorio, estos están conectados a dos redes distintas.

Para realizar este paso, de igual forma que en la “micro nube” se generaron cuatro cadenas de conexión (*connection strings*) en el nodo de control para adjuntar los cuatro nodos de cómputo a él y conformar el clúster de la nube de múltiples nodos. Las cadenas de conexión se generaron en el nodo de control por medio del comando:

```
$ sudo microstack add-compute
```

Después, en cada nodo de cómputo, se inició *MICROSTACK* de forma manual, introduciendo la cadena de conexión respectiva para cada uno de ellos, luego de tipear el rol de nodo de cómputo y la dirección IP de los mismos, como se muestra a continuación en el caso de la máquina #1.

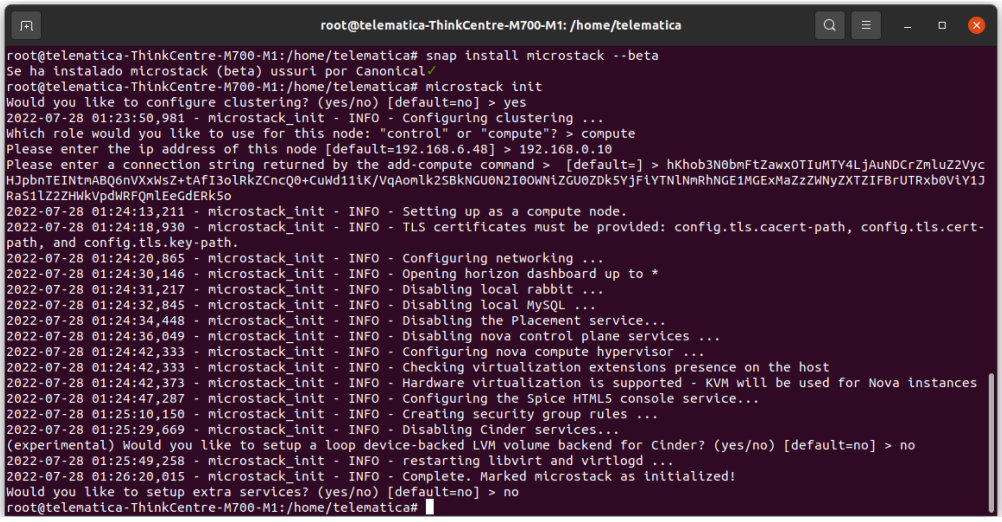


Figura 76. Máquina#1 inicializada como nodo de cómputo.

El proceso descrito fue replicado en las máquinas #2, #5 y #6 del laboratorio, con lo que se conformó un clúster final de un total de cinco máquinas.

La generación de cadenas y adjuntado de cada uno de los nodos de cómputo se observa en la siguiente figura:

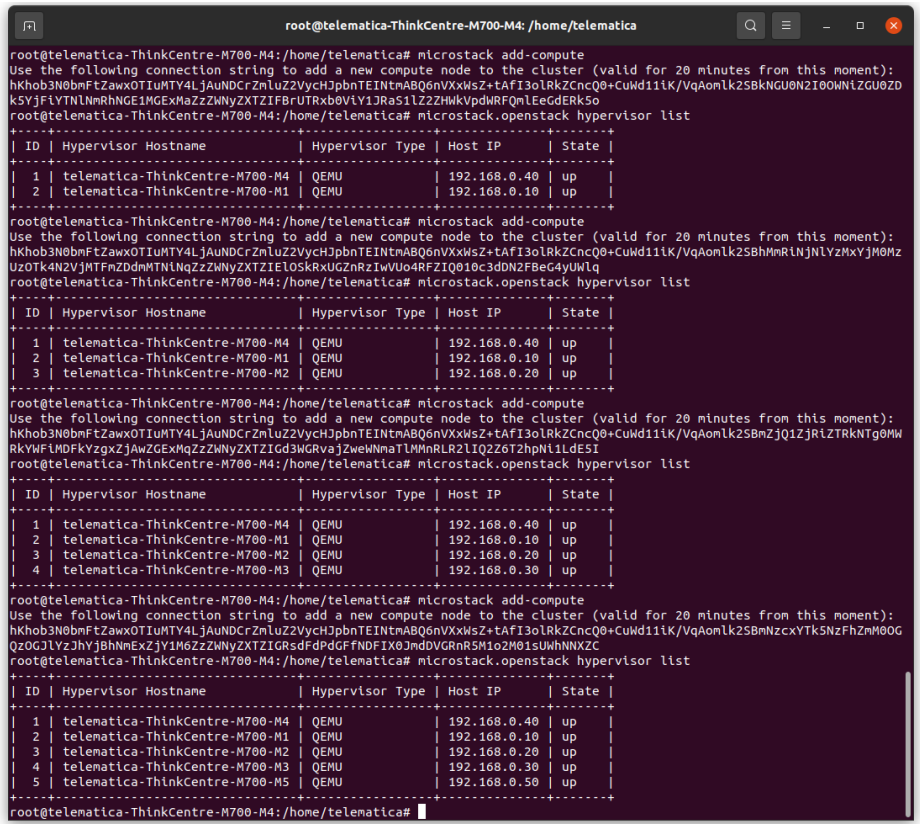


Figura 77. Generación de cadenas y adición de cada nodo de cómputo.

Una vez concluida la creación del clúster de la nube con cada uno de sus nodos, al igual que en el desarrollo de la "micro nube", se generó una contraseña de acceso al *dashboard* para el usuario administrador por defecto de *OPENSTACK*, "*admin*", con la que se accedió al mismo de forma satisfactoria.

A partir de la presente sección del documento, se cambió el tema personalizado de colores del *dashboard* de *OPENSTACK* por defecto, al tema UBUNTU, debido a que, visualmente, resulta más cómodo a la vista y más fácil de leer e interpretar los datos, a criterio de los autores del presente tomo.

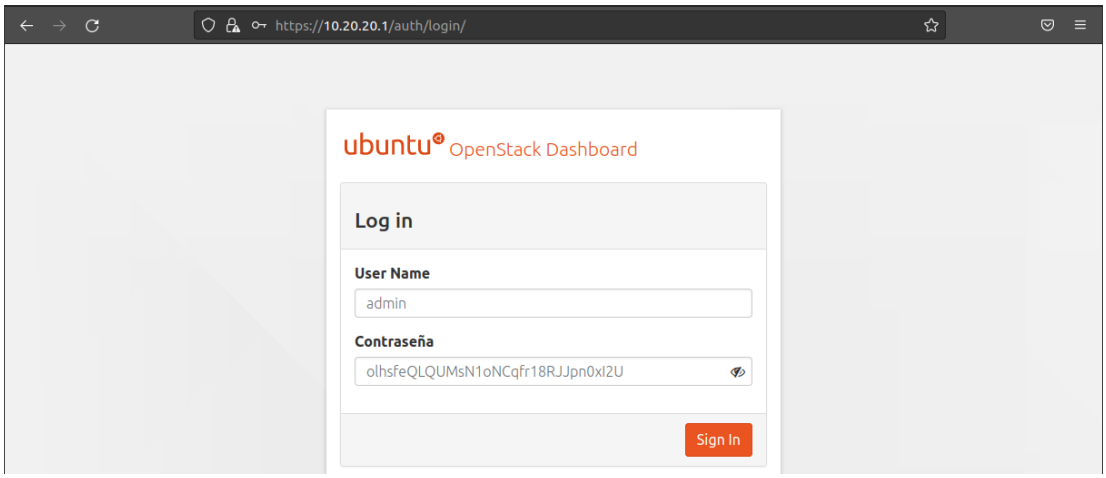


Figura 78. Acceso al *dashboard* desde nodo de control del laboratorio.

Desde el *dashboard*, se pueden visualizar los hipervisores y los servicios de cómputo de forma independiente, además del sumario de las memorias principales y secundarias como un total de *hardware* disponible para la gestión de la nube.

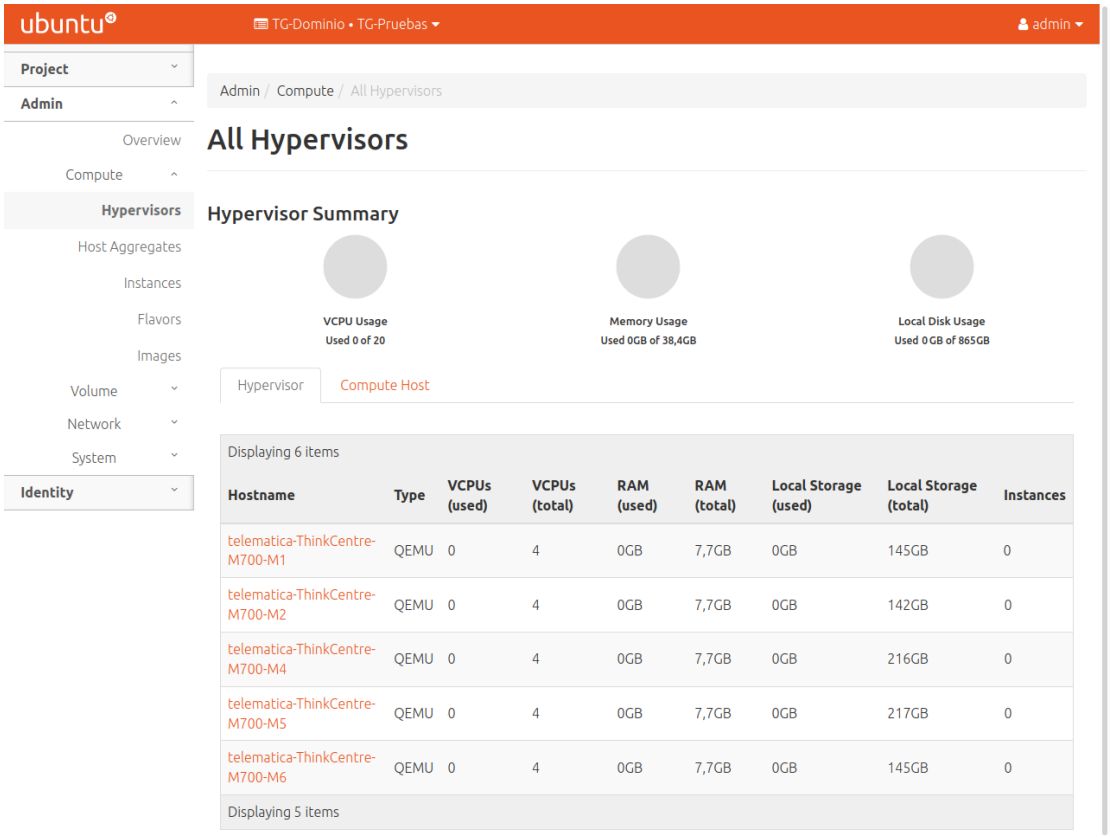


Figura 79. Hipervisores de la nube vistos desde el *dashboard*.

IV.4.3. Despliegue final

Ya con *MICROSTACK* operativo en todos sus nodos, se desarrolló un amplio despliegue de la nube del Laboratorio de Telemática, similar al despliegue realizado en la "micro nube". Este despliegue consiste en la creación de un entorno y ambiente acondicionado para la virtualización de equipos y funciones de red y obtención de múltiples imágenes para sistemas operativos.

El despliegue fue desarrollado siguiendo las siguientes fases:

IV.4.3.1. Creación de plantillas de imágenes y flavors

Adicionalmente a la imagen por defecto de *CirrOS*, se descargaron imágenes de los siguientes sistemas operativos desde sus respectivos repositorios oficiales de cada sistema, accesibles desde la página oficial de *OPENSTACK* [31]:

- UBUNTU 20.04.3 LTS [28].
- UBUNTU 18.04.3 LTS [28].
- Debian 10 [32].

- CentOS Stream 9 [33].
- Fedora 36 [34].
- Windows 10 [35].

Dichas imágenes fueron cargadas a *Glance*, gestor de imágenes de *OPENSTACK*, pueden observarse a continuación:

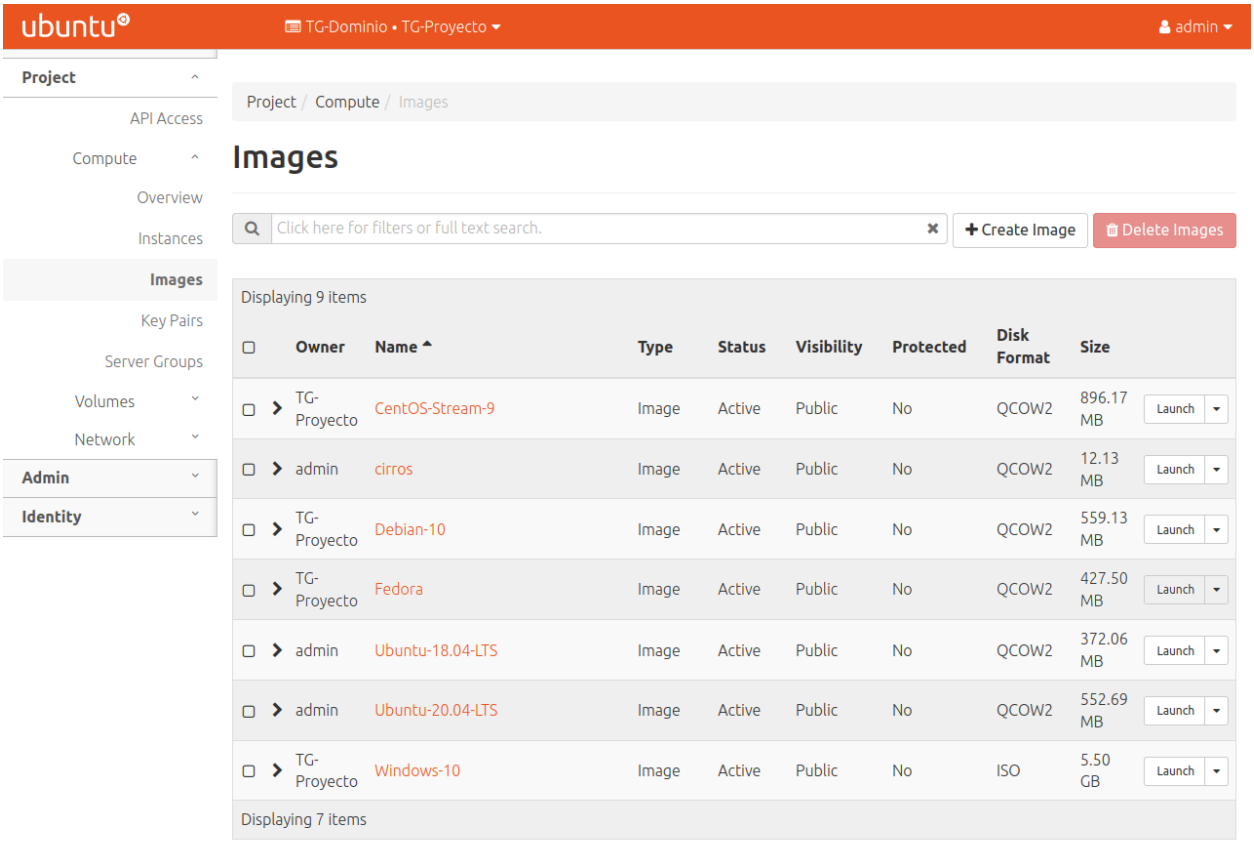


Figura 80. Imágenes descargadas para la creación de instancias.

IV.4.3.2. Identidades del despliegue

IV.4.3.2.1. Dominios

Al igual que en el despliegue de la “micro nube”, se llevó a cabo la creación de un nuevo dominio para la gestión de la nube, denominado “TG-Dominio”, en el que se llevaron a cabo todas las pruebas posteriores de la nube. Este dominio fue creado desde la consola, como se muestra a continuación:

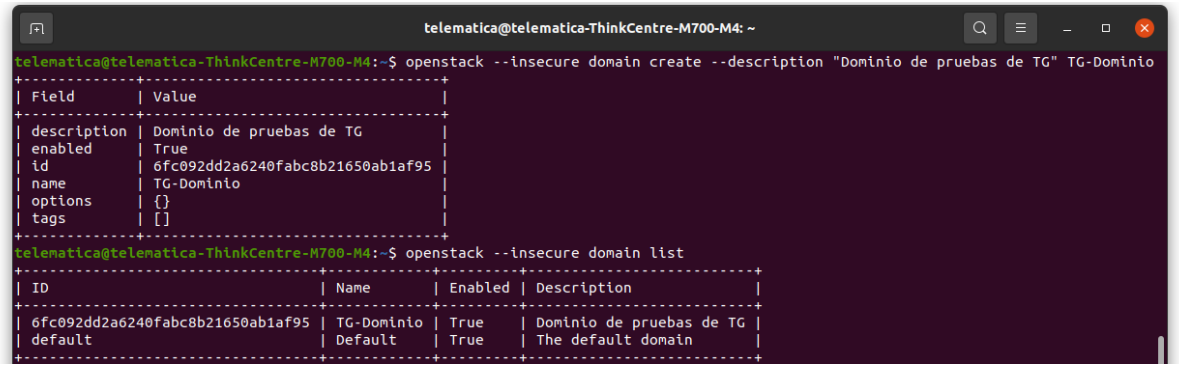


Figura 81. Creación de "TG-Dominio".

Este dominio, requiere de al menos un usuario que lo opere, por lo que se creó primeramente el usuario administrador "*admin*", con el que se llevó a cabo el resto de las posteriores pruebas, por su mayor cantidad de privilegios y opciones, por poseer el rol de administrador. De igual forma que en la "micro nube", se habilitó el soporte multi-dominios en la nube del laboratorio.

Figura 82. TG-Dominio, usuario administrador y soporte multi-dominios.

Adicionalmente, para efectos de probar la gestión de más identidades, se creó un usuario adicional denominado "TG-Usuario", con el rol de "TG-Miembro", sin los privilegios de usuarios administradores.

IV.4.3.2.2. Proyectos

Posteriormente, se procedió a crear el proyecto en el que se desarrollaron el resto de las pruebas descritas, denominado "TG-Proyecto" que, a diferencia de los dominios, pudo ser creado por medio del *dashboard*.

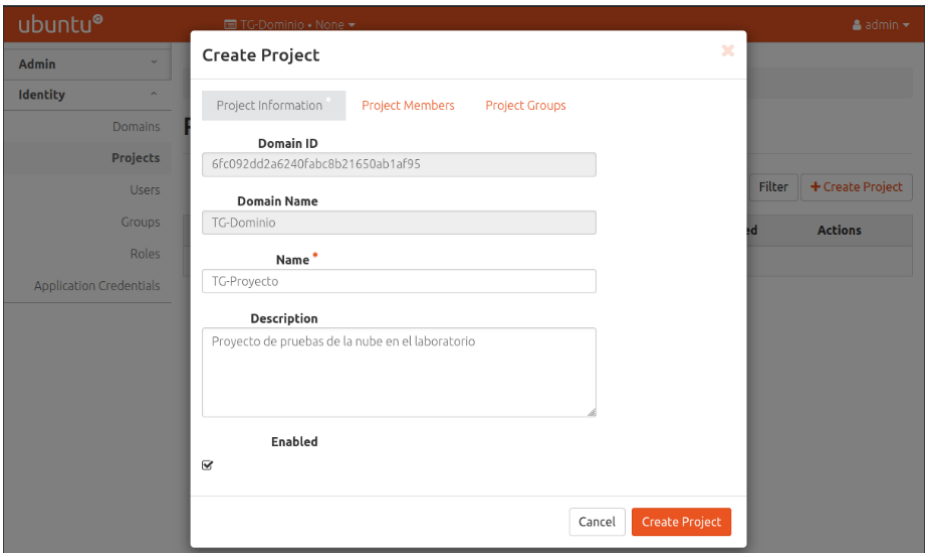


Figura 83. Creación de TG-Proyecto.

IV.4.3.2.3. Acceso con nuevas credenciales

Una vez creadas estas nuevas identidades, usuarios, proyecto y dominio nuevos, se procedió a desconectarse y a volver a acceder al *dashboard* con las nuevas credenciales:

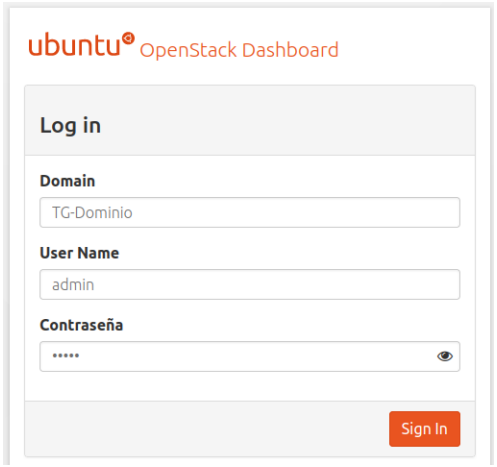


Figura 84. Acceso al *dashboard* con las nuevas credenciales, nótese el campo de especificación del dominio.

IV.4.3.2.4. Gestión de parámetros de redes

Previamente a la creación de las redes, *routers* e instancias para las pruebas, se crearon un conjunto de parámetros que permiten la completa operatividad y acceso remoto a las instancias por medio del protocolo SSH.

Llaves de acceso SSH: se crearon un total de tres llaves de acceso remoto SSH para las instancias, denominadas “TG-KeyPair1”, “TG-KeyPair2” y “TG-KeyPair3”, de las cuales, se descargaron sus

archivos de extensión “pem” correspondientes y fueron cambiados sus permisos de lectura y escritura, al igual que en la sección anterior, por medio del comando “chmod”.

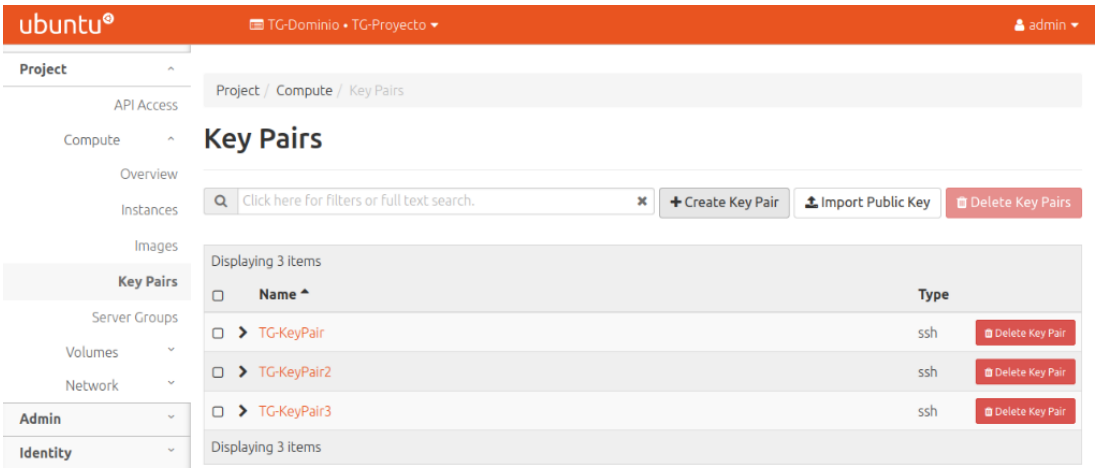


Figura 85. Llaves totales de acceso SSH.

Grupos de seguridad: Se creó un nuevo grupo de seguridad denominado “*TG-Security-Group*”, al que le fue añadido la regla de acceso remoto SSH, al igual que en el despliegue de la “micro nube”.

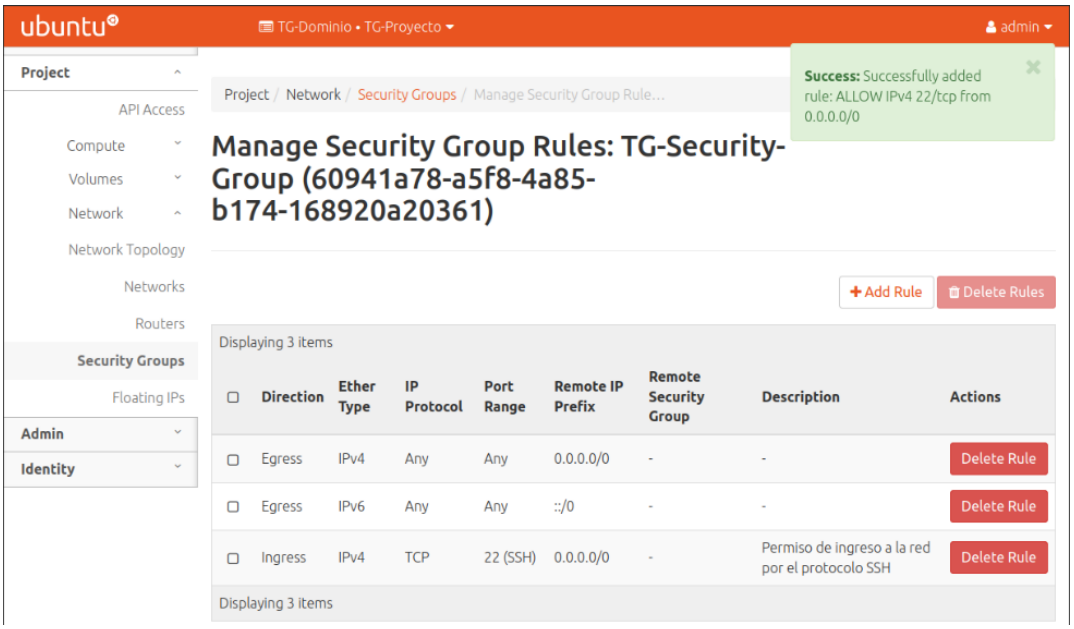


Figura 86. Regla de acceso SSH añadida a "*TG-Security-Group*".

Luego de haber creado este conjunto de herramientas y funciones de red virtuales, el entorno de la nube quedó completamente acondicionado para desplegar topologías con instancias funcionales y operativas para la realización de pruebas de funcionalidad.

IV.5. Fase de realización de pruebas.

IV.5.1. Pruebas de rendimiento

Para probar el rendimiento de la infraestructura de la nube, se seleccionaron tres parámetros que permiten estudiar esta característica, los cuales son:

- Tiempo de creación de instancias.
- Tiempo de encendido de instancias.
- Latencias de instancias y los nodos físicos de la nube.

Se diseñó una topología para la realización de estas pruebas, conformada por cinco instancias, utilizando el dominio *default* esta vez, como se muestra a continuación:

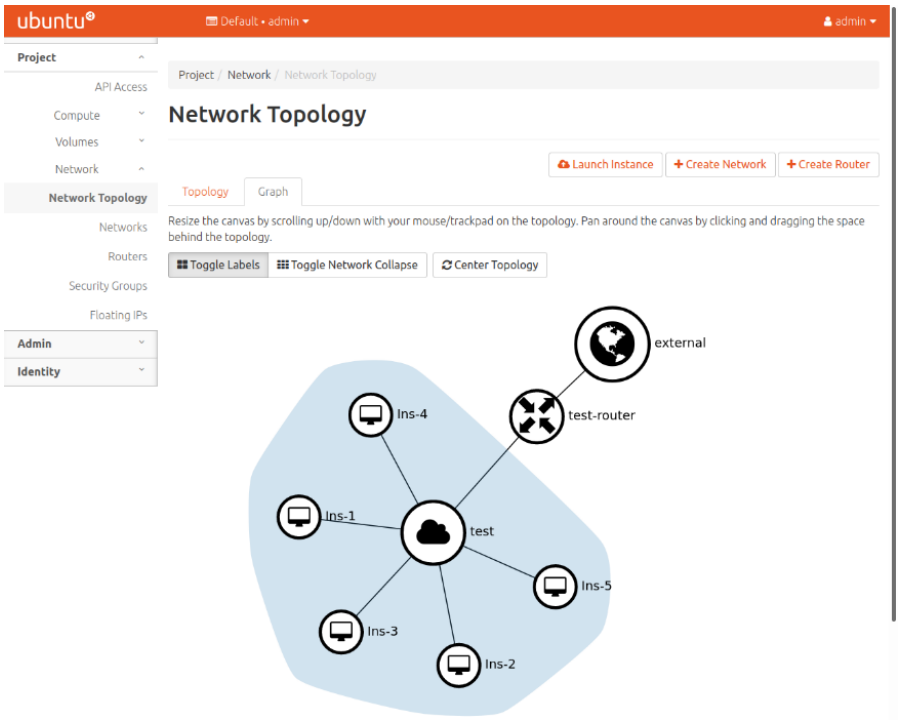


Figura 87. Topología para las pruebas de rendimiento.

Instancia	*Flavors*		Red
1	TG-Flavor	m1.small	"test"; 192.168.222.0/24
2			
3			
4			
5			

Tabla 2. Especificaciones para las pruebas de rendimiento.

Pese a que la topología mostrada es la misma para cada una de las pruebas, fue replicada con algunas diferencias de acuerdo al tipo de parámetro que se midió, se explican a continuación:

IV.5.1.1. Tiempo de creación de instancias

El tiempo de creación de instancias fue estudiado en dos casos: cuando son creadas en el nodo de control y cuando son creadas en nodos de cómputo. Para ambos casos se midieron los tiempos utilizando dos *flavors* de configuración para cada instancia: "TG-Flavor" y "m1.small". Adicionalmente, se repitió esta prueba con la aplicación *VirtualBox*, con el fin de comparar el rendimiento de ambas implementaciones en términos de creación de instancias; de igual forma, se emularon las características de imagen y *flavors* de las instancias medidas en *MICROSTACK*, para que comparar en igualdad de condiciones, los resultados se muestran en el capítulo siguiente.

IV.5.1.2. Tiempo de encendido de instancias

Luego de crear las instancias de la prueba anterior, fueron apagadas y encendidas nuevamente, tomando el tiempo que tardó cada una en encender de acuerdo a su creación en el nodo de control como en un nodo de cómputo, con el objetivo de medir el tiempo de demora debido a la limitación de la capacidad del canal físico para la transmisión de datos entre cada nodo mientras transmite procesos de cómputo, que requieren de velocidades sustancialmente superiores, propias de las memorias de las computadoras. Al igual que el parámetro anterior, el tiempo de encendido también fue medido con las instancias creadas en VirtualBox, con el fin de extender la comparación hasta este tópico. Los resultados se describen en el próximo capítulo

IV.5.1.3. Latencias

Luego de crear y encender las instancias, se eliminaron algunas de ellas y se replicó la topología con instancias creadas en distintos nodos de la nube, incluyendo el nodo de control, esto permitió medir la latencia entre las instancias en relación al nodo donde fueron creadas por medio del envío y recepción de paquetes ICMP, de acuerdo a los cuatro siguientes casos:

- Entre instancias creadas en el nodo de control.
- Entre instancias creadas en el nodo de control y un nodo de cómputo.
- Entre instancias creadas en el mismo nodo de cómputo.
- Entre instancias creadas en nodos de cómputo distintos.

Los resultados de las mediciones fueron plasmados en el próximo capítulo.

IV.5.2. Prueba de enrutamiento estático

Por defecto, *OPENSTACK* cuenta con las herramientas necesarias para el despliegue de topologías que requieran de la creación de rutas estáticas para el establecimiento de la conexión y comunicación entre sus redes y dispositivos. *Neutron*, módulo gestor de las funciones de red de *OPENSTACK*, permite la creación de *routers* virtuales a los que se le pueden añadir las interfaces que el usuario necesite para cada subred y permite el establecimiento de rutas estáticas de forma nativa por medio del *dashboard* de forma interactiva y más amigable para el usuario, en comparación a las rutas que se establecen por medio de la consola de comandos.

Una topología de ejemplo que requiere de establecimiento de rutas estáticas es la que se muestra a continuación:

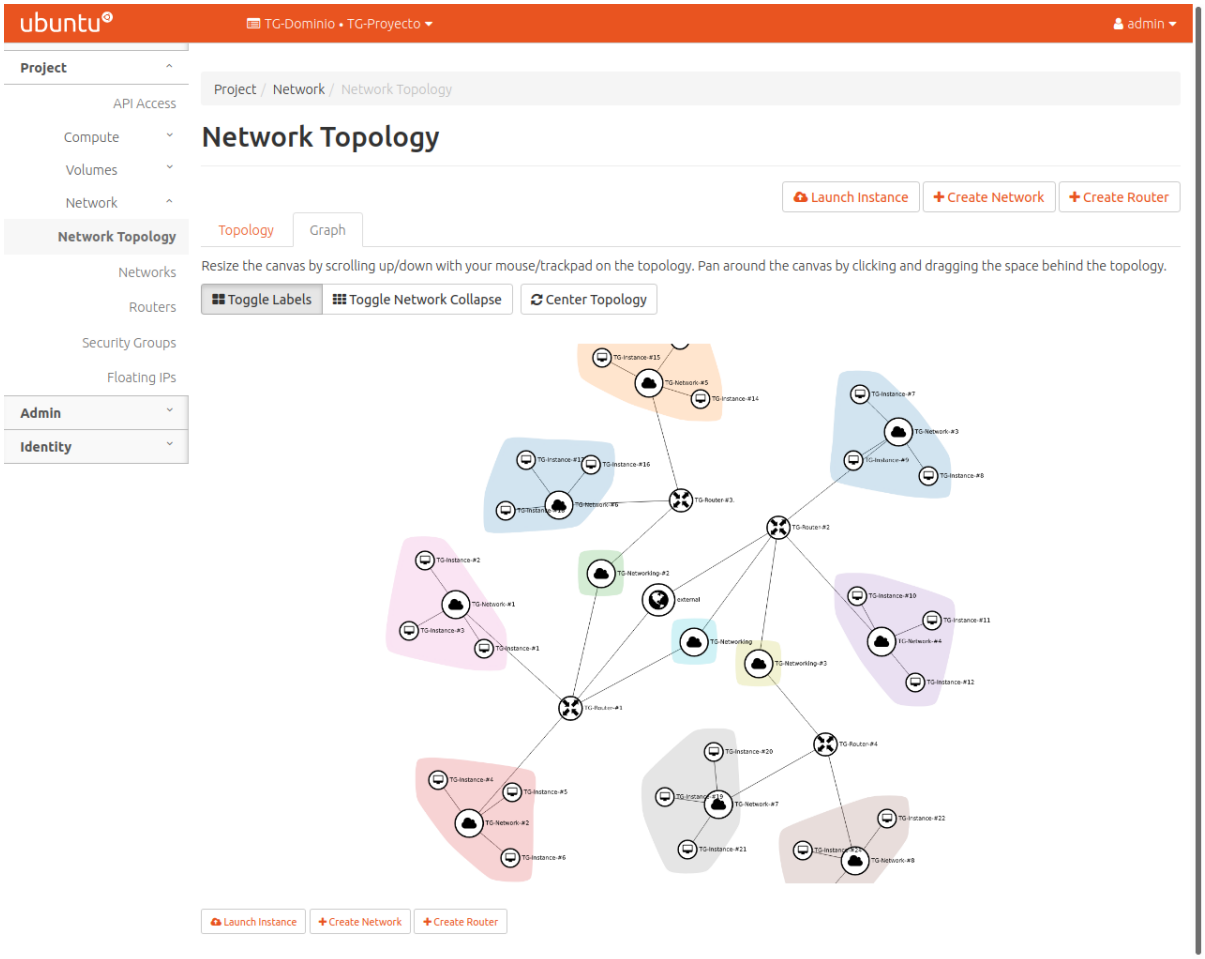


Figura 88. Topología de prueba de enrutamiento estático.

Para la comunicación entre todos los dispositivos que conforman la red, se aplicó el protocolo de enrutamiento estático por medio del *dashboard* de *OPENSTACK* por medio de la ruta del tablero:

Project → *Networks* → *Routers* → <*Nombre del router*> → *Static Routes*

Se establecieron diferentes tipos de rutas en los cuatro *routers* que sirven para verificar la practicidad del protocolo en la red sin afectar la comunicación:

Para el *router* "TG-Router-#1"

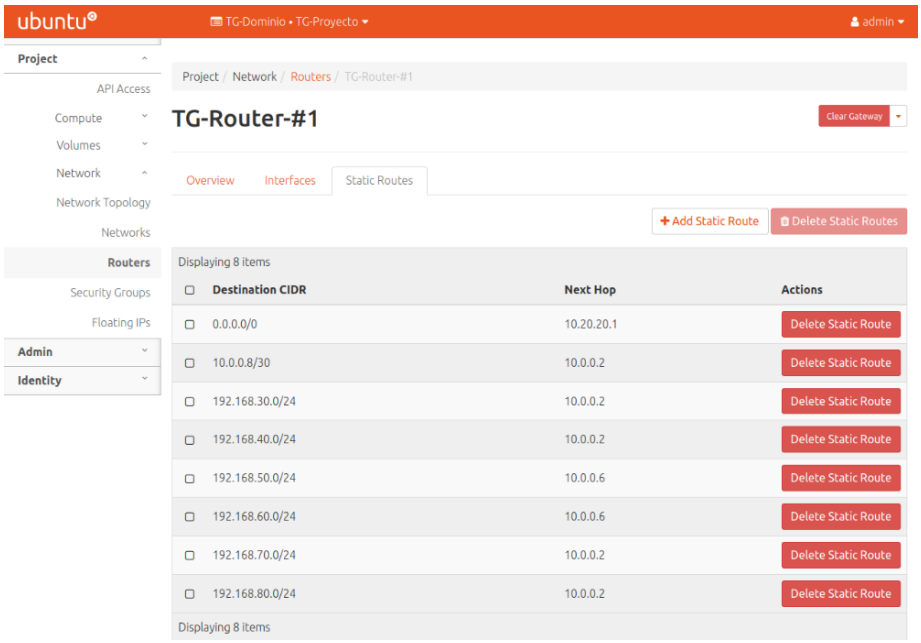


Figura 89. Establecimiento de rutas estáticas en "TG-Router-#1".

Para el *router* "TG-Router-#2"

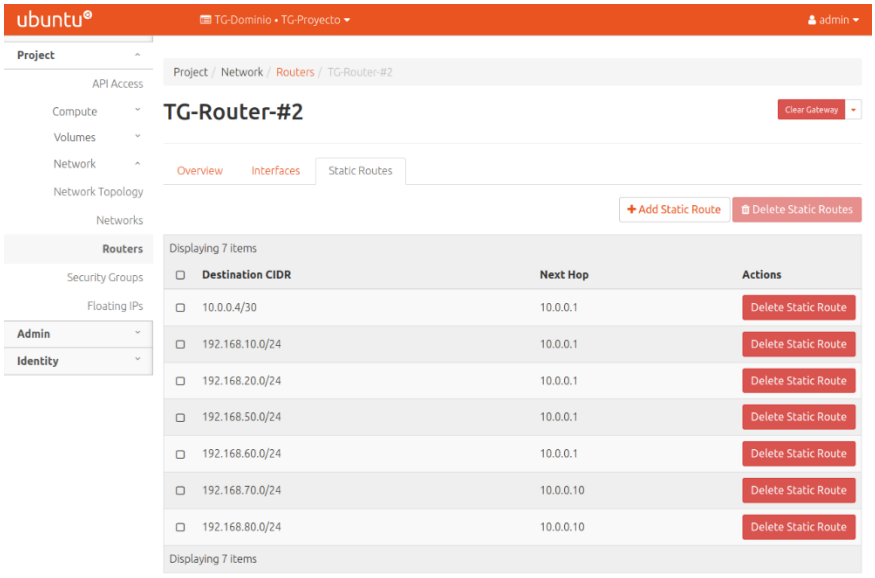


Figura 90. Establecimiento de rutas estáticas en "TG-Router-#2".

Para el router “TG-Router-#3”

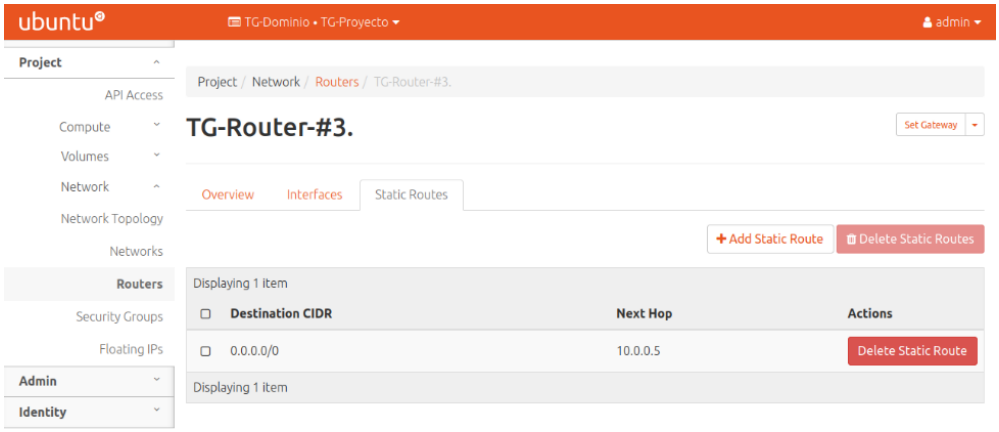


Figura 91. Ruta estática para el *router* "TG-Router-#3".

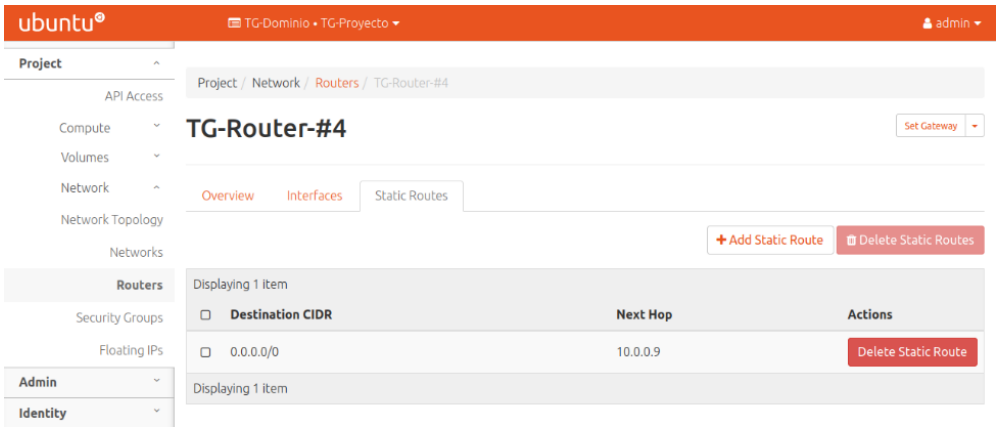


Figura 92. Rutas estáticas para el *router* "TG-Router-#4”.

Nota: sólo es posible establecer rutas estáticas por medio de la dirección IP de siguiente salto.

La comunicación ICMP de los elementos que conforman la topología, luego de esta converger, se muestra en el próximo capítulo.

IV.5.3. Prueba de enrutamiento dinámico

Por defecto, *Neutron*, no cuenta con una función disponible en los *routers* virtuales que permita ejecutar protocolos de enrutamiento dinámico en los proyectos que se desarrollen desde la infraestructura de la nube; por este motivo, para añadir esta funcionalidad, se trabajó con instancias que funcionan como enrutadores, equipadas con la imagen del sistema operativo Debian 10, a las que se le añadió el paquete de *software* de protocolos de enrutamiento *Free Range Routing* (FRR), una mejora del *fork* del proyecto original GNU *Zebra Quagga*, desarrollado por Kunihiro Ishiguro entre 1996 y 2005, permite ejecutar protocolos de enrutamiento tales como BGP, OSPF, RIP, IS-IS [36] [37], entre otros, de los cuales, se escogió OSPF (*Open Shortest Path First*).

OSPF es un protocolo de direccionamiento de paquetes de red clasificado como un protocolo de enlace-estado, y se basa en el algoritmo de la primera vía más corta (*shortest path first*, SPF) [38].

IV.5.3.1. Configuración en el nodo de control

Para implementar el enrutamiento dinámico, las instancias que cumplieron esta función fueron equipadas primeramente con el *software* Quagga y posteriormente FRR, debido a que son de idéntica configuración, por medio de una descarga de paquetes “apt” desde internet por medio de la consola de comandos, se mostrará el proceso con FRR. Previo a dicha descarga, se provisionó acceso a internet a las instancias de la nube por medio de la habilitación de dos funciones en el nodo de control: Función de *postrouting* y enmascaramiento NAT en el nodo de control, para que el mismo redirija el tráfico con destino a la nube por medio de su interfaz virtual “br-ex” que es la puerta de enlace de la misma. Y la función de *IP-forwarding*, que permite a un nodo que redirigir el tráfico entrante por otra interfaz de salida distinta, función que cumplen por defecto los enrutadores o *routers*.

Ambas funciones fueron añadidas por medio de la consola del nodo de control, a través de los siguientes comandos (Figura 93):

```
$ sudo iptables -t nat -A POSTROUTING -s 10.20.20.1/24 ! -d 10.20.20.1/24 -j MASQUERADE

$ sudo sysctl net.ipv4.ip_forward=1
```

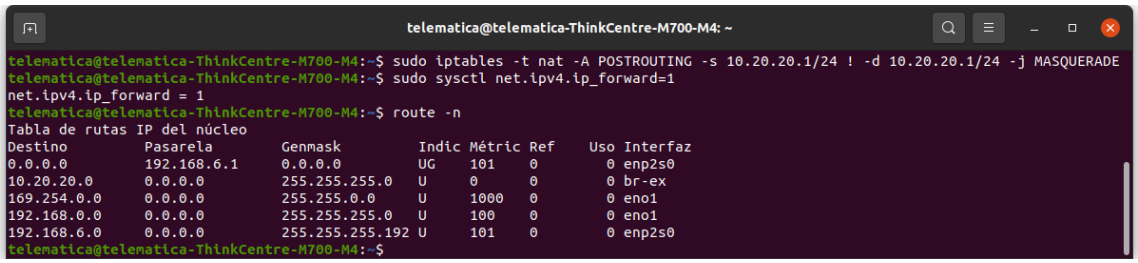


Figura 93. Adición de reglas de conexión para *postrouting* e *IP forwarding* al nodo de control.

IV.5.3.2. Descarga y configuración de la aplicación en las instancias.

Luego, se creó una primera instancia a la que le fue descargada la aplicación FRR por medio de los comandos de consola:

`# apt-get update:` Obtiene las últimas versiones disponibles de los paquetes apt en los repositorios de *software* de la distribución y en repositorios de terceros que hayan sido añadidos, sin embargo, no las descargará [39].

`# apt-get upgrade:` Descarga e instala las mejoras de los paquetes apt obsoletos y sus dependencias del sistema que fueron obtenidas con el comando anterior.

`# apt-get install FRR:` Instala en el disco duro de la máquina el paquete FRR especificado y crea los archivos y ficheros embebidos en el mismo.

IV.5.3.2.1. Configuración de los daemons y del IP Forwarding

Los *daemons* son módulos embebidos en una aplicación, FRR cuenta con un total de 14 *daemons* que, por defecto, vienen deshabilitados, como se muestra a continuación, en el archivo "/etc/frr/daemons":

```
bgpd=no
ospfd=no
ospf6d=no
ripd=no
ripngd=no
```

```
isisd=no
pimd=no
ldpd=no
mhrpd=no
eigrpd=no
babeld=no
sharpd=no
pbrd=no
bfdd=no
```

Para habilitar la función de OSPF, se cambió el valor de la configuración del *daemon* "ospfd" de "no" a "yes" en el archivo "/etc/frr/daemons", y luego se habilitó la función de *IP Forwarding* en la instancia, para que el tráfico entrante por una interfaz pueda salir por las otras, función esencial en los enrutadores reales. La habilitación del *daemon* OSPF y el *IP forwarding* se muestra en la siguiente figura:

```
root@debian:~# nano /etc/frr/daemons
```

Dentro del archivo, se cambió `ospfd=no` por `ospfd=yes`, se guardaron los cambios.

```
root@debian:~# sysctl net.ipv4.ip_forward=1
```

Posteriormente, se añadieron los puertos TCP/UDP que utilizan los protocolos de enrutamiento que ofrece FRR, en el archivo "/etc/services", como se muestra continuación:

```
zebrasrv    2600/tcp          # zebra service
zebra       2601/tcp          # zebra vty
ripd        2602/tcp          # RIP vty
ripngd      2603/tcp          # RIPngd vty
ospfd       2604/tcp          # OSPFd vty
bgpd        2605/tcp          # BGPd vty
ospf6d      2606/tcp          # OSPF6d vty
ospfapi     2607/tcp          # ospfapi
isisd       2608/tcp          # ISIS vty
babeld      2609/tcp          # BABELd vty
nhrpd       2610/tcp          # nhrpd vty
pimd        2611/tcp          # PIMd vty
```

```
ldpd        2612/tcp        # LDPd vty
eigrpd      2613/tcp        # EIGRPd vty
bfdd        2617/tcp        # bfdd vty
fabricd     2618/tcp        # fabricd vty
vrrpd       2619/tcp        # vrrpd vty
```

IV.5.3.2.2. Archivos de configuración

La descarga del paquete de FRR finaliza creando archivos de configuración de ejemplo para cada uno de los *daemons* y módulos de la aplicación en el directorio "/usr/share/doc/frr/examples/", los *daemons* de interés en este caso son los de los módulos OSPF y VTYSH (*Virtual Terminal Shell*), este último, correspondiente al módulo que integra las funciones de los *daemons* en una sola interfaz de línea de comandos [40]. Estas configuraciones de ejemplo fueron recicladas copiando y pegando los archivos de configuración desde la ubicación anterior, al directorio "/etc/frr/", los archivos copiados fueron "*ospfd.conf*", "*vtysh.conf*" y "*frr.conf*", donde el último, se encarga de escribir en la memoria la configuración establecida en la aplicación.

El copiado de estos archivos de realizó con los comandos que se muestran a continuación:

```
# cp /usr/share/doc/frr/examples/ospfd.conf:
# cp /usr/share/doc/frr/examples/vtysh.conf:
# cp /usr/share/doc/frr/examples/frr.conf:
```

Luego, la instalación fue culminada reiniciando la aplicación, por medio del reinicio del módulo *frr*, por medio del comando de consola: `# service frr restart`

Una vez finalizadas estas configuraciones, la instancia ya está completamente equipada para implementar OSPF. Para no tener que repetir esta configuración en cada instancia que sea enrutadora, se tomó una instantánea o *snapshot* de la instancia actual que permitió crear una imagen de sistema operativo basada en el estado actual su disco duro para usarla en otras instancias. Esto permitió crear las demás instancias enrutadoras automáticamente equipadas con FRR y OSPF, sin necesidad de instalar nuevamente FRR en ellas ni de configurar sus *daemons* y puertos.

IV.5.3.3. Definición de topología de red

La topología para realizar la prueba de enrutamiento dinámico se muestra a continuación (Figura 94). Fue diseñada con la idea de plasmar un caso en el que sea necesario establecer rutas para la comunicación completa de la red, con el objetivo de que las mismas se establezcan con OSPF de forma dinámica.

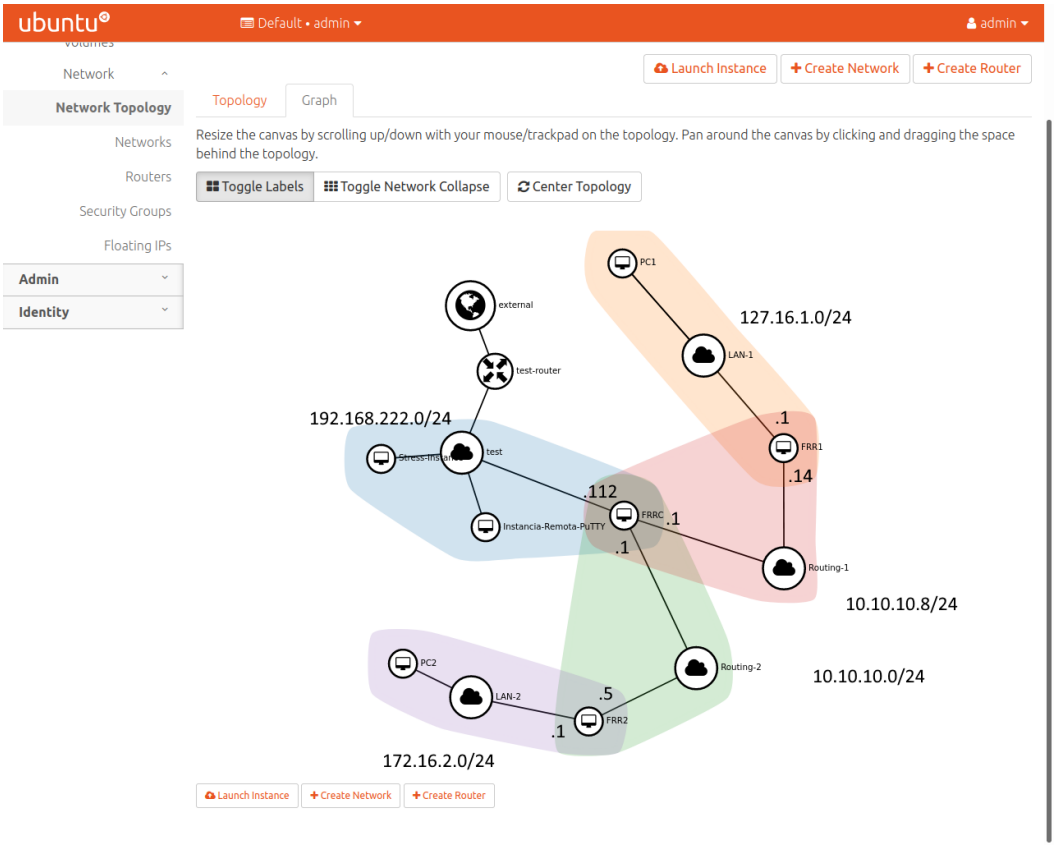


Figura 94. Topología para enrutamiento dinámico.

La descripción de las redes se muestra en la siguiente tabla:

Nombre	CIDR	Máscara en octetos
test	192.168.222.0/24	255.255.255.0
Routing-1	10.10.10.0/29	255.255.255.248
Routing-2	10.10.10.8/29	255.255.255.248
LAN-1	172.16.1.0/24	255.255.255.0
LAN-2	172.16.2.0/24	255.255.255.0

Tabla 3. Redes de la topología para enrutamiento dinámico.

La descripción de cada elemento de la prueba es la siguiente:

Nombre	Direcciones IP	CIDRs que no conoce
FRRC	192.168.222.112 10.10.10.1 10.10.10.9	172.16.1.0/24 172.16.2.0/24
FRR1	10.10.10.5 172.16.1.1	192.168.222.0/24 10.10.10.8/29 172.16.2.0/24
FRR2	10.10.10.14 172.16.2.1	192.168.222.0/24 10.10.10.0/29 172.16.1.1/24

Tabla 4. Descripción de cada dispositivo que interviene en la prueba de FRR.

En la topología mostrada (Figura 94), el *router* FRR sólo conoce las redes "*test*", "*Routing-1*" y "*Routing-2*", debido a que son las que tiene directamente conectadas a sus puertos, para saber redirigir el tráfico a las redes terminales "*LAN-1*" y "*LAN-2*", es necesario que se añadan ambas redes a su tabla de rutas de forma dinámica.

Lo mismo ocurre con FRR1 y FRR2, donde el primero, sólo conoce a la red LAN-1 y Routing-1 y desconoce a "test", "Routing-2" y a "LAN-2"; mientras que el segundo, conoce a "LAN-2" y "*Routing-2*", y desconoce a "test", "*Routing-1*" y a "LAN-1".

IV.5.3.4. Configuración de OSPF en las instancias

Conocidas las redes desconocidas para cada uno de las instancias enrutadoras, se implementó el protocolo OSPF en cada una de ellas usando la aplicación FRR por medio del *Shell* virtual VTYSH. En OSPF, los elementos se reconocen entre sí mediante la publicación de los respectivos CIDRs de las redes que tienen conectadas directamente en sus puertos de red, las redes publicadas en cada una de las instancias enrutadoras fueron los siguientes:

Nombre	CIDR's
FRRC	192.168.222.0/24 10.10.10.0/29 10.10.10.8/24
FRR1	10.10.10.0/29 172.16.1.1
FRR2	10.10.10.8/29 172.16.2.0/24

Tabla 5. Redes publicadas en cada instancia enrutadora con OSPF.

La publicación de redes se realizó en el *Shell* virtual VTYSH de FRR en cada instancia, accediendo al modo de configuración del terminal, configurando de forma redundante las direcciones IP de cada una de sus interfaces y añadiendo los CIDRs de las redes directamente conectadas a cada una de las mismas al "área" 0 de OSPF, como se muestra a continuación en cada instancia:

IV.5.3.4.1. En instancia FRR1

La siguiente figura muestra la configuración de la instancia FRR1:

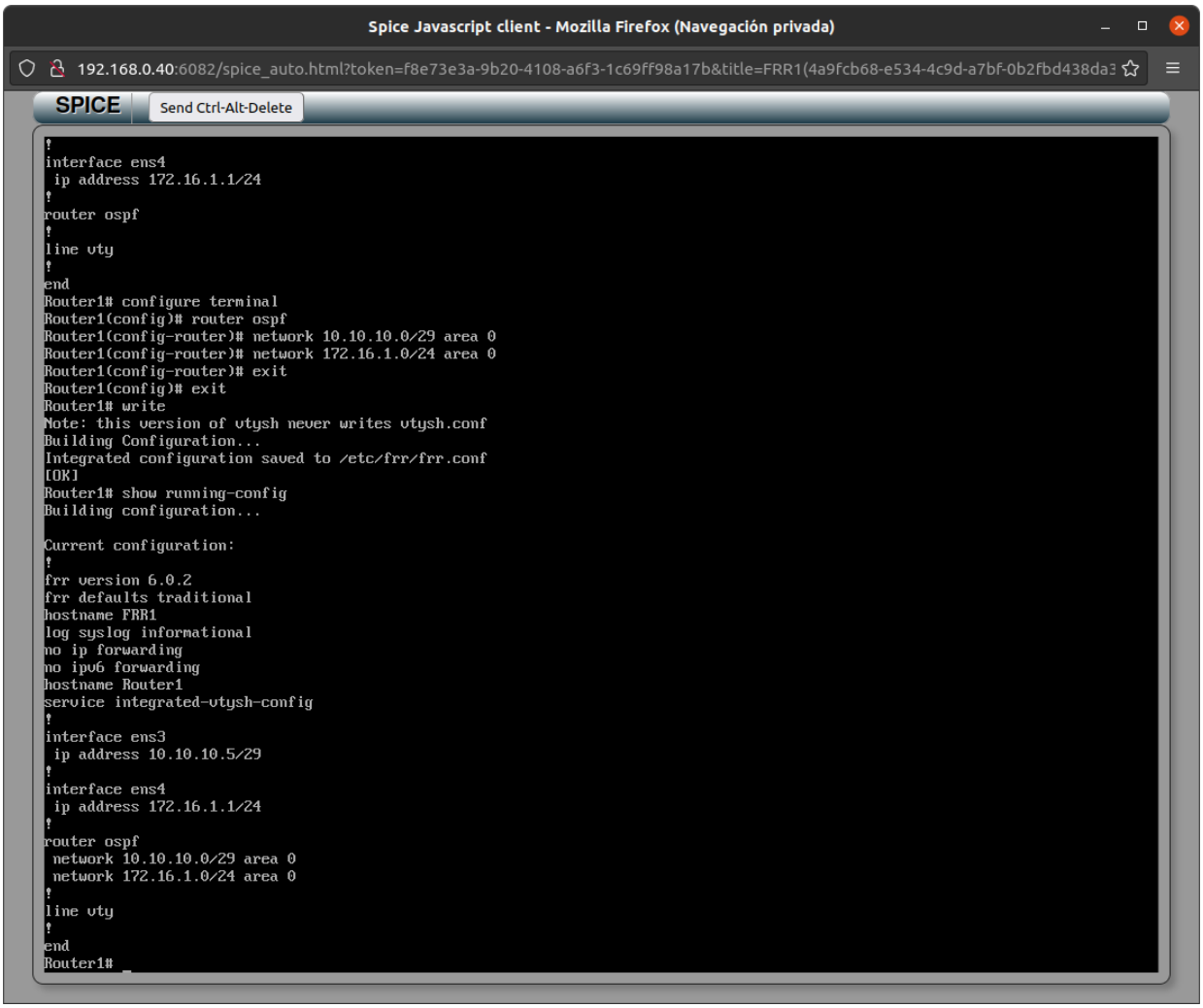


Figura 95. Configuración en VTYSH de FRR1.

IV.5.3.4.2. En instancia FRR2

La siguiente figura muestra la configuración de la instancia FRR2:

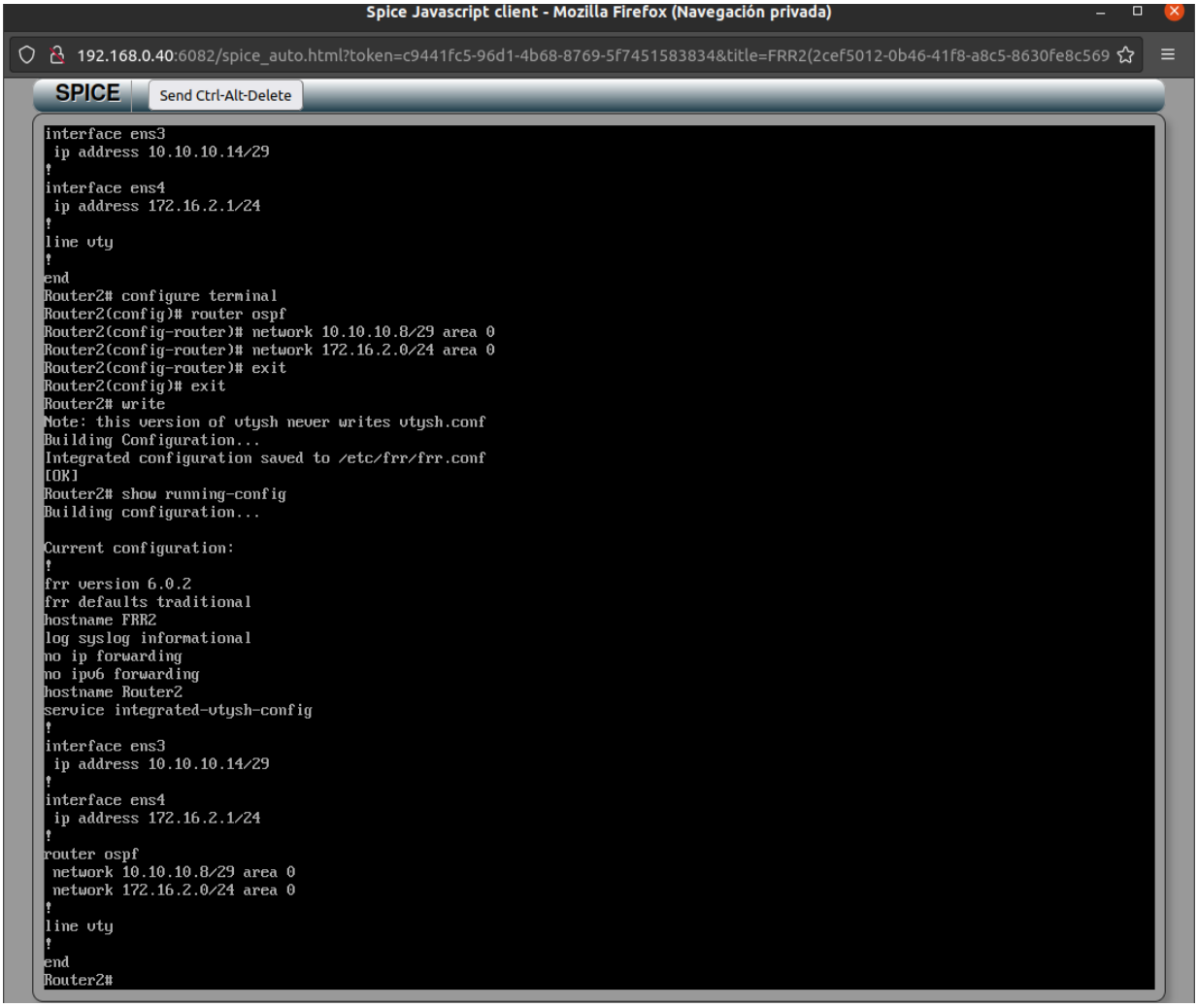


Figura 96. Configuración en VTYSH de FRR2.

IV.5.3.4.3. En instancia FRRC

La siguiente figura muestra la configuración de la instancia FRRC:

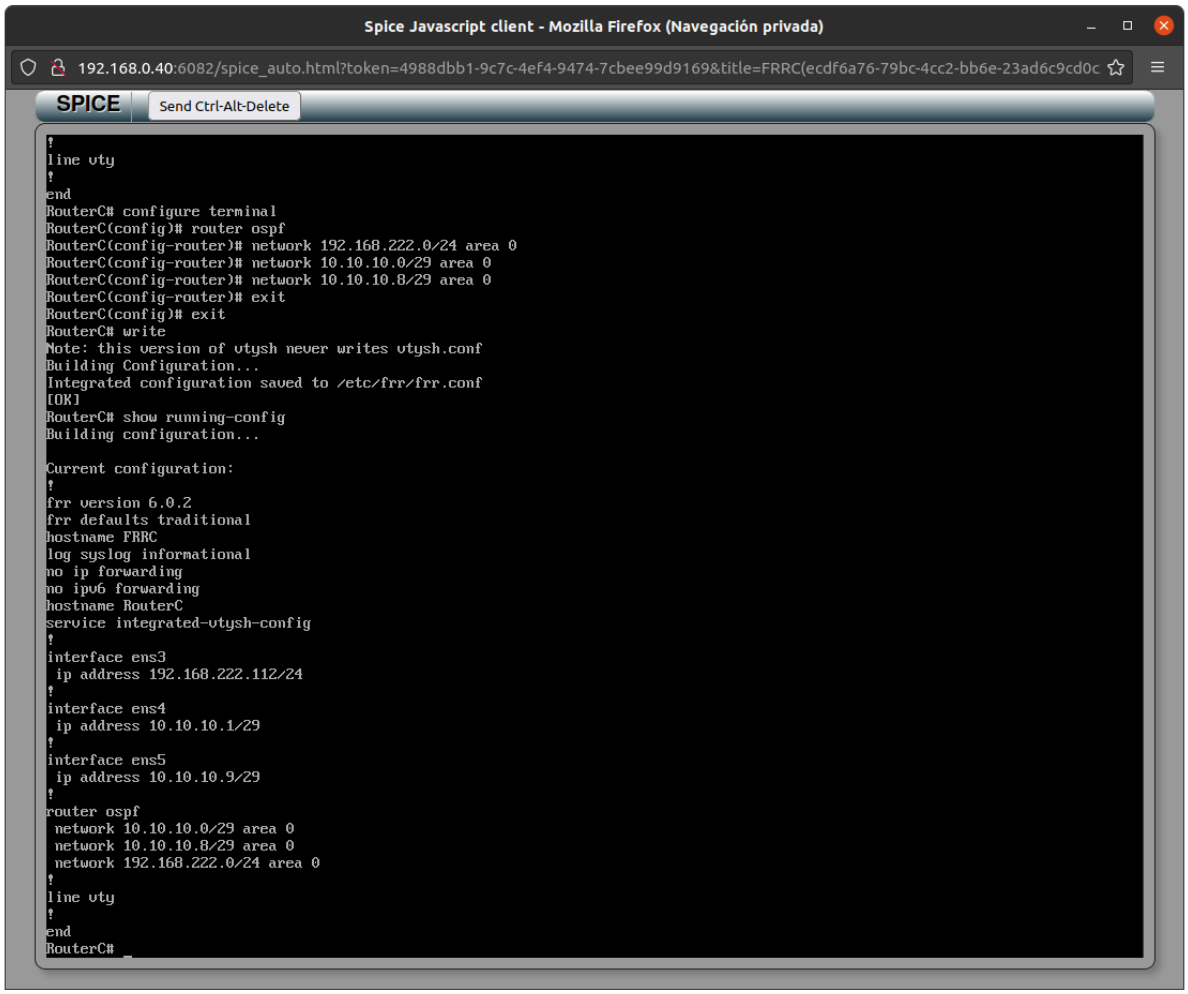


Figura 97. Configuración en VTYSH de FRRC.

IV.5.3.5. Establecimiento de rutas

Luego que estas configuraciones fueron guardadas en las memorias de las instancias, al cabo de unos segundos, se establecieron las rutas dinámicas de las redes desconocidas por cada una de ellas, pueden visualizarse por medio del comando del VTYSH `show ip route`, como se muestra a continuación:

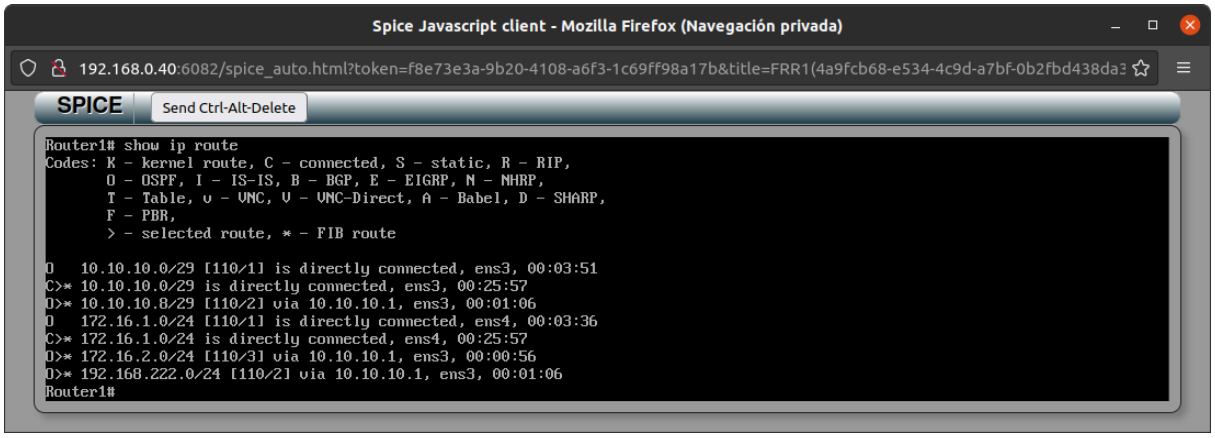


Figura 98. Rutas dinámicas de FRR1.

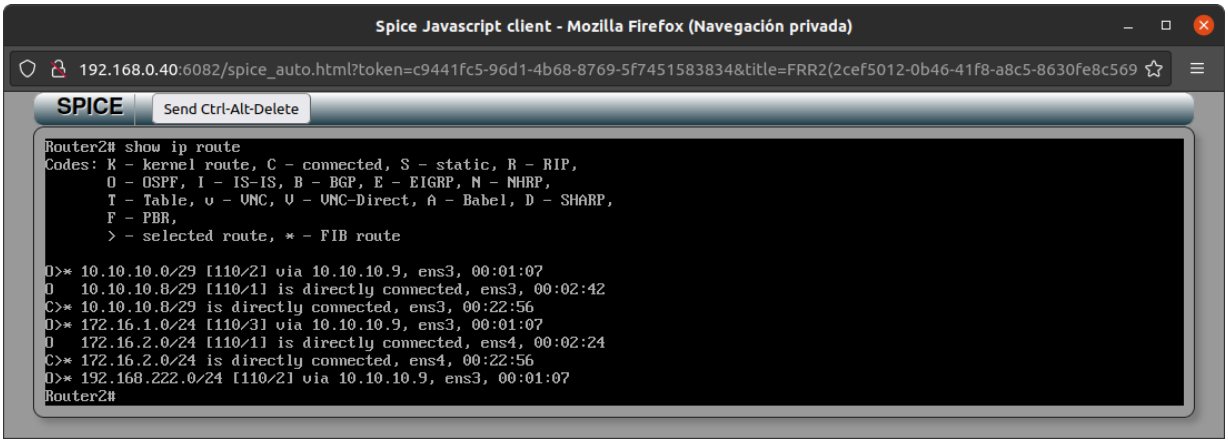


Figura 99. Rutas dinámicas de FRR2.

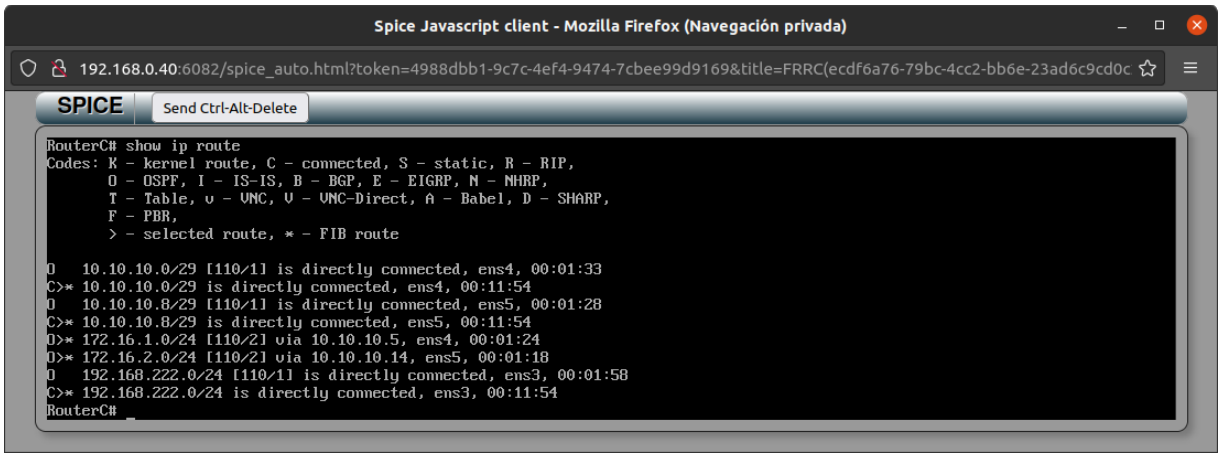


Figura 100. Rutas dinámicas de FRRC.

Con el establecimiento de las rutas, el enrutamiento dinámico de la red está completo.

CAPÍTULO V. RESULTADOS

El presente capítulo recopila un conjunto de resultados obtenidos de las actividades ejecutadas durante el desarrollo del proyecto. Se presenta como la titulación del resultado específico de cada uno de los resultados, seguida de una descripción detallada de los mismos. La estructura del capítulo engloba resultados producidos por la sinergia y compaginación de cada una de las fases y objetivos del proyecto, no son excluyentes de cada una. La organización de los resultados estará clasificada en tres grupos principales: Resultados de pruebas locales, resultados de implementación en el laboratorio y resultados de funcionabilidad.

V.1. Resultados de pruebas locales.

V.1.1. Latencia de la micro nube

La latencia entre los dispositivos de red, tanto físicos como virtuales, permite determinar el tiempo que tardan los mismos en comunicarse entre sí; es directamente proporcional a la distancia física y lógica que los separa a la cantidad de dispositivos intermediarios entre el emisor y el receptor. Este parámetro fue medido con el envío de paquetes ICMP entre el nodo de control y las instancias creadas, los resultados se muestran a continuación:

V.1.1.1. Entre instancias y el nodo de control

Al enviar paquetes ICMP desde el equipo físico controlador de la nube y una de las instancias creadas, “myinstance1”, se obtuvieron los siguientes resultados:

Paquete	Latencia (ms)	Promedio (ms)
1	2,420	0,820
2	1,710	
3	0,308	
4	0,252	
5	0,250	
6	2,450	
7	0,262	
8	0,136	
9	0,198	
10	0,212	

Tabla 6. Latencia entre el nodo de control de la “micro nube” y “myinstance1”, en milisegundos.

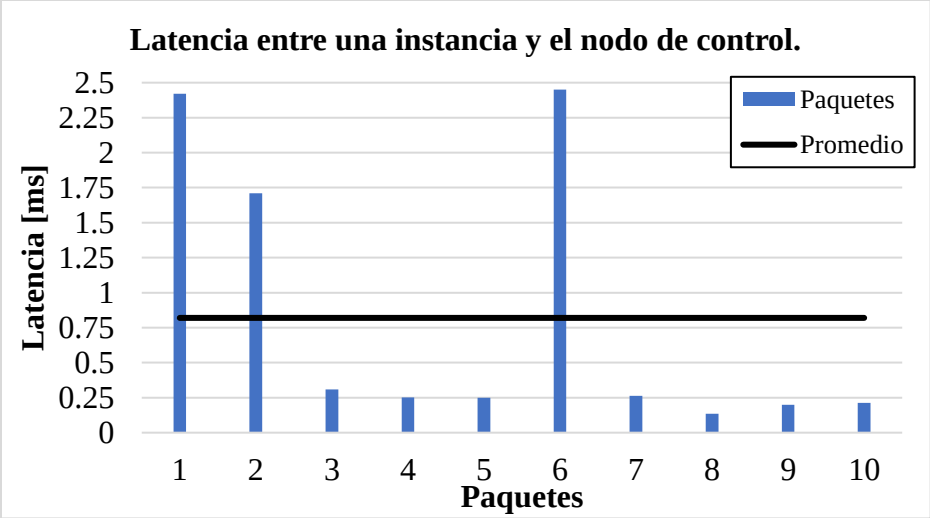


Gráfico 1. Latencia entre el nodo de control de la "micro nube" y "*myinstance1*", en milisegundos.

Nótese que, en general, la mayoría de los paquetes tardó menos de medio milisegundo en ir y volver entre ambos dispositivos, y esto se debe a que no existe un salto físico entre ellos, sino saltos lógicos relacionados a la virtualización de funciones de red de *MICROSTACK*; no obstante, paquetes 1, 2 y 6, de mayor latencia, incrementan el promedio de estos de forma considerable.

Esta mayor latencia ocurre porque el dispositivo receptor del protocolo, no conoce cuál es la dirección de control de acceso (MAC) asociada a la dirección IP del dispositivo emisor, por lo que envía un paquete a sus redes preguntando quién de los dispositivos conocidos usa dicha dirección IP, el dispositivo emisor le indica su dirección IP proviene de él con un segundo paquete de respuesta. Este intercambio de información sólo ocurre la primera vez que se ejecuta el protocolo y es la causa de los primeros paquetes más demorados con respecto al resto, los demás paquetes que demoren más que el promedio serán por cualquier causa distinta a esta.

V.1.1.2. Entre instancias

La latencia entre instancias, además de los parámetros descritos, está ligada a las limitaciones de características de *hardware* que las conforman y el tipo de funcionalidad que se simulan en sus redes. Se crearon dos instancias pertenecientes a dos redes separadas por un *router* central, al medir dos veces la latencia entre ellas, se obtuvo el siguiente resultado:

V.1.1.2.1. Desde “MyInstance1” hasta “MyInstance2”

Paquete	Latencia (ms)	Promedio (ms)
1	1,770	0,770
2	1,750	
3	0,541	
4	0,498	
5	0,531	
6	0,540	
7	0,470	
8	0,626	
9	0,504	
10	0,468	

Tabla 7. Latencia de envío de paquetes desde "*MyInstance1*" hasta "*MyInstance2*", en milisegundos.

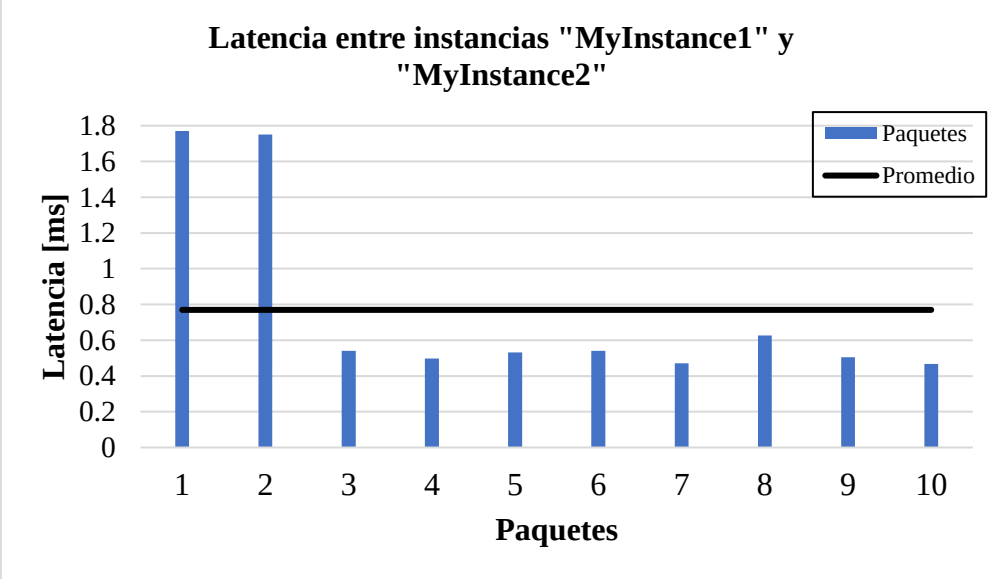


Gráfico 2. Latencia de envío de paquetes desde "MyInstance1" hasta "MyInstance2", en milisegundos.

Como puede observarse, pese al bajo tiempo de envío de los paquetes, los primeros dos paquetes tardaron un más de un milisegundo (1ms) que el resto de los mismos, esto se debe a que *myrouter* tomó ese tiempo adicional en resolver la ruta por la que tenía que redirigir el tráfico saliente desde la red de *MyInstance1*, *mynetwork1* hasta llegar a la red de destino, a la que pertenece *MyInstance2*, *mynetwork2*.

V.1.1.2.2. Desde “MyInstance2” hasta “MyInstance1”

Paquete	Latencia (ms)	Promedio (ms)
1	1,680	0,691
2	1,710	
3	0,490	
4	0,468	
5	0,436	
6	0,434	
7	0,480	
8	0,493	
9	0,164	
10	0,555	

Tabla 8. Latencia de envío de paquetes desde "*MyInstance2*" hasta "*MyInstance1*", en milisegundos.

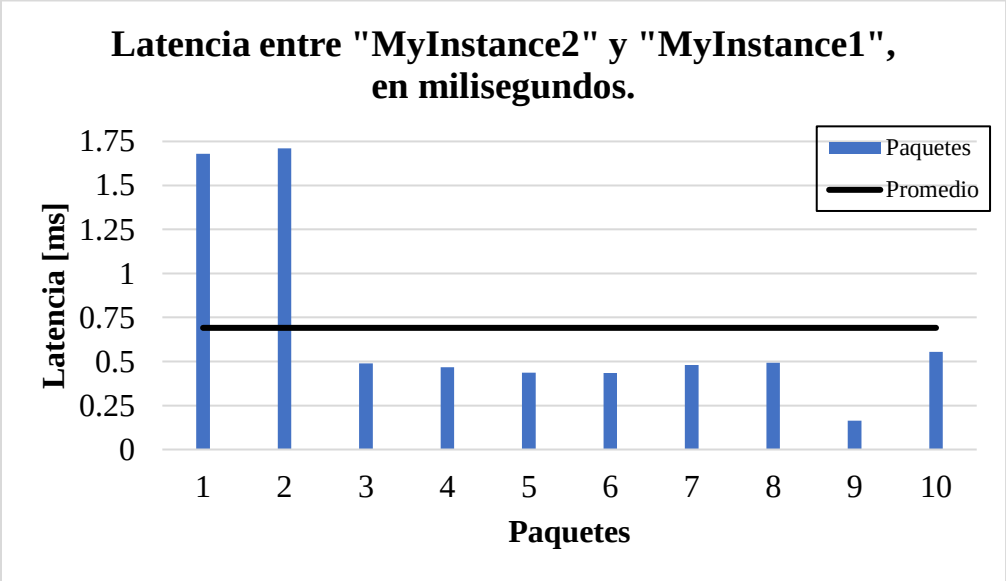


Gráfico 3. Latencia de envío de paquetes desde “*MyInstance2*” hasta “*MyInstance1*”, en milisegundos.

De igual forma en este caso, pese a que el tiempo promedio de envío de paquetes es similar a cuando se enviaron en sentido opuesto, en este caso, los dos primeros paquetes también tardan un milisegundo más que el resto de los mismos ocurre porque, de igual forma que el caso anterior *myrouter* debe resolver la ruta por la que redirige los paquetes salientes desde la red *mynetwork2* hasta *mynetwork1*.

V.1.2. Uso de puertos de cada módulo

Los módulos de *OPENSTACK* utilizan un conjunto específico de puertos TCP y UDP tanto del nodo de control como del nodo de cómputo, a los que acceden vía HTTP por medio de la web, estos puertos pueden observarse a continuación:

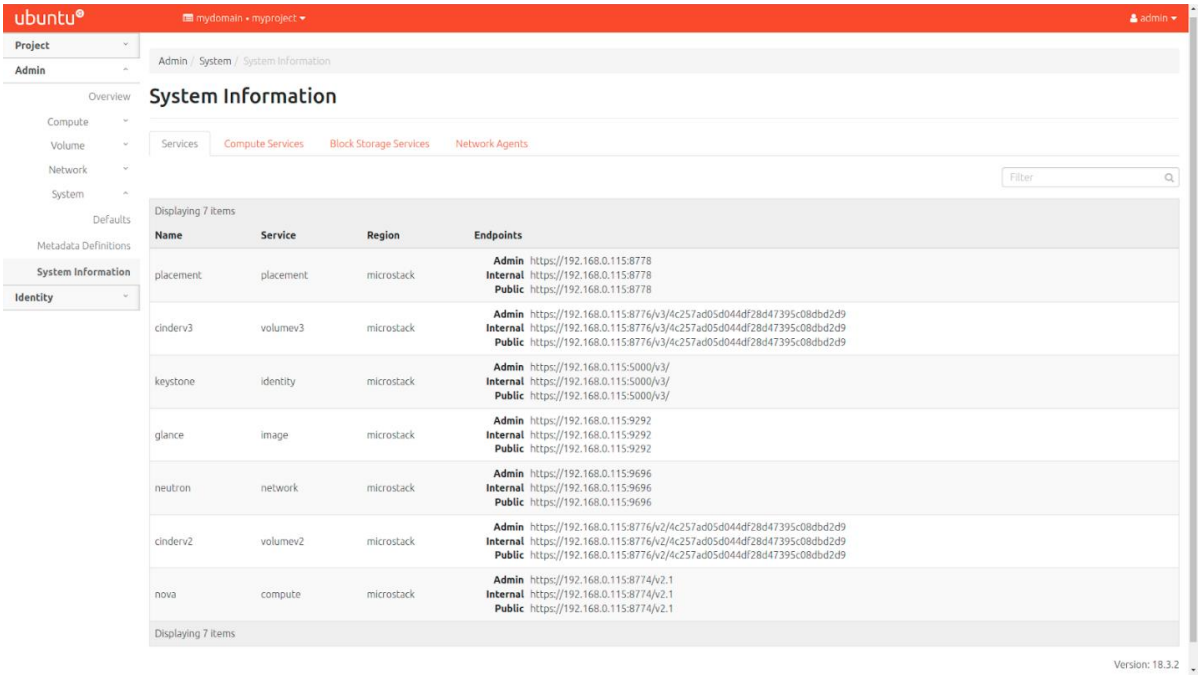


Figura 101. Uso de puertos de cada servicio de *MICROSTACK*.

V.1.3. Acceso remoto a la micro nube

En el despliegue desarrollado en el capítulo anterior, se configuraron las instancias para que fuesen accesibles por medio del protocolo SSH, no obstante, no sólo son accesibles desde los nodos que conforman la nube, sino que también se puede acceder a ellas de forma remota, realizando una segunda conexión SSH desde un dispositivo externo, perteneciente a la misma red de área local de los nodos que conforman la nube. Al realizar una segunda conexión SSH desde dicho dispositivo hasta el nodo de control, fue posible conectarse con las instancias de forma directa, utilizando la aplicación PuTTY, como se muestra a continuación (Figura 102 y Figura 103):

V.1.3.1. Conexión desde el dispositivo externo hasta el nodo de control

Utilizando la aplicación PuTTY, ingresando la dirección IP del nodo de control e indicando el protocolo SSH, se estableció conexión desde el dispositivo externo hasta el nodo de control.

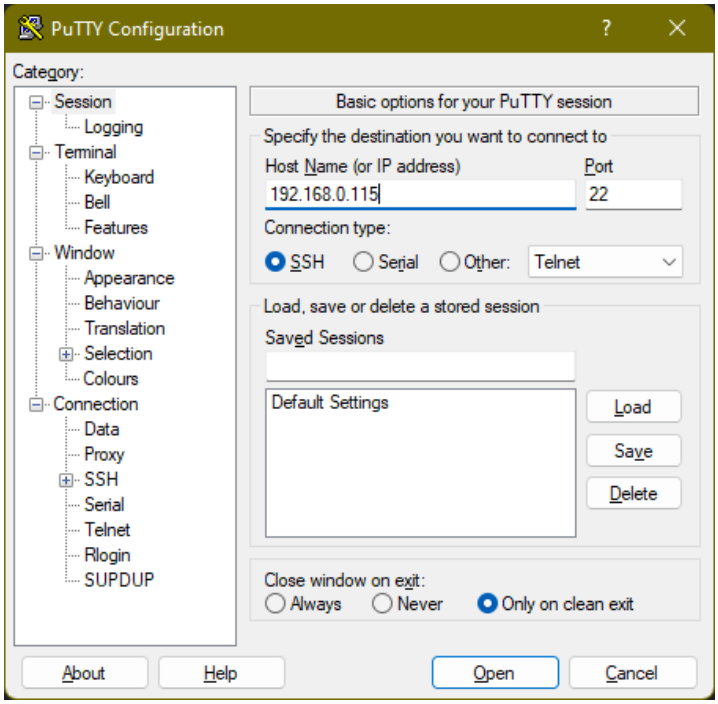


Figura 102. Parámetros de conexión SSH hacia el nodo de control, por medio de PuTTY.

Luego de abrir la conexión, la ventana solicita los datos del usuario para el inicio de sesión en el nodo de control y su correspondiente contraseña. Se ingresaron las credenciales del usuario "luis-david-casique" y se logró el acceso de forma satisfactoria, permitiendo ver parámetros tanto del equipo como de la nube desde un dispositivo externo, como se muestra a continuación:

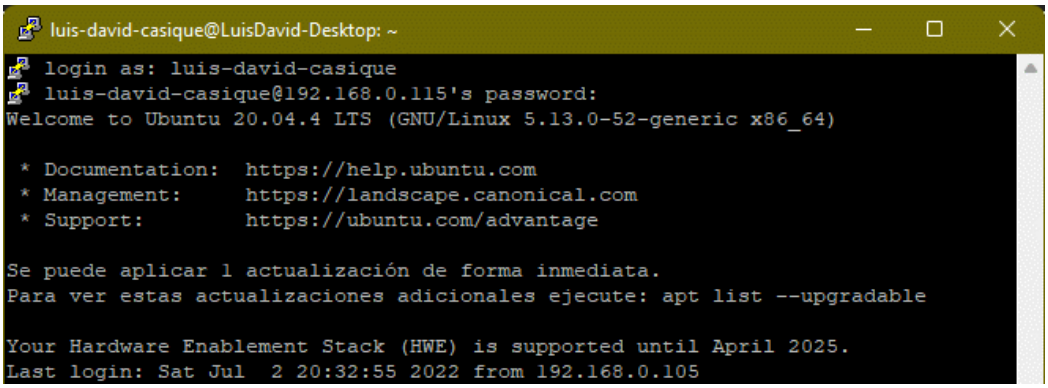


Figura 103. Acceso remoto al nodo de control por medio de la aplicación PuTTY.

V.1.3.2. Visualización de parámetros de conexión por medio de PuTTY

Una vez se inicia sesión al nodo de control por medio de PuTTY, se pueden ejecutar cualquier tipo de comandos como si se estuviese manipulando físicamente al nodo por sus propios periféricos, algunos de ellos se muestran a continuación.

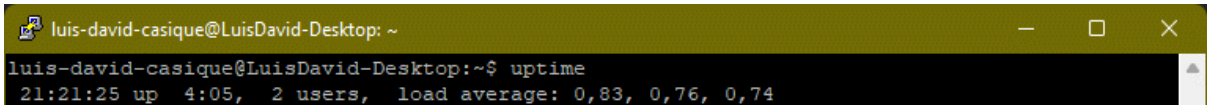


Figura 104. Visualización de hora y usuarios del nodo de control por medio de PuTTY.

Como puede observarse, el resultado anterior, indica que hay dos usuarios utilizando el equipo en el momento de tomar la captura, debido a que el nodo de control se encontraba encendido y a su vez, fue accedido de forma remota.

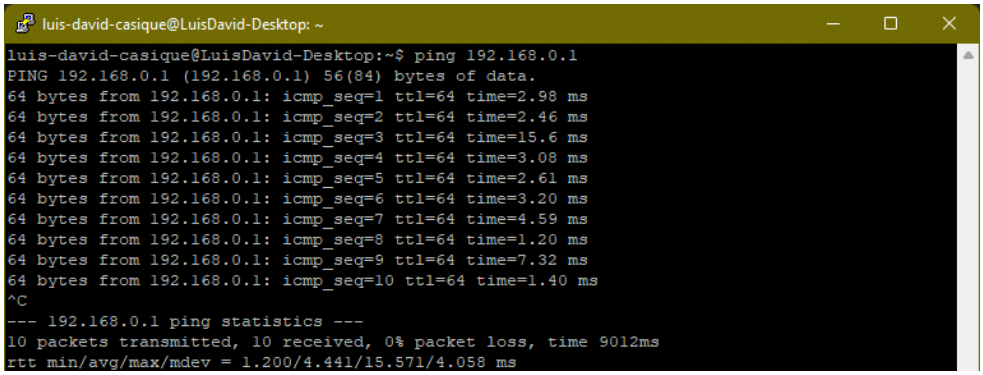


Figura 105. Comunicación exitosa con la puerta de enlace de la red de área local de los dispositivos, por medio de PuTTY.

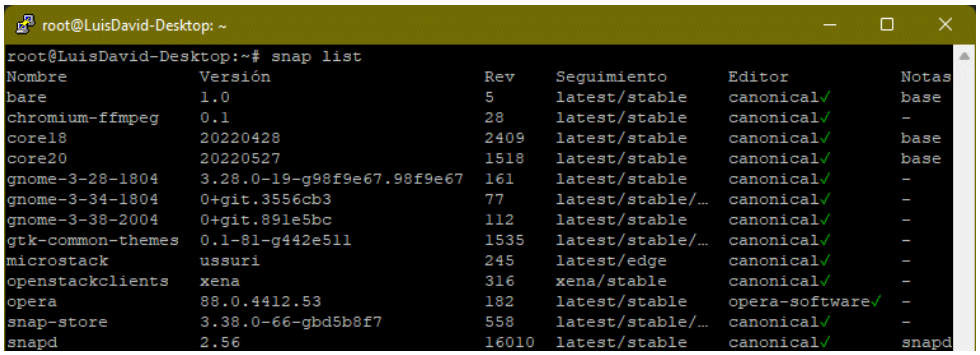


Figura 106. Visualización de paquetes *snap* del nodo de control, incluyendo *MICROSTACK*, por medio de PuTTY.

V.1.3.3. Interacción con servicios de OPENSTACK

De igual forma, como se accedió a la consola de comandos del nodo de control, se logró interactuar con los módulos de *OPENSTACK* de forma satisfactoria, como si se estuviese manipulando el nodo de control directamente, como se muestra a continuación:

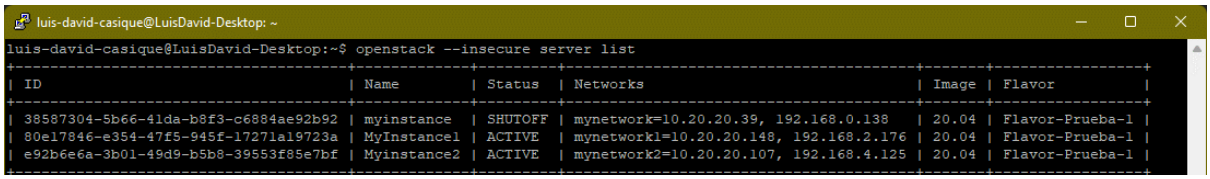


Figura 107. Instancias de la nube visibles desde PuTTY.

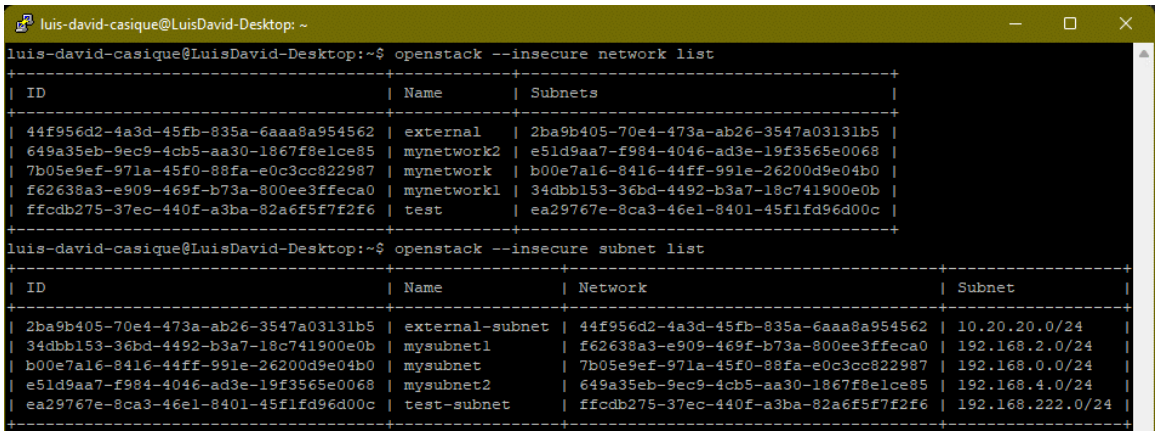


Figura 108. Redes y subredes de la nube, vistas por medio de PuTTY.

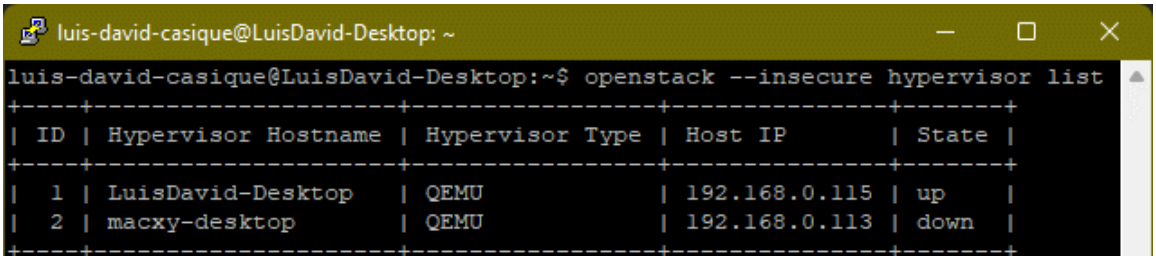


Figura 109. Hipervisores visibles, los nodos que conforman la nube, desde PuTTY.

V.1.3.4. Conexión remota a las instancias usando PuTTY

Luego de ingresar al nodo de control de forma remota, es posible interactuar directamente con las instancias de la nube, incluso, acceder a ellas con una segunda conexión SSH, como si se estuviese físicamente en el nodo de control.

En primer lugar, se mostrará la conexión e interacción con *MyInstance1*:

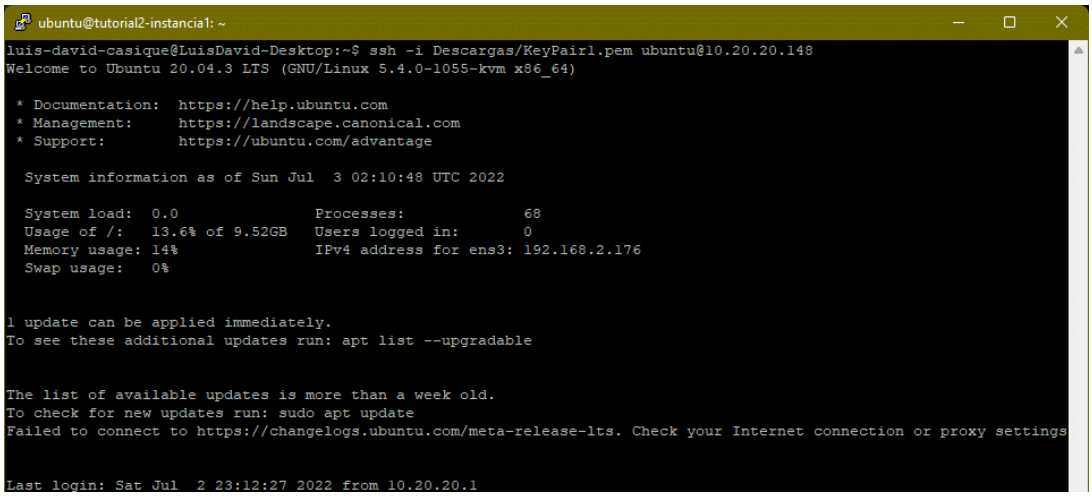


Figura 110. Acceso remoto a *MyInstance1* desde el dispositivo externo, por medio de PuTTY.

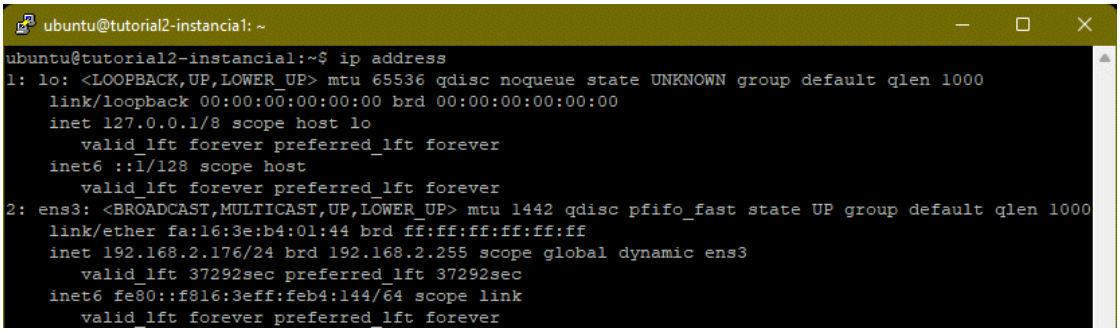


Figura 111. Interfaces de red de *MyInstance1*, vistas desde PuTTY.

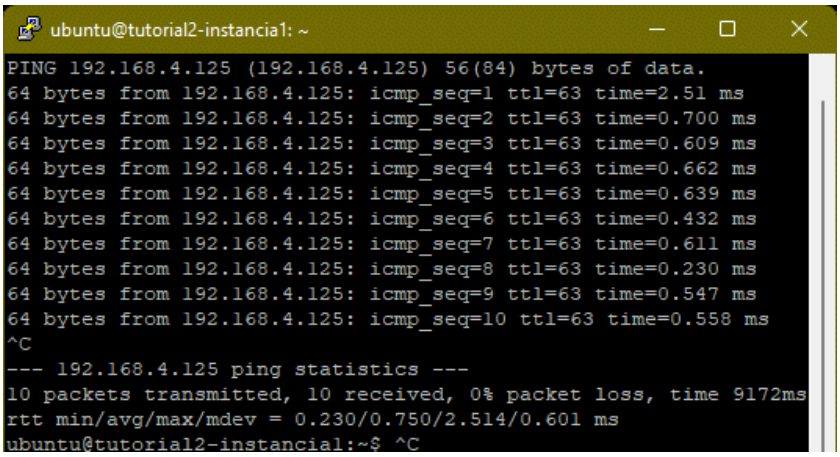


Figura 112. Comunicación entre *MyInstance1* y *MyInstance2*, solicitada desde PuTTY.

A continuación, se mostrará la conexión e interacción con *MyInstance2*:

Como puede observarse, en ambas sesiones remotas iniciadas con PuTTY, los tiempos de carga, descarga, envío y recepción de paquetes no tiene diferencias perceptibles con respecto a los tiempos de las mismas transmisiones de datos realizadas físicamente desde el nodo de control de la nube; lo cual indica que la doble conexión remota establecida para ambas sesiones no se traduce en algún retraso adicional en los tiempos de ejecución de las actividades de los dispositivos y nodos remotos, es decir, no perjudica su rendimiento en absoluto.

V.2. Resultados de implementación en el laboratorio

Desde esta sección, se plasman los resultados de la implementación de la nube en la infraestructura del laboratorio de telemática, organizándose en la formación del clúster de la nube con los respectivos nodos que la conforman, resultados de las pruebas de rendimiento y algunas actividades que ponen en práctica conocimientos impartidos en las prácticas de los laboratorios de telemática I y II.

V.2.1. Creación exitosa del clúster de la nube del laboratorio

Los nodos de cómputo, al adjuntarse correctamente a un nodo de control, conforman un clúster que combina las características de *hardware* de cada uno de ellos como un bloque de cómputo, esta propiedad se comprobó tanto en la consola de comandos como en el *dashboard* de *OPENSTACK*.

V.2.1.1. Por medio de la consola de comandos

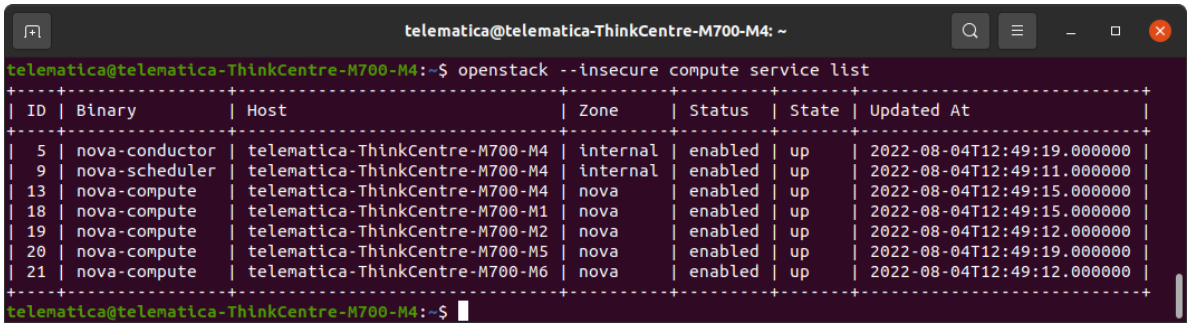


Figura 113. Servicios de cómputo en el clúster del laboratorio.

Nótese que el nodo de control, denominado “telematica-ThinkCentre-M700-M4”, muestra tres servicios de cómputo con dos “zonas de disponibilidad”. El servicio denominado “nova-conductor”, se encarga de la gestión y orquestación del *hardware* de las cinco computadoras del clúster; el servicio “*nova-scheduler*” se encarga de la programación de la creación, iniciado, detención y borrado de los elementos de cómputo virtualizados tales como las instancias o los *flavors*; mientras que el servicio “nova-compute” es la provisión en sí del propio *hardware* del nodo de control, como si fuese un nodo de cómputo más.

V.2.1.2. Por medio del dashboard

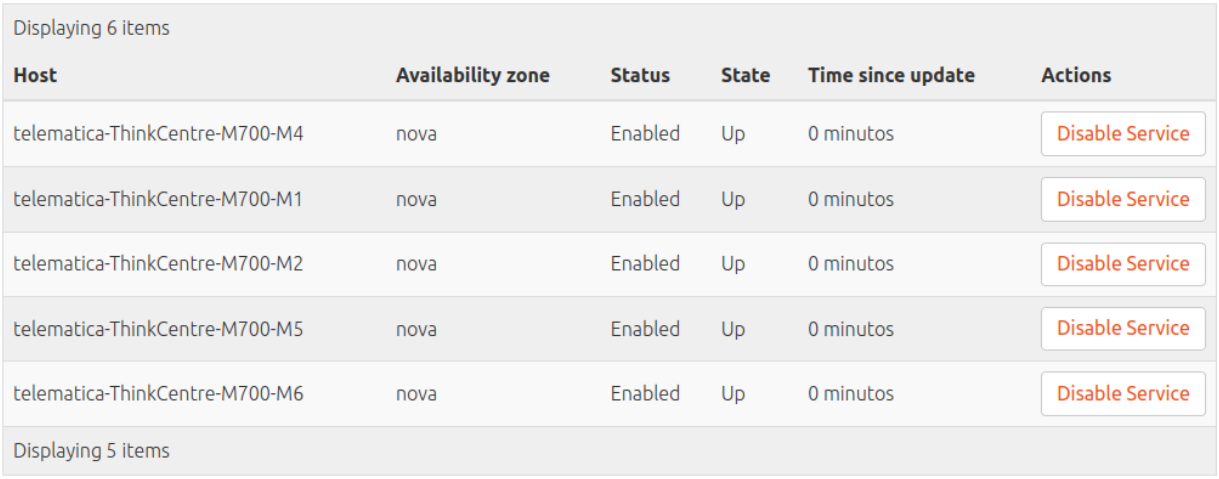


Figura 114. Servicios de cómputo de la nube vistos desde el *dashboard*.

V.2.2. Rendimiento

V.2.2.1. Tiempo de creación de las instancias.

La medición del tiempo de creación de las instancias descritas en la prueba arrojó los siguientes resultados, los cuales fueron clasificados de acuerdo al tipo de prueba en la que se recogieron los datos:

V.2.2.1.1. Tiempos de creación de instancias en el nodo de control

Flavor	**Instancia**	**Tiempo [s]**	**Promedio [s]**
TG-Flavor	1	19,30	7,802
	2	5,04	
	3	5,06	
	4	5,30	
	5	4,31	
m1.small	1	7,20	6,23
	2	6,35	
	3	4,94	
	4	7,20	
	5	5,46	

Tabla 9. Tiempo de creación de instancias en el nodo de control, en segundos.

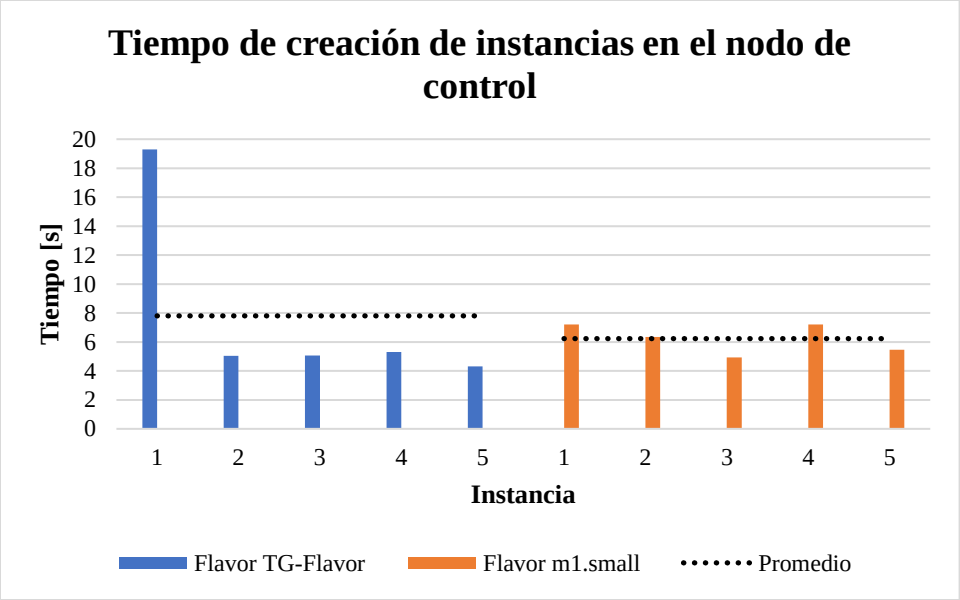


Gráfico 4. Tiempo de creación de instancias en el nodo de control, en segundos.

Como puede observarse, la primera instancia equipada con el *flavor* de menor capacidad, "TG-Flavor", fue la que más tiempo tardó en crearse, con una diferencia considerable con el resto de instancias de ambos *flavors*. Esto se debe a que fue la primera instancia creada en el día de la realización de esta prueba, con la creación de esta instancia se realizó la primera solicitud a *Nova* de utilizar el *hardware* del nodo de control de la nube, proceso que requirió un tiempo mayor en la activación y ejecución de los algoritmos y programas dentro de las memorias de los nodos que conforman la nube por primera vez en la jornada, tardando por esta causa, más tiempo que las instancias posteriores.

Es necesario resaltar que el tiempo de creación considerablemente elevado de la primera instancia de la prueba causó un incremento en el promedio asociado al *flavor* "TG-Flavor"; sin embargo, si sólo se considerasen los tiempos de creación de las instancias posteriores, la media sería menor y cercana a la las instancias asociadas con el *flavor* "m1.small".

V.2.2.1.2. Tiempo de creación de instancias en un nodo de cómputo

Flavor	**Instancia**	**Tiempo [s]**	**Promedio [s]**
TG-Flavor	1	4,64	5,42
	2	6,53	
	3	6,99	
	4	4,14	
	5	4,80	
m1.small	1	4,99	4,942
	2	6,14	
	3	4,04	
	4	4,38	
	5	5,16	

Tabla 10. Tiempo de creación de cada instancia en un nodo de cómputo, en segundos.

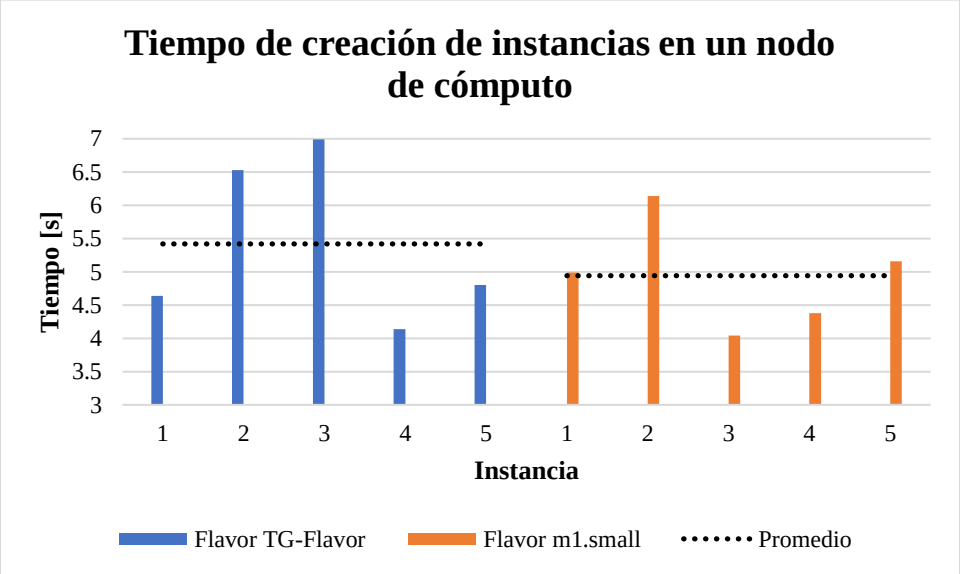


Gráfico 5. Tiempo de creación de instancias en un nodo de cómputo, en segundos.

Nótese en los tiempos de esta prueba que, la instancia que tardó más tiempo en crearse fue la tercera equipada con el *flavor* "TG-Flavor"; no obstante, sin una diferencia considerable como en la primera instancia del caso anterior. Esto se debe a que *Nova*, gestiona a los nodos de cómputo de la nube como un "todo", un solo bloque de cómputo con la suma de las memorias principales y secundarias de los mismos, por lo que la mayor carga de proceso recayó en el nodo de control, en la primera instancia que se creó en él, en la prueba anterior.

Nótese también que, en promedio, los tiempos de creación de las instancias de los dos *flavors*, en esta prueba no tienen una diferencia perceptible.

V.2.2.2. Tiempo de encendido de instancias

Encendiendo las instancias con las que desarrolló la prueba anterior, la medida de los tiempos respectivos devolvió los siguientes datos:

V.2.2.2.1. Tiempo de encendido de instancias en el nodo de control

Flavor	**Instancia**	**Tiempo [s]**	**Promedio [s]**
TG-Flavor	1	12,1	9,948
	2	9,28	
	3	10,14	
	4	9,09	
	5	9,13	
m1.small	1	9,36	9,308
	2	9,60	
	3	9,06	
	4	9,25	
	5	9,27	

Tabla 11. Tiempos de encendido de instancias en el nodo de control, en segundos.

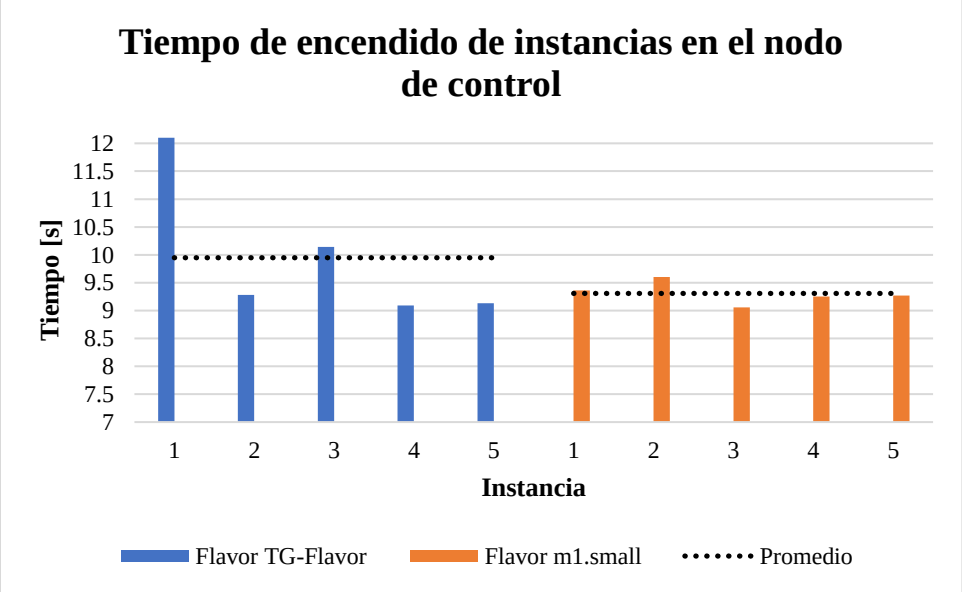


Gráfico 6. Tiempo de encendido de instancias en el nodo de control, en segundos.

A diferencia del tiempo de creación de las instancias, el tiempo de encendido es considerablemente mayor en todos los casos, esto se debe a que *MICROSTACK* emula todos los procesos que se involucran en el encendido de una computadora, independientemente de ser máquinas virtuales.

En esta prueba se observa que, al igual que en la medición del tiempo de creación, la instancia que demoró más en encender fue la primera equipada con el *flavor* menor, "TG-Flavor", no obstante, sin una diferencia tan importante como en el tiempo de creación. Esto se debe al mismo principio descrito en la prueba de tiempo de creación, al ser la primera instancia en ser encendida luego de haber creado todas las demás, realizó la primera solicitud a *Nova* de gestionar el *hardware* del nodo para encender instancias, tardando más tiempo que el resto.

V.2.2.2.2. Tiempo de encendido de instancias en un nodo de cómputo

Flavor	**Instancia**	**Tiempo [s]**	**Promedio [s]**
TG-Flavor	1	29,03	29,448
	2	30,76	
	3	29,74	
	4	31,06	
	5	26,55	
m1.small	1	26,80	30,04
	2	25,12	
	3	29,37	
	4	37,81	
	5	31,10	

Tabla 12. Tiempo de encendido de instancias en un nodo de cómputo, en segundos.

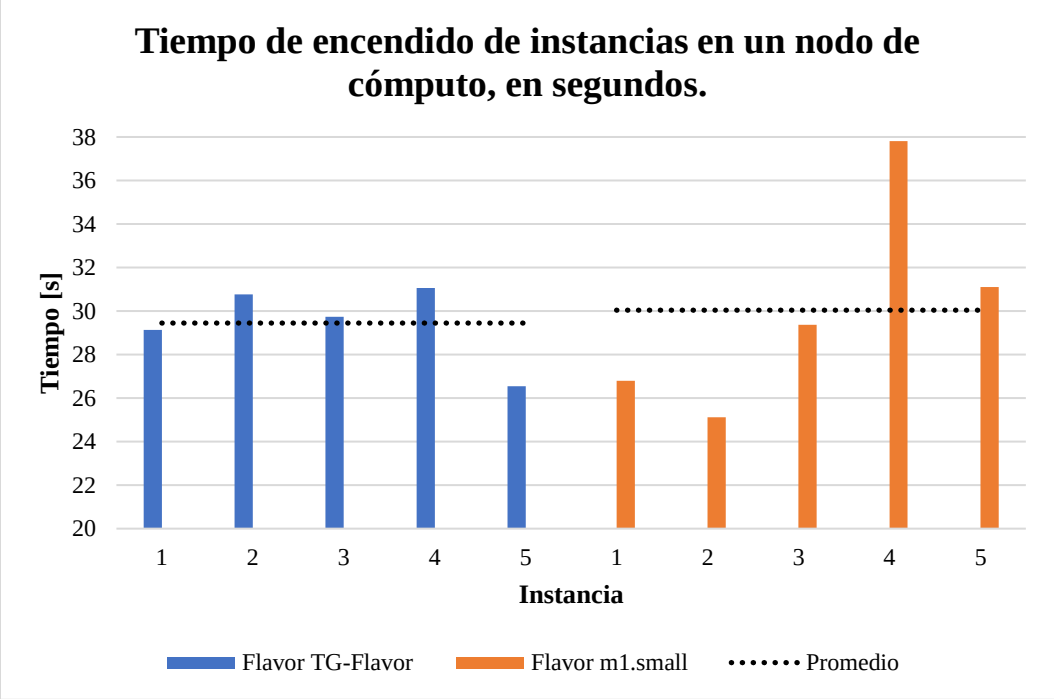


Gráfico 7. Tiempo de encendido de instancias en un nodo de cómputo, en segundos.

Nótese que, en este caso, el tiempo de encendido de las instancias creadas en los nodos de cómputo es considerablemente mayor que cuando están creadas en el nodo de control, tardando en promedio aproximadamente el triple de tiempo que tardaban las instancias creadas en el mismo.

Esto se debe a que, pese a cada nodo de cómputo funciona como un hipervisor, *Nova* gestiona algunos servicios de control de los nodos de cómputo, desde el nodo de control; y al estar este separado físicamente de los nodos de cómputo, hace uso de los cables *Ethernet* con los que están conectados los nodos de la nube para comunicarse; y este medio físico tiene una limitación de velocidad que es masivamente menor a la velocidad de trabajo de las memorias físicas de una computadora. Esta limitación, en el caso del encendido de las instancias, se traduce en un retardo adicional en el tiempo de encendido, que requiere de la ignición y corrido de múltiples procesos iniciales en la computadora que se ven ralentizados por la capacidad del medio físico que interconecta a los nodos de la nube.

Cabe destacar que, el promedio de encendido de las instancias con los dos *flavors* es prácticamente igual, cuando la diferencia entre las configuraciones de los mismos, en teoría, debería traducirse en un menor tiempo de encendido de las instancias que equipan el *flavor* "m1.small", al tener memorias de mayor capacidad.

V.2.2.3. Comparación con el entorno de VirtualBox

Luego de medir los tiempos descritos en *MICROSTACK*, se replicaron en la aplicación *VirtualBox*, donde se obtuvieron los siguientes resultados:

Instancia	**Creación**	**Promedio**	**Encendido**	**Promedio**	***Flavor***
1	69.94	79.13	12	11.008	*TG-Flavor*
2	100.82		13.26		
3	81.93		10.6		
4	66.56		9.53		
5	76.39		9.65		
6	158.36	170.71	8.98	9.072	*m1.small*
7	168.61		9.78		
8	195.31		8.5		
9	160.42		9.37		
10	170.85		8.73		

Tabla 13. Tiempos de creación y encendido de instancias en VirtualBox, en segundos.

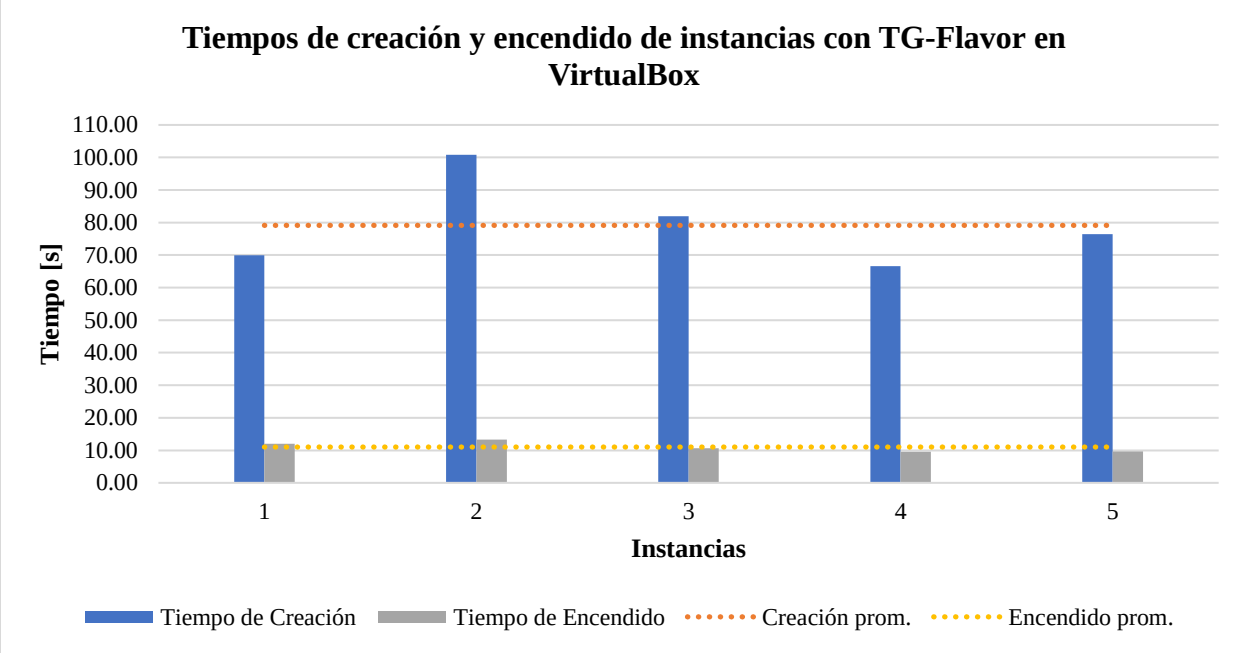


Gráfico 8. Tiempos de creación y encendido de instancias con *TG-Flavor* en *VirtualBox*, en segundos.

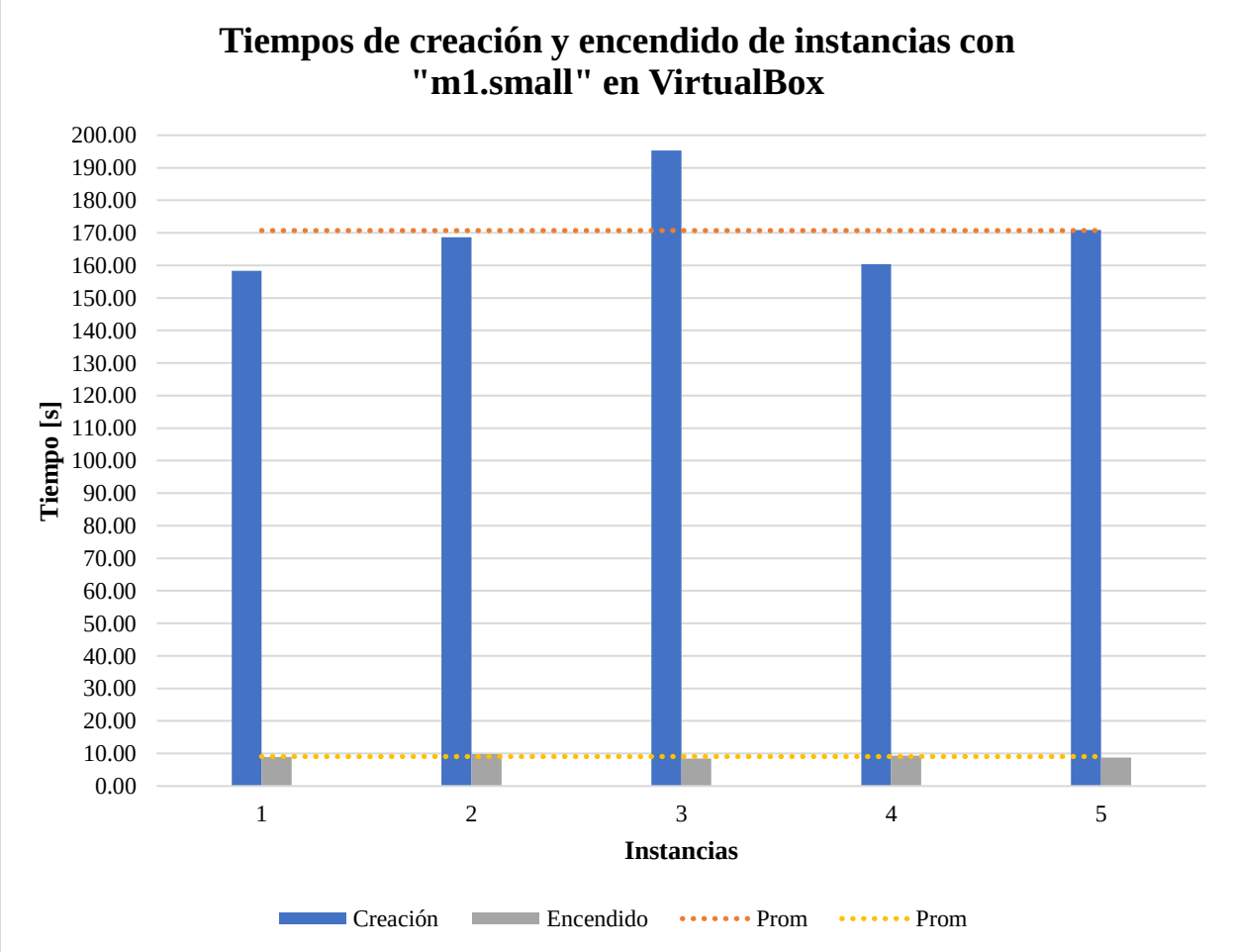


Gráfico 9. Tiempos de creación y encendido de instancias con *flavor* "*m1.small*" en *VirtualBox*, en segundos.

Como puede observarse, los tiempos de creación de las máquinas virtuales en este entorno son considerablemente mayores que los de las instancias de la nube. Esto ocurre principalmente por las diferencias en los procesos de creación de las mismas en ambos entornos; por un lado, en *MICROSTACK*, es posible crear instancias en tiempos reducidos porque permite la asignación de recursos de cómputo con los perfiles de configuración predefinidos, los *flavors*, utilizando un conjunto de características de *hardware* de los nodos físicos que ya ha sido preparado por *Nova* para su utilización en la nube; mientras que, en la plataforma de *VirtualBox*, es necesario crear de forma manual las particiones de disco duro y memoria principal para cada una de las instancias, lo cual ocupa una mayor cantidad de tiempo en la virtualización de los discos físicos para cada instancia, lo cual es una diferencia considerable a favor de *MICROSTACK*.

Por otra parte, los tiempos de encendido de las instancias son más acordes y similares a los registrados con *MICROSTACK* en el nodo de control, dado que las máquinas de *VirtualBox* fueron creadas físicamente usando un mismo nodo, incluso son inversamente proporcionales a las características de memorias asignadas a las mismas, lo cual es consecuente con que el uso de memorias de mayor capacidad, reduce los tiempos de carga de las computadoras, entre ellos, el encendido.

V.2.3. Latencia

V.2.3.1. Latencia inter-nodo

Por medio del envío y recepción de paquetes ICMP desde el nodo de control hasta cada nodo de cómputo, la latencia general de la nube arrojó los siguientes resultados:

Máquina	Paquete	Latencia [ms]	Promedio [ms]
1	1	0,686	0,676
	2	0,680	
	3	0,641	
	4	0,697	
2	1	0,677	0,662
	2	0,633	
	3	0,585	
	4	0,754	
3	1	0,466	0,412
	2	0,400	

Máquina	Paquete	Latencia [ms]	Promedio [ms]
3	3	0,389	0,412
	4	0,393	
4	1	0,423	0,456
	2	0,476	
	3	0,416	
	4	0,509	

Tabla 14. Latencia entre los nodos de la nube, en milisegundos.

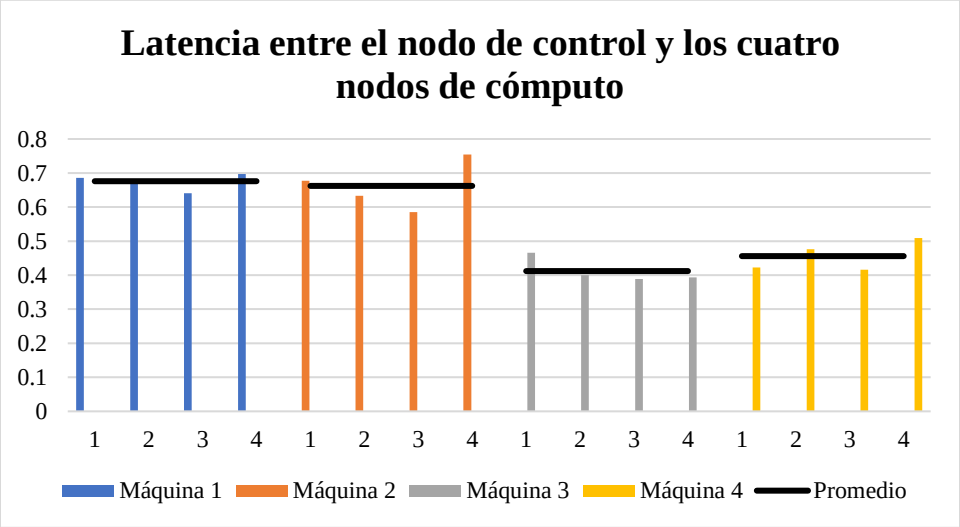


Gráfico 10. Latencia entre los nodos de la nube.

La importancia de medir esta latencia, a pesar de ser perceptiblemente baja, radica en que las instancias creadas en nodos distintos (ver secciones V.2.3.2.3 y V.2.3.2.4) mostraron una latencia entre sí mayor que aquellas que fueron creadas en el mismo nodo (secciones V.2.3.2.1 y V.2.3.2.2), siendo la diferencia entre ambas latencias aproximadamente igual al promedio de latencias inter nodo medida. Los casos descritos serán mostrados a continuación:

V.2.3.2. Latencia entre las instancias con respecto a sus respectivos nodos

Luego de crear instancias en el nodo de control y en distintos nodos de cómputo, se midió la latencia existente en la comunicación entre unas con otras, de acuerdo a cuatro situaciones distintas, arrojando los siguientes resultados:

V.2.3.2.1. Entre instancias creadas en el nodo de control

Paquete	Latencia [ms]	Promedio [ms]
1	1,282	0,874
2	0,716	
3	0,738	
4	0,758	

Tabla 15. Latencia entre instancias creadas en el nodo de control, en milisegundos.

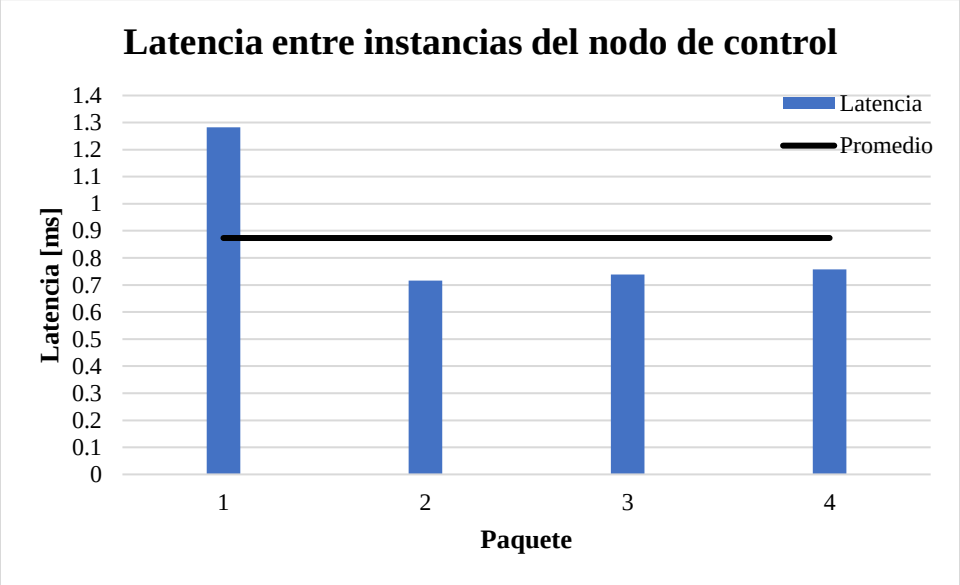


Gráfico 11. Latencia entre instancias creadas en el nodo de control, en milisegundos.

Nótese que, en general la latencia es baja, similar a la latencia general, únicamente algunas décimas de milisegundo mayor consecuentes a la virtualización de las interfaces de red. La latencia es baja porque, al estar embebidas en el mismo nodo de control, los paquetes sólo realizan un salto hacia la instancia de destino. Cabe destacar que en este caso se repite la mayor latencia del primer paquete.

V.2.3.2.2. Entre instancias creadas en un mismo nodo de cómputo

Paquete	Latencia [ms]	Promedio [ms]
1	2,795	1,311
2	0,926	
3	0,760	
4	0,763	

Tabla 16. Latencia entre instancias creadas en el mismo nodo de cómputo, en milisegundos.

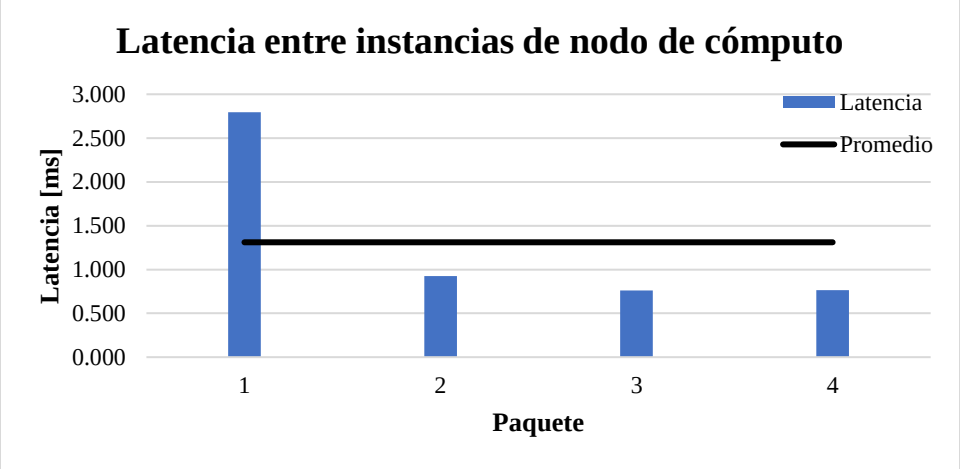


Gráfico 12. Latencia entre instancias creadas en el mismo nodo de cómputo.

A diferencia del primer paquete, tardando poco menos de tres milisegundos, en general la latencia fue similar al caso de las instancias creadas en el nodo de control, debido a que se cumple el mismo criterio, sólo existe un salto entre los paquetes que recorren las instancias, y el primer paquete tarda más en ir y volver.

V.2.3.2.3. Latencia entre instancias del nodo de control y de un nodo de cómputo

Paquete	Latencia [ms]	Promedio [ms]
1	3,240	1,805
2	1,673	
3	1,051	
4	1,257	

Tabla 17. Latencia entre instancias del nodo de control y de un nodo de cómputo.

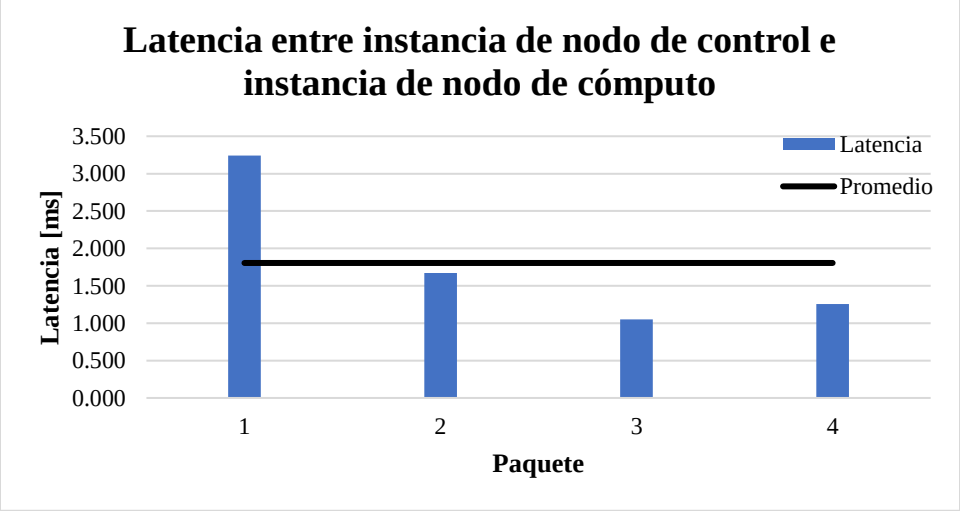


Gráfico 13. Latencia entre instancias del nodo de control y del nodo de cómputo.

Como puede observarse, las latencias crecen con respecto a los dos casos anteriores debido a que, al estar las dos instancias en cuestión embebidas en el nodo de control y en un nodo cómputo, los paquetes ejecutan dos saltos en su recorrido de una instancia a otra, añadiendo una latencia adicional similar a los promedios de las anteriores pruebas. La mayor duración del primer paquete también es reflejada en los datos.

V.2.3.2.4. Latencia entre instancias de dos nodos de cómputo distintos

Paquete	Latencia [ms]	Promedio [ms]
1	1,160	1,274
2	1,307	
3	1,331	
4	1,297	

Tabla 18. Latencia entre instancias de dos nodos de cómputo distintos.

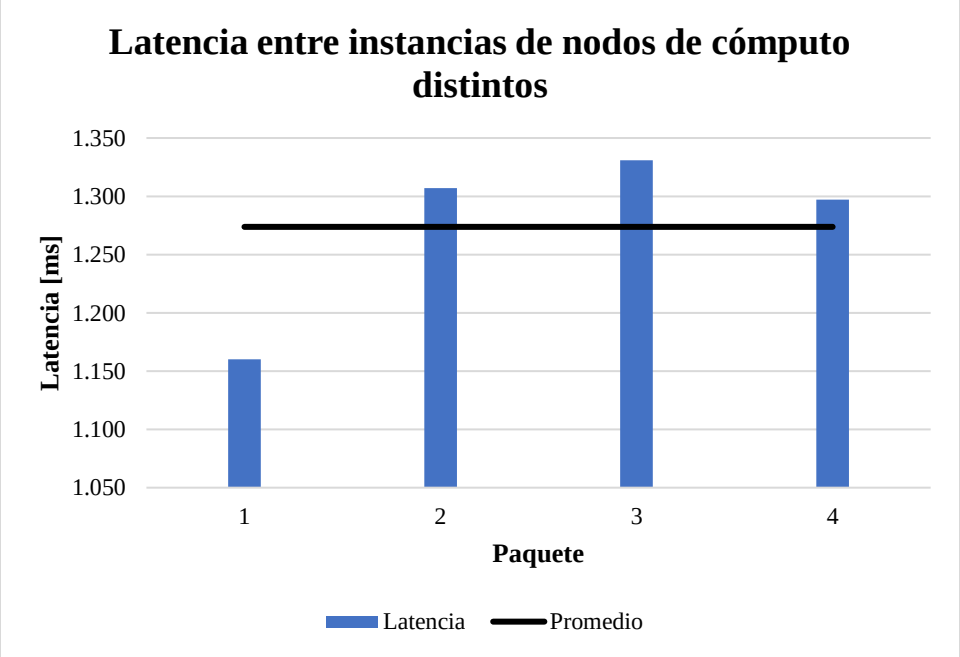


Gráfico 14. Latencia entre instancias de dos nodos de cómputo distintos.

En este caso, las latencias son similares al del caso anterior, debido a que los paquetes igualmente realizan dos saltos en su recorrido de una instancia hacia la otra. Nótese también que el primer paquete fue el de menor latencia, esto se debe a que, al momento de realizar esta última medición, ya se habían enviado y recibido algunos paquetes entre las instancias involucradas, por lo que el primer paquete reflejado en esta medición tardó en realizar su recorrido un tiempo similar al promedio, llamativamente en este caso, siendo el menor de todos.

V.2.4. Acceso remoto a la nube

De igual forma que en la "micro nube", se accedió a la infraestructura de la nube de forma remota por medio de la aplicación de SSH, PuTTY, desde una computadora adicional, conectada a la misma red de la nube. Igualmente, mediante el acceso SSH al nodo de control, como se muestra a continuación (ver Figura 115, Figura 116, Figura 117, Figura 118 y Figura 119):

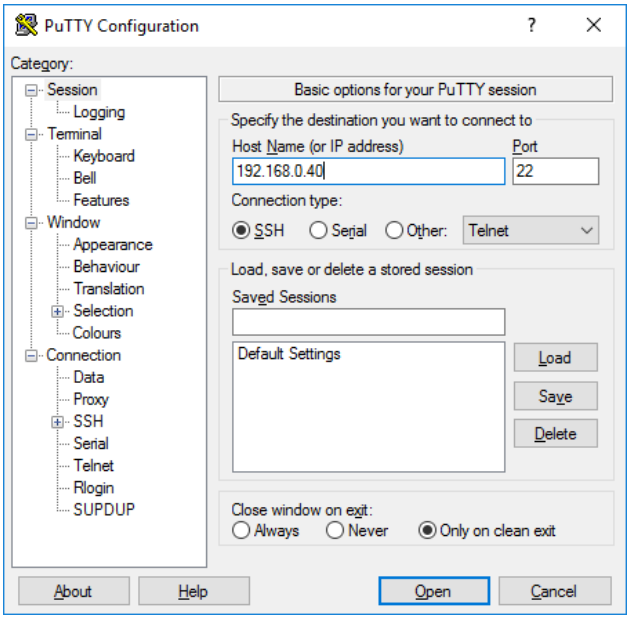


Figura 115. Ingreso de dirección IP del nodo de control desde PuTTY.

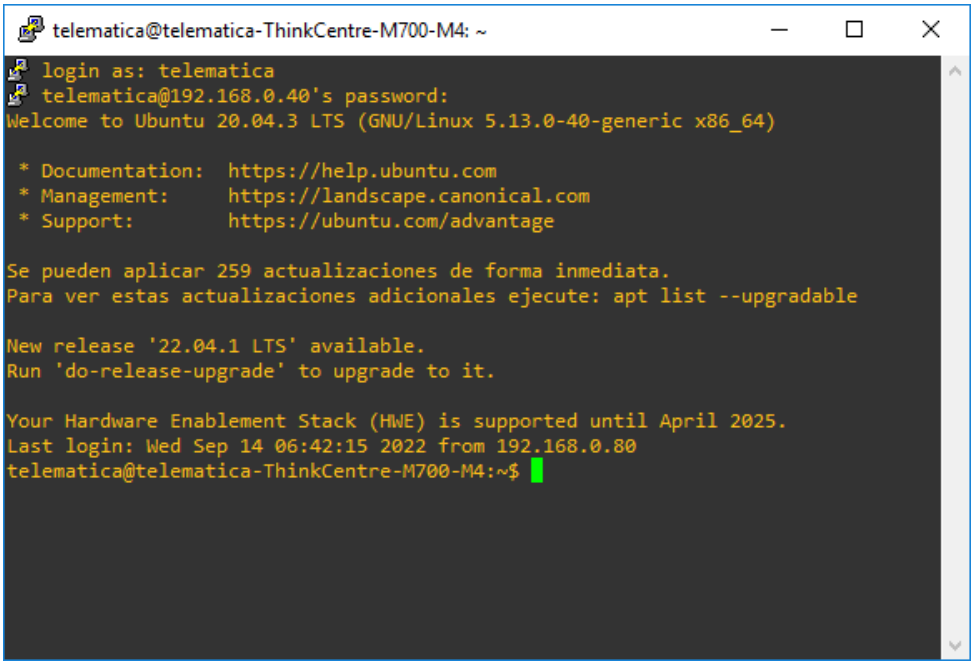


Figura 116. Acceso SSH al nodo de control del laboratorio.

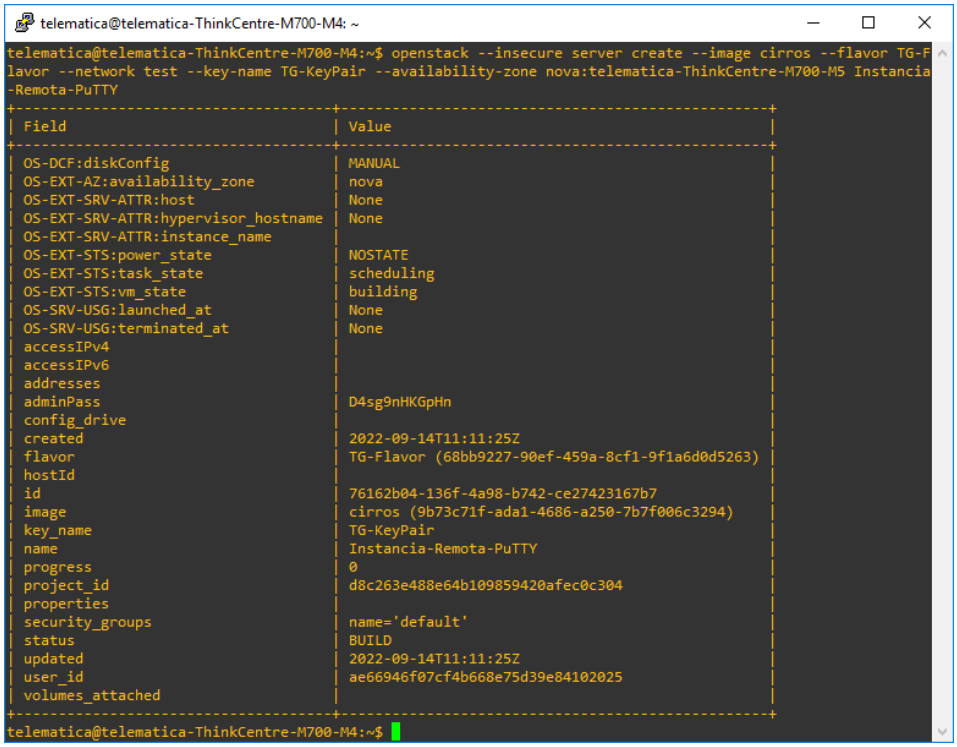


Figura 117. Creación remota de instancia desde PuTTY.

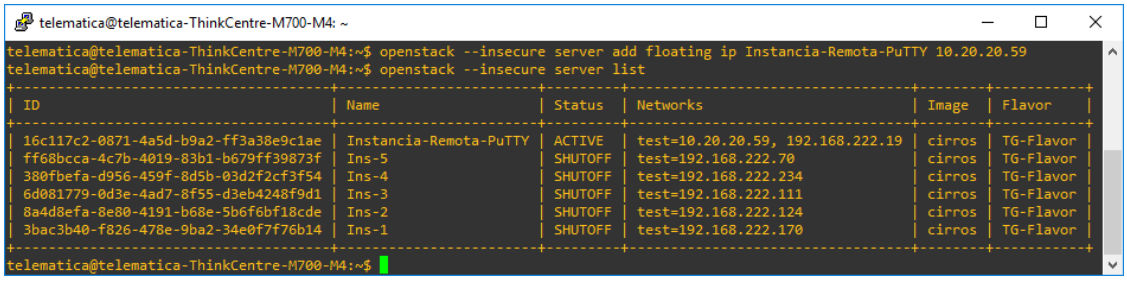


Figura 118. Asignación de IP flotante y visualización de la instancia creada.

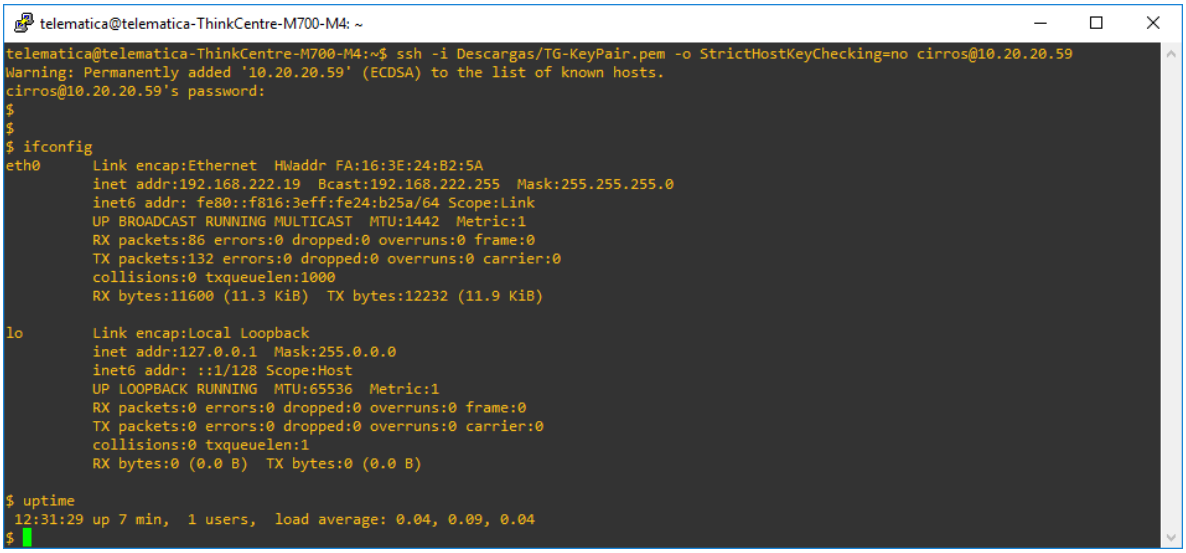


Figura 119. Acceso remoto a la instancia creada desde PuTTY.

Como puede observarse, de forma remota, se logró acceder a la totalidad de las funcionalidades del control de la infraestructura de la nube por medio del *OPENSTACK Client*, y a su vez, acceder de forma satisfactoria a las instancias creadas en la nube por medio de las llaves SSH y direcciones IP flotantes, por lo que se ha cumplido una de las premisas del TG de forma satisfactoria.

V.3. Actividades de las cátedras de laboratorio que pueden sacar provecho a la nube

El diseño y operatividad de las redes informáticas de computadoras es el principal objeto de estudio en las cátedras de laboratorio del área de telemática en la institución [41]. Este estudio se realiza mediante la utilización de la infraestructura real del laboratorio y sus equipos de red [1], y adicionalmente apoyándose en un conjunto de herramientas de virtualización de máquinas [2] y su monitoreo con aplicaciones de *software* dedicadas para este fin [3].

V.3.1. Diseño de redes y subredes basadas en IP

El estudio físico y electrónico del diseño de redes, es llevado a cabo en el laboratorio utilizando su infraestructura física y equipos de red, no obstante, para el estudio del diseño lógico de las redes, su dimensionamiento y direccionamiento IP, se emplean aplicaciones de diseño y dimensionamiento de redes tales como CISCO *Packet Tracer* o GNS3, contando las mismas con interfaces gráficas de usuario que optimizan su usabilidad para el aprendizaje práctico.

La plataforma de *MICROSTACK/OPENSTACK* permite el diseño de redes de varias topologías y dimensiones de acuerdo con el tipo de despliegue que el usuario desee implementar. Se podría hacer uso de la herramienta para la creación de redes de distintas dimensiones con su respectivo direccionamiento IP, utilizando instancias y enrutadores como principales dispositivos virtualizados.

Con el fin de aprender sobre el diseño lógico y topológico, por medio del *dashboard* de *OPENSTACK*, el estudiante puede entenderse y familiarizarse con la virtualización de funciones de red y el diseño de las mismas de forma interactiva y didáctica, de forma similar a los simuladores como *Packet Tracer* y GNS3, con la diferencia que se desarrollaría en una plataforma

basada en la nube, detalle que sirve de apertura al estudiante al tópico del *cloud computing* y los elementos que se involucran.

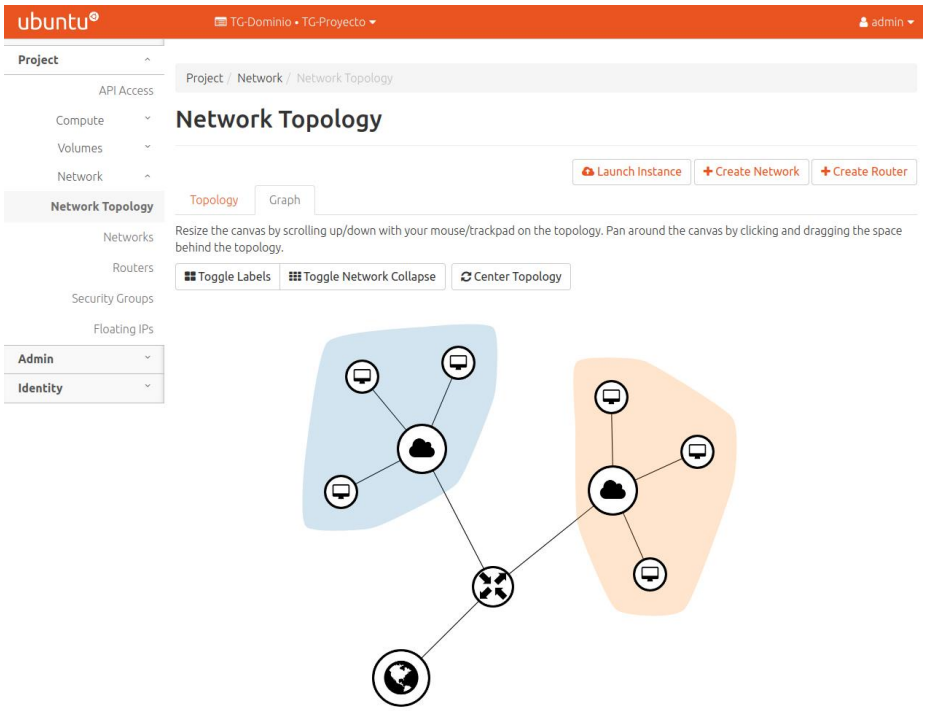


Figura 120. Esquematización una topología vista desde la interfaz gráfica de usuario de *OPENSTACK*.

V.3.2. Enrutamiento estático

La prueba de establecimiento de rutas de red estáticas realizada en el capítulo anterior, permitió que todas las instancias de la red se comuniquen entre sí, como se muestra a continuación:

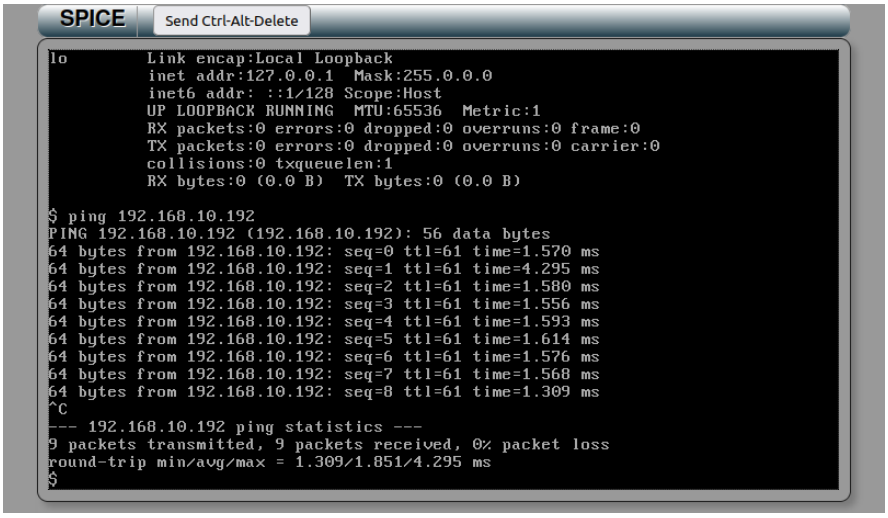


Figura 121. Comunicación entre “TG-Instance-#1” y “TG-Instance-#16”, pertenecientes a las redes “TG-Network-#1” y TG-Network-#6” respectivamente.

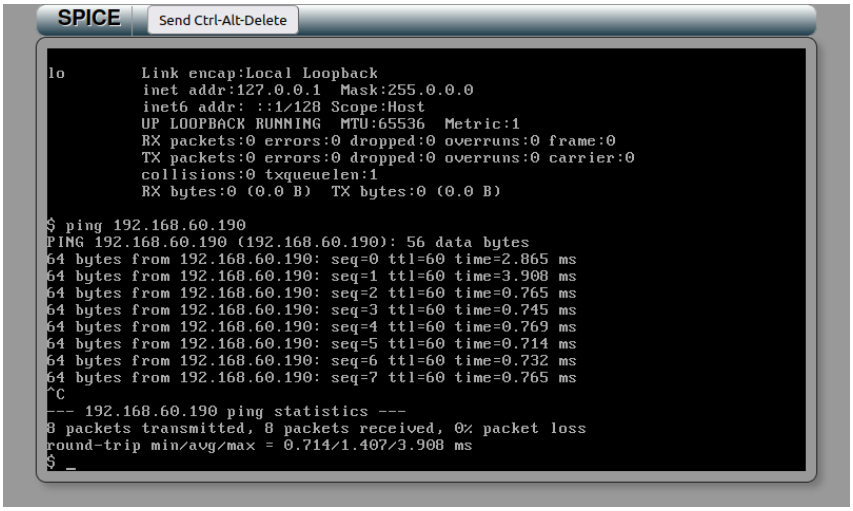


Figura 122. Comunicación entre “TG-Instance-#17” y “TG-Instance-#24”, pertenecientes a las redes “TG-Network-#6” y “TG-Network-#8” respectivamente.

Las instancias de las figuras anteriores, están separadas lógicamente por los cuatro enrutadores de la topología, por lo que los paquetes debieron recorrer y realizar los saltos establecidos de forma estática en cada uno de ellos, que, al ser recibidos los paquetes ICMP, el enrutamiento estático se concluye como exitoso.

V.3.3. Enrutamiento dinámico

La prueba de enrutamiento dinámico, realizada primeramente con la aplicación *Quagga* y luego con *FRRouting*, arrojó los siguientes resultados:

V.3.3.1. Establecimiento de rutas

Una forma alternativa de mostrar exclusivamente las rutas dinámicas asociadas a OSPF es empleando el comando `# show ip ospf route`, en la consola virtual VTYSH, en las tres instancias enrutadoras, como se muestra a continuación:

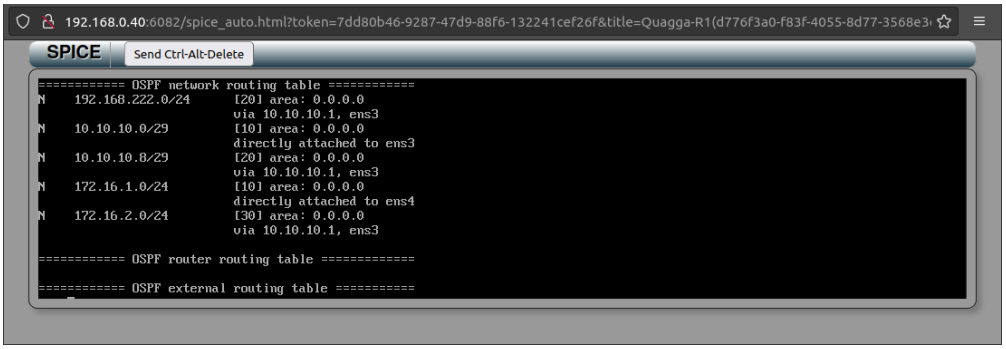


Figura 123. Rutas exclusivamente dinámicas de FRR1.

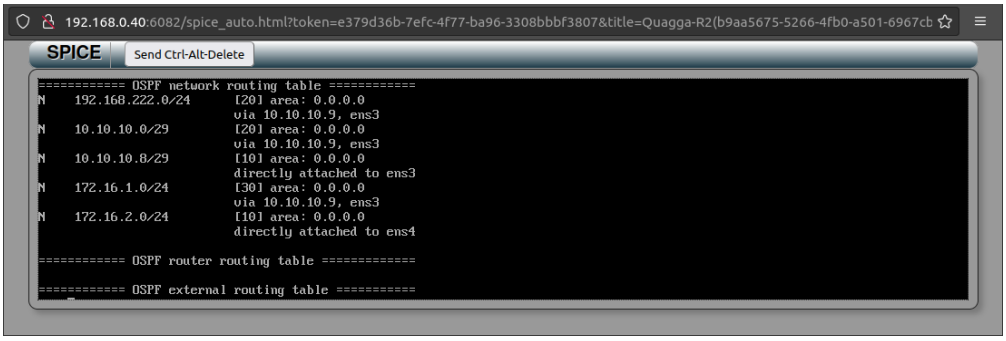


Figura 124. Rutas exclusivamente dinámicas de FRR2.

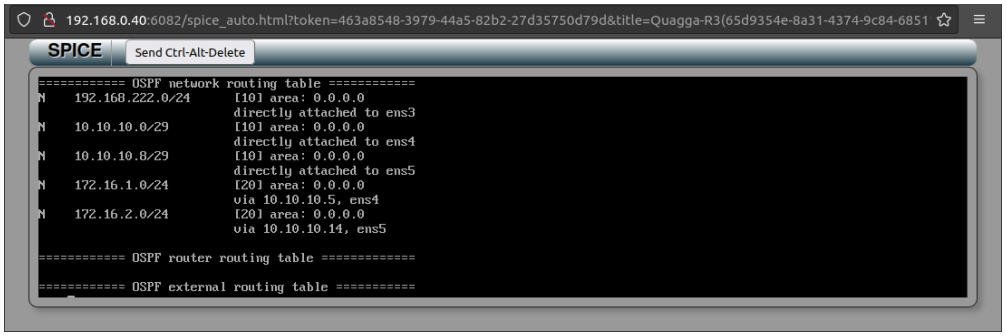


Figura 125. Rutas exclusivamente dinámicas de FRRC.

La visualización de estas rutas indica la convergencia completa de la red y el conocimiento por cada instancia enrutadora de cada ruta por la que deben redirigir los paquetes entrantes y salientes para llegar a cada una de las redes que conforman la topología, lo que quiere decir que el protocolo de enrutamiento dinámico sirvió de forma satisfactoria.

V.3.3.2. Comunicación ICMP

No obstante, la comunicación ICMP entre las instancias enrutadoras, arrojó los siguientes resultados:

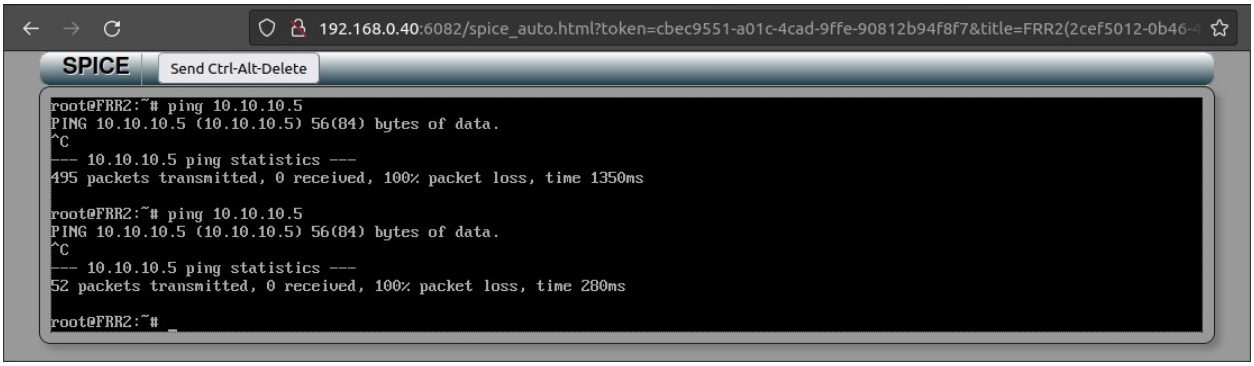


Figura 126. Comunicación entre FRR2 y FRR1 reiterativa fallida.

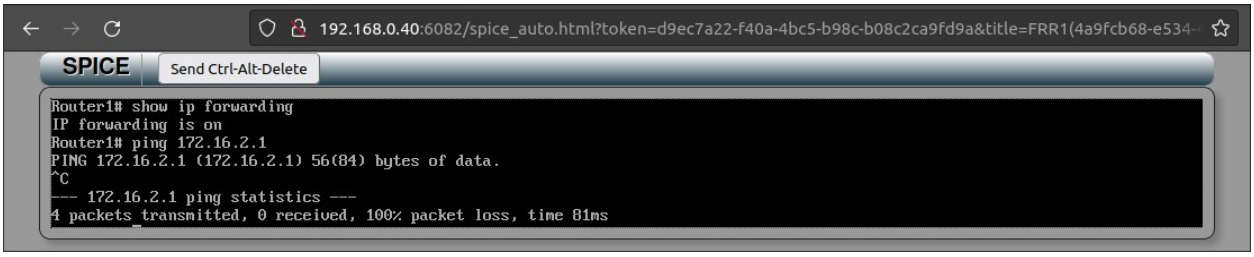


Figura 127. IP *forwarding* activo y comunicación fallida entre FRR1 y FRR2, desde VTYSH.

Como puede observarse, pese a estar establecidas, las instancias no son capaces de intercambiar paquetes cuando deben saltar al menos una instancia enrutadora, a diferencia de las comunicaciones directas punto a punto, como se muestra a continuación:

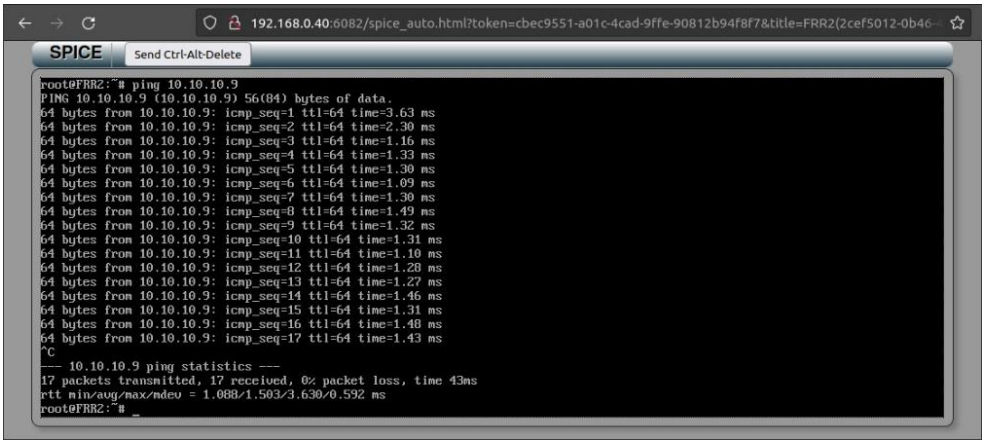


Figura 128. Comunicación exitosa entre FRR2 y FRRC, adyacentes.

Para analizar a profundidad esta comunicación fallida, se procedió a ejecutar un protocolo de captura de paquetes dentro de la red, por medio del capturador "*tcpdump*". Se escogió analizar los paquetes en la instancia FRRC, debido a que es un elemento central de toda la topología, por el que deben pasar la mayoría de los paquetes para llegar desde un extremo de la red al otro, se escogió la interfaz "ens5", directamente conectada a FRR2. Se analizó en dos casos: cuando la solicitud ICMP tiene como destino la interfaz descrita, y cuando tiene destino a FRR1:

V.3.3.2.1. Con solicitud ICMP con destino a ens5 de FRRC

Previo a esta medición, se ejecutó un envío de paquetes desde FRR2. Con el comando `tcpdump -i ens5`, se obtuvieron los siguientes resultados:

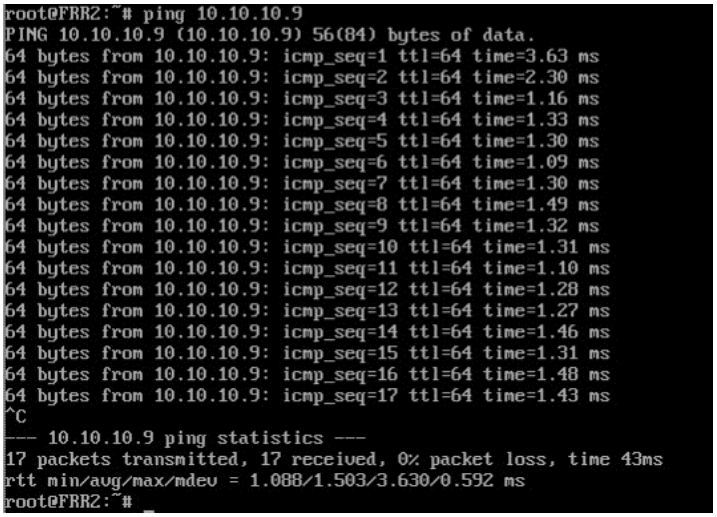


Figura 129. Comunicación directa desde FRR2 hasta ens5 de FRRC.

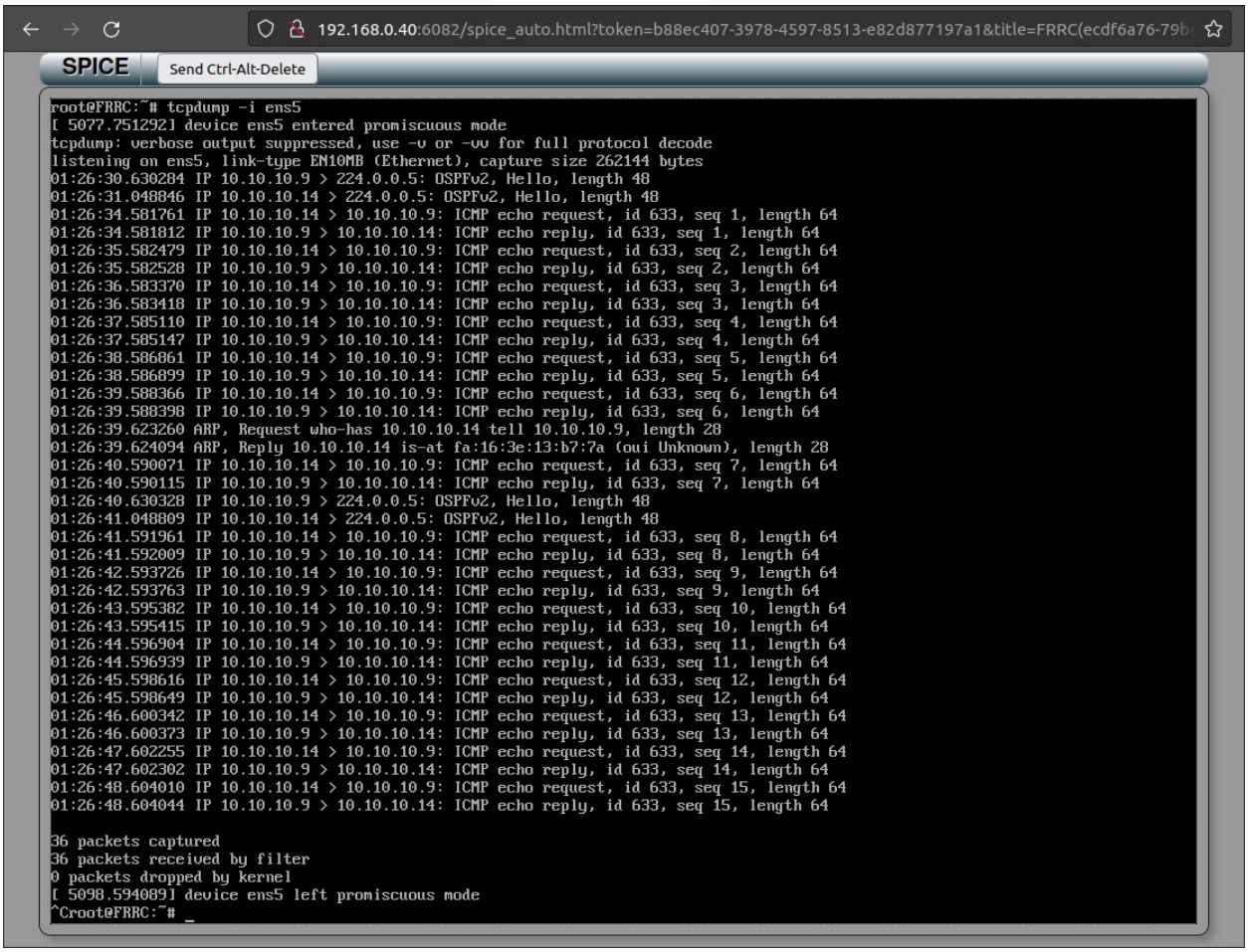


Figura 130. Captura de paquetes ICMP con "*tcpdump*".

Como puede observarse, "*tcpdump*" captura los paquetes ICMP que se le llegaron directamente desde FRR2 a FRRC, al estar directamente conectados, los paquetes llegan sin

ninguna dificultad, se distingue la distinción de los paquetes ICMP al observarse las solicitudes (*request*), respuestas (*reply*) y solicitudes de direcciones de capa de enlace (*Who has…*)

V.3.3.2.2. Con solicitud ICMP con destino a FRR1

Previo a esta medición, se ejecutó un envío de paquetes desde FRR2, con destino a FRR1.Con el comando `tcpdump -i ens5`, se obtuvieron los siguientes resultados:

```
root@FRRC:~# tcpdump -i ens5
[ 4401.281828] device ens5 entered promiscuous mode
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens5, link-type EN10MB (Ethernet), capture size 262144 bytes
01:15:20.622528 IP 10.10.10.9 > 224.0.0.5: OSPFv2, Hello, length 48
01:15:21.046595 IP 10.10.10.14 > 224.0.0.5: OSPFv2, Hello, length 48
01:15:30.622774 IP 10.10.10.9 > 224.0.0.5: OSPFv2, Hello, length 48
01:15:31.046069 IP 10.10.10.14 > 224.0.0.5: OSPFv2, Hello, length 48
01:15:40.622870 IP 10.10.10.9 > 224.0.0.5: OSPFv2, Hello, length 48
01:15:41.046211 IP 10.10.10.14 > 224.0.0.5: OSPFv2, Hello, length 48
01:15:50.622909 IP 10.10.10.9 > 224.0.0.5: OSPFv2, Hello, length 48
01:15:51.046234 IP 10.10.10.14 > 224.0.0.5: OSPFv2, Hello, length 48
01:16:00.622956 IP 10.10.10.9 > 224.0.0.5: OSPFv2, Hello, length 48
01:16:01.046258 IP 10.10.10.14 > 224.0.0.5: OSPFv2, Hello, length 48
01:16:10.623002 IP 10.10.10.9 > 224.0.0.5: OSPFv2, Hello, length 48
01:16:11.046228 IP 10.10.10.14 > 224.0.0.5: OSPFv2, Hello, length 48
01:16:20.623228 IP 10.10.10.9 > 224.0.0.5: OSPFv2, Hello, length 48
01:16:21.046206 IP 10.10.10.14 > 224.0.0.5: OSPFv2, Hello, length 48
01:16:30.623299 IP 10.10.10.9 > 224.0.0.5: OSPFv2, Hello, length 48
01:16:31.046249 IP 10.10.10.14 > 224.0.0.5: OSPFv2, Hello, length 48
^C
16 packets captured
16 packets received by filter
0 packets dropped by kernel
[ 4481.732956] device ens5 left promiscuous mode
root@FRRC:~#
```

Figura 131. Captura general de paquetes en interfaz ens5 de FRRC.

Como puede observarse, la interfaz captura de forma continua los paquetes *hello* de OSPF con sus horas exactas de recepción, la diferencia entre cada paquete *hello* de cada interfaz del enlace se envían periódicamente, cumpliendo con los tiempos de *hello* determinados en el capítulo anterior. No obstante, nótese que la captura no registra los paquetes ICMP de ningún tipo, tanto de solicitud. como de respuesta y tampoco de solicitud de dirección de capa de enlace.

Cuando FRR2 intenta comunicarse con FRR1, FRRC no recibe sus paquetes de solicitud, por lo que no es capaz de reenviarlos hacia su destino en FRR1.

De esta situación, se concluye que el uso de instancias como elementos enrutadores, hace forzar la gestión de las funciones de red a *Nova*, gestor del aprovisionamiento y funcionamiento de cómputo de las instancias, en lugar de ser *Neutron* el gestor de las funciones de red, entre las que se incluye el direccionamiento de paquetes incluyendo la función de virtualización de

elementos enrutadores embebida, de forma nativa; caso contrario de FRR, que es un *software* que fue forzado a usarse en la infraestructura de la nube.

V.3.4. Comparación entre los recursos utilizados originalmente para las prácticas de laboratorio con respecto a los recursos de la nube

Luego de haber desplegado la nube y analizar las distintas actividades de los laboratorios de telemática I y II, las prácticas impartidas en los mismos, pueden utilizar la infraestructura de la nube de la siguiente manera, mostrada en las siguientes tablas (ver Tabla 19 y Tabla 20):

Asig.	Práctica	Herramienta actual	¿Puede usarse la nube?	Observación / Uso de *MICROSTACK*
Lab. de telemática I	#1 - Análisis de las unidades de datos de protocolo (PDU) (modelo tcp/ip) usando *Wireshark*	*Wireshark*, analizador de paquetes TCP/IP	Sí	Es posible equipar las instancias con un analizador de paquetes, como "tcp-dump" nativo para Linux, o incluso el mismo *WireShark.*
	#2 - *Ethernet* clásica - Estudio de la capa de acceso al medio, el *hub*, dominios de colisión y protocolo de resolución de direcciones	No aplica	No	*MICROSTACK* no realiza simulación de dispositivos de capa física ni de acceso al medio.
	#3 - *Ethernet* conmutada - Estudio del switch, direcciones MAC y dominios de *Broadcast*.	CISCO *Packet Tracer*	No	El enfoque de diseño de las redes en *MICROSTACK* está en la capa de red, de forma nativa, este no provee virtualización de *switches* en los que se estudie la capa de enlace.
	#4 - Red de área local virtual (VLAN) - Reducción de dominios de broadcast en *switch.*	*Switches* físicos	No	*MICROSTACK* no provee simulación de dispositivos de capa enlace, como el *switch*.
	#5 - Estudio del protocolo IP - Direcciones IP, máscaras, subredes, VLSM.	Wireshark, analizador de paquetes TCP/IP	Sí	*MICROSTACK* ofrece una plataforma para la creación de redes de distintas dimensiones lógicas, con distintas máscaras de subred y delimitación de direcciones IP. Además de la utilización de analizadores de paquetes.
	#6 - Introducción al enrutamiento estático - Rutas por interfaz de salida, rutas por dirección de siguiente salto y rutas por defecto.	VirtualBox - Máquinas virtuales con S.O. basados en Linux	Sí	Por defecto, *Neutron*, módulo gestor de las funciones de red, ofrece la virtualización de *routers* en los que se puede configurar rutas estáticas. No obstante, sólo es posible por medio de la dirección IP de siguiente salto.
	#7 - Estudio de los protocolos TCP-UDP - Protocolos de capas superiores, conexiones cliente/servidor, servidores DHCP y SSH	VirtualBox - Máquinas virtuales con S.O. basados en Linux	Sí	*Neutron* tiene embebida una función de servidor DHCP para los despliegues realizados. Al poder crear instancias con imágenes de distintos sistemas operativos, pueden utilizarse para la ejecución de protocolos de este tipo, como el acceso remoto SSH.

Tabla 19. Actividades del Laboratorio de Telemática I que le sacan provecho a la nube [1].

Asig.	Práctica	Herramienta actual	¿Puede usarse la nube?	Observación / Uso de *MICROSTACK*
Lab. de telemática II	#1 - Configuración básica del Router - Nombre de equipo, direcciones IP, contraseñas y guardados de configuración	*Graphical Network Simulator 3 (GNS3)* y *VirtualBox* con Máquinas Linux.	Sí	*Neutron* ofrece configuración básica de los routers vortuales que provee, no obstante, a nivel mucho más básico que el estudiado en la práctica. Si se desea estudiar a un nivel similar, es necesario utilizar una instancia con *software* de enrutamiento, como *Quagga* o *FRRouting*, que no funcionan del todo.
	#2 - Repaso de enrutamiento estático - Rutas con direcciones de siguiente salto, rutas con interfaz de salida, rutas con interfaces *Ethernet* y rutas predeterminadas.	CISCO *Packet Tracer*	Parcialmente	*Neutron* ofrece función de enrutamiento estático, no obstante, sólo se puede implementar con dirección de siguiente salto, y no se simula la especicación del tipo de medio físico para las interfaces de los *routers.*
	#3 - Protocolo de enrutamiento dinámico - RIPv2, convergencias, métricas y parámetros.	CISCO *Packet Tracer* y *WireShark*	Parcialmente.	Por defecto, *Neutron* no ofrece función de enrutamiento dinámico, por lo que se debe utilizar una instancia como router equipada con un *software* de enrutamiento, como *Quagga* o *FRRouting*, los mismos soportan RIPv2 y la convergencia de la red, sin embargo, sin comunicación ICMP.
	#4 - Protocolo *Open Shortest Path First* (OSPF) - Paquetes *hello*, adyacencias, vecindades, costos, configuraciones y parámetros.	CISCO *Packet Tracer* / VirtualBox con Máquinas Linux equipadas con *Zebra.*	Parcialmente	Al igual que con RIPv2, se puede usar una instancia Linux equipada con Quagga o FRRouting para implementar OSPF y observar los parámetros descritos y la convergencia de la red, no obstante, sin comunicación ICMP.
	#5 - Filtrado de Paquetes y NAT - Listas de control de acceso, bloqueo y habilitación de paquetes, reglas y cadenas de *firewall*	CISCO *Packet Tracer* / VirtualBox con Máquinas Linux equipadas con *Zebra.*	Parcialmente	*MICROSTACK* provee funciones de cadenas de *firewall* para las instancias creadas, no obstante, se pueden implementar de forma manual utilizando el firewall de Linux en las mismas. No obstante, si se usan instancias como routers, no habrá comunicación ICMP, como en los casos de los protocolos de enrutamiento dinámico.

Tabla 20. Actividades del Laboratorio de Telemática II que le sacan provecho a la nube [2].

CAPÍTULO VI. CONCLUSIONES Y RECOMENDACIONES

Del presente proyecto realizado, se pueden destacar y concluir algunos aspectos importantes y relevantes, que parten desde detalles y observaciones sobre el desarrollo del producto y su utilización, hasta para implementaciones o despliegues similares al desarrollado o para futuras y factibles implementaciones o mejoras al producto actual.

VI.1. Conclusiones

En la actualidad, el mundo de la telemática se encuentra expuesto al crecimiento continuo y de gran escala de las plataformas de nube, las cuales permiten suministrar servicios a clientes finales y proveedores de servicios de telecomunicaciones e Internet, a través de diferentes formas, dentro de las cuales resaltan la provisión de infraestructura como servicio (IaaS), de plataforma como servicio (PaaS), y de *software* como servicio (SaaS).

Teniendo presentes las ideas previas, se seleccionó el *software MICROSTACK* para el despliegue de la nube, y haciendo uso de este último se consiguió implementar en el Laboratorio de Telemática una infraestructura de nube con cinco (5) computadoras, las cuales cuentan con el sistema operativo UBUNTU; una actúa como nodo de control y de cómputo, y el resto como nodos de cómputo exclusivamente. A partir de dicha nube, fue posible reservar recursos de cómputo, almacenamiento y red.

Lo anteriormente descrito permitió determinar que *MICROSTACK* hace posible llevar a cabo la instalación simple de una versión preconfigurada y reducida de *OPENSTACK*, dando lugar al despliegue rápido y eficaz de una nube, que puede ser útil en escenarios donde se requiera una nube sin funcionalidades de mayor complejidad, y en caso de llegar a necesitarse, se puede ampliar la nube mediante el esquema convencional de instalación de módulos de *OPENSTACK*. La gestión de la nube no constituye mayor dificultad, pues *MICROSTACK* incorpora el *dashboard* o interfaz gráfica de *OPENSTACK*.

En el mismo orden de ideas previo, se debe mencionar que *MICROSTACK* permite el despliegue de nubes de diferentes dimensiones, desde nubes con un solo nodo, como la “micro nube”, hasta nubes con múltiples nodos, como la implementada en el Laboratorio de Telemática.

La nube desplegada permitió hacer un uso más eficiente de los recursos de *hardware* de las computadoras del Laboratorio de Telemática, a la vez que incorporó en el banco de conocimiento de la Escuela los tópicos de computación en la nube, sin comprometer los usos cotidianos y prácticos que actualmente se les da a las máquinas, pues los procesos de gestión y operación de la nube tienen lugar en segundo plano.

Por otro lado, para evaluar el desempeño y rendimiento de la infraestructura de la nube, se realizaron un grupo de pruebas que consistieron en la medición de los tiempos de creación y encendido de instancias, y fueron comparados con los tiempos de creación y encendido de máquinas virtuales tradicionales, utilizando el *software VirtualBox*. En ambos entornos, las pruebas fueron realizadas utilizando dos configuraciones de memorias para las instancias, correspondientes a dos *flavors* distintos de *MICROSTACK*.

Tras medir los tiempos de creación de instancias y analizar sus resultados, se concluye que en la virtualización de los distintos recursos y funciones que provee la infraestructura de la nube para el aprovisionamiento de instancias con *MICROSTACK*, no se encontraron diferencias perceptibles en los tiempos de creación de las instancias con distintas configuraciones de *flavors*, creadas en los diferentes nodos que la conforman, y al ser comparados con los medidos en *VirtualBox*, se percibe una diferencia considerable en función de la configuración de memorias asignada a cada máquina virtual.

En cuanto a los tiempos de encendido de las instancias se concluye que la nube tiene un desempeño regular que depende directamente del tipo de nodo en el que se hayan creado las instancias. Si se crean en el nodo de control, los tiempos medidos son comparables, sin diferencias perceptibles, con los medidos en el entorno de *VirtualBox*; sin embargo, cuando las instancias se crean en un nodo de cómputo, el tiempo de encendido es considerablemente mayor, independientemente de la capacidad de memorias del *flavor* utilizado.

Adicionalmente, para complementar el desempeño de la implementación de la nube, se midieron las latencias existentes entre cada uno de los nodos que la conforman y las instancias creadas dentro de cada uno de los mismos. De las pruebas, se concluye que la implementación de nodos múltiples no supone una latencia importante que pueda lastrar mínimamente la comunicación de los elementos virtualizados.

Para evaluar la accesibilidad remota de la nube, empleando PuTTY desde una computadora externa a la nube, fue posible establecer conexión remota con el nodo de control, y desde esa sesión remota, como se refleja en los resultados, se logró interactuar con los recursos de la nube desde dicha computadora y ejecutar sus funciones, como la creación y eliminación de instancias, a las que incluso se logró acceder nuevamente con una nueva conexión SSH en cascada, luego de haber establecido la primera. Con estas acciones, se concluye que la nube es accesible de forma remota y puede gestionar sus recursos de esta forma con éxito.

Otra de las premisas principales por las que se implementó la nube fue evaluar su posible uso académico, para ello, se procedió a probar la efectividad de la nube al ejecutar un grupo de aplicaciones y funcionalidades impartidas en las prácticas de Laboratorio de Telemática I y II. De esta prueba se observó que, la infraestructura de nube implementada permite llevar a cabo, de manera limitada, algunos de los tópicos abordados en las prácticas de los laboratorios mencionados, más no todos.

Las pruebas realizadas sobre las prácticas de los Laboratorio de Telemática demostraron que es posible implementar esquemas de enrutamiento estático sin inconvenientes y con resultados positivos utilizando las funcionalidades embebidas del módulo *Neutron*; sin embargo, se encontraron limitantes al desplegar topologías que empleasen protocolos de enrutamiento dinámico como consecuencia de las restricciones del módulo *Neutron* en *MICROSTACK*, el cual no admite que las instancias actúen como enrutadores, al no permitir la comunicación de paquetes entre instancias no adyacentes. Tras analizar estos resultados, se concluye que el proyecto tiene potencial para ser utilizado como la herramienta de estudio del tópico de implementación de esquemas de *cloud computing*, de manera independiente a las prácticas de laboratorio, que es la competencia que se quiere abordar y añadir al estudio académico en la carrera. Esto permitirá desarrollar las habilidades de implementación de nubes, mediante experiencias de ejemplo y modelo de forma independiente a las prácticas de laboratorio ya existentes.

VI.2. Recomendaciones

Luego de finalizar el recorrido de la realización del presente proyecto, para futuras implementaciones similares o mejoras del mismo, se recomiendan los siguientes aspectos.

VI.2.1. Conocimiento del S.O. UBUNTU

Fundamentalmente, se recomienda la documentación o estudio previo de los comandos de consola de UBUNTU u otros sistemas derivados de LINUX, específicamente aquellos relacionados a: Manipulación de archivos y directorios, jerarquías de permisos, funciones de red (direccionamiento IP, enrutamiento, captura de paquetes), *firewall*, e instalación de paquetes y programas.

Recomendaciones de entorno de despliegue

Para combatir y evitar las demás eventualidades mencionadas, externas al *software* de UBUNTU, se recomienda tomar las siguientes acciones y medidas para acondicionar el entorno:

- Disponer de una versión reciente del sistema UBUNTU, al menos la versión 20.04.3 LTS, y preferiblemente la última disponible con soporte a largo plazo. Se recomienda equipar la misma en todos los nodos.
- Utilizar la misma versión de la aplicación *MICROSTACK*, proveniente del mismo canal de instalación, se recomienda específicamente la versión que se indica en la página oficial de los desarrolladores: `--beta` [42].
- Utilizar una infraestructura de red libre de cualquier *software* de gestión masiva de seguridad en la red. Evitar los corta fuegos estrictos, servidores proxy, etc.
- Implementar direccionamiento IP estático, no variable.
- Verificar el óptimo funcionamiento de los medios físicos de transmisión de datos, tanto guiados, como no guiados:
  - De utilizar comunicación inalámbrica, disminuir lo más posible la cantidad de obstáculos que generen pérdidas o interferencia en las señales de radio, que puedan repercutir en la conexión de los nodos.
  - De utilizar comunicación cableada, asegurar que los cables *Ethernet* o *patch cords*, estén bien construidos, con las puntas y conectores bien colocados y sin roturas o dobleces bruscos.

VI.2.2. Recomendaciones para presunta mejora del proyecto

Estas recomendaciones contemplan varios aspectos a tomar en cuenta si se considera implementar una mejora en un futuro a mediano plazo del proyecto, distribuidas en las siguientes categorías:

Para ampliar el abanico de funciones de cada módulo del *software* gestor de la nube, se recomienda lo siguiente:

- Descargar cada módulo de *OPENSTACK* de forma independiente, fuera del paquete *snap* de *MICROSTACK*.
- Reutilizar los módulos previamente instalados en *MICROSTACK*.

VI.2.2.1. Para funcionalidad de enrutamiento dinámico

Se recomienda implementar las funciones que ofrece la instalación independiente del módulo *Neutron* de *OPENSTACK*. El mismo, cuenta con un conjunto de funciones embebidas, deshabilitadas por defecto, que permiten implementar protocolos de enrutamiento dinámico, específicamente el Protocolo de Puerta de Enlace de Frontera (*Border Gateway Protocol*, BGP). Existe documentación oficial de *Neutron* para implementar este protocolo [43] [44], por lo que se recomienda su revisión y aplicación en caso de necesitar su uso.

VI.2.3. Recomendaciones para el uso académico de la nube

VI.2.3.1. Estudio independiente de las prácticas

Si se reservan espacios en las asignaturas de laboratorio para el estudio de la nube utilizando *MICROSTACK*, se recomiendan las siguientes acciones:

- Elaborar prácticas específicas para llevar a cabo el estudio de la implementación y uso de la infraestructura de nube.
- Evitar migrar completamente las prácticas actuales desde las herramientas de su estudio para la infraestructura de la nube.
- No adaptar *software* que no sea compatible de forma nativa con *MICROSTACK*, o que no esté embebido en el mismo, debido a los conflictos de compatibilidad que pueden presentarse.

VI.2.3.2. Forma sugerida de segmentación de asignaturas, prácticas y usuarios

Esta plataforma, además de los servicios que provisiona, podría soportar el segmentado de los múltiples proyectos, diseños e implementaciones que el/los estudiantes de las diferentes asignaturas desarrollen por medio de la gestión de identidades de uso de la nube pertenecientes al servicio *Keystone*, tales, como: Dominios, proyectos, grupos, usuarios y contraseñas.

La gestión de los proyectos y prácticas de laboratorio que usen la nube podrían gestionar sus respectivas identidades de la siguiente forma ejemplificada:

- Creación de múltiples **dominios**, uno para cada **asignatura**.
- Creación e inclusión de **usuarios** en su **dominio respectivo**, **de acuerdo a la asignatura** que estén cursando.
- Creación de **grupos** de usuarios por cada **mesa del laboratorio** y su respectiva inclusión de los usuarios correspondientes a los estudiantes que las conformen.
- Creación y distinción de **proyectos** por **cada práctica de laboratorio**, con sus respectivos elementos privados o compartidos, de acuerdo si se necesitan utilizar en prácticas posteriores.

BIBLIOGRAFÍA

[1] Universidad Católica Andrés Bello, «Programa de Asignatura - Telemática I,» [En línea]. Available: http://w2.ucab.edu.ve/tl_files/Ingenieriatelecom/Pensum/sexto_semestre/TELE-00047.pdf. [Último acceso: 2 Octubre 2021].

[2] Universidad Católica Andrés Bello, «Programa de Asignatura - Telemática II,» [En línea]. Available: http://w2.ucab.edu.ve/tl_files/Ingenieriatelecom/Pensum/septimo_semestre/TELE-00052.pdf. [Último acceso: 2 Octubre 2021].

[3] Universidad Católica Andrés Bello, «Programa de Asignatura - Telemática IV,» [En línea]. Available: http://w2.ucab.edu.ve/tl_files/Ingenieriatelecom/Pensum/noveno_semestre/TELE-00089.pdf. [Último acceso: 2 Octubre 2021].

[4] Canonical Limited, «Microstack - Openstack in a snap,» Canonical Limited, [En línea]. Available: https://microstack.run/docs. [Último acceso: 2 Octubre 2021].

[5] La Tercera, «Cómo el cloud computing está influyendo en las ciudades - La Tercera,» 30 Abril 2022. [En línea]. Available: https://www.latercera.com/pulso/noticia/como-el-cloud-computing-esta-influyendo-en-las-ciudades/KHZNLV5VZ5GVJBPNXMX7VVNVQU/. [Último acceso: 20 Mayo 2022].

[6] Microsoft Corporation, «¿Qué es IaaS? Infraestructura como Servicio | Microsoft Azure,» Sin fecha. [En línea]. Available: https://azure.microsoft.com/es-es/overview/what-is-iaas/#overview. [Último acceso: 21 Mayo 2022].

[7] D. A. Tong y K. Wade, «Guía de NFV y SDN para operadores y proveedores de servicio,» 2017. [En línea]. Available: https://media.ciena.com/documents/Blue+Planet+Essentials_NFV+and+SDN+Guide_033017_A5_es_LA.pdf. [Último acceso: 2 Octubre 2021].

[8] D. Marinescu, Cloud ComputingTheory and Practice, Cambridge, MA.: Elsevier Inc., 2018.

[9] B. Kezherashvili, «Computación en la Nube,» *trabajo de fin de máster.*

[10] ManagementSolutions, «La nube: oportunidades y retos para los integrantes de la cadena de valor,» 2012. [En línea]. Available: https://www.managementsolutions.com/sites/default/files/publicaciones/esp/La-nube.pdf. [Último acceso: 2 Octubre 2021].

[11] Microsoft Azure, «¿Qué es la nube?,» [En línea]. Available: https://azure.microsoft.com/es-es/overview/what-is-the-cloud/. [Último acceso: 2 Octubre 2021].

[12] A. S. Tanembaun, Sistemas operativos modernos, Tercera ed., Naucalpan de Juárez: PEARSON EDUCACIÓN, 2009.

[13] D. J. W. Andrew S. Tanenbaum, Redes de Computadoras, Quinta ed., Naucalpan de Juárez: PEARSON EDUCACIÓN, 2012.

[14] Ubuntu, «Our Mission | Ubuntu,» 2022. [En línea]. Available: https://ubuntu.com/community/mission. [Último acceso: 30 Enero 2022].

[15] R. Chayapathi, S. Hassan y P. Shah, Network Functions Virtualization (NFV) with a Touch of SDN, Addison-Wesley Professional, 2017.

[16] Openstack, «Open Source Cloud Computing Platform *Software* - Openstack,» [En línea]. Available: https://www.openstack.org/*software*/. [Último acceso: 2 Octubre 2021].

[17] Red Hat, «El concepto de OpenStack,» Red Hat, Inc., [En línea]. Available: https://www.redhat.com/es/topics/openstack. [Último acceso: 2 Octubre 2021].

[18] OpenStack, «Open Source Cloud Computing Platform *Software* - OpenStack,» [En línea]. Available: https://www.openstack.org/*software*/project-navigator/openstack-components#openstack-services. [Último acceso: 31 Enero 2022].

[19] Canonical Limited, «Learn about OpenStack services and their functions | Ubuntu,» 2022. [En línea]. Available: https://ubuntu.com/tutorials/learn-about-openstack-services-and-their-functions#3-explore-the-keystone-service. [Último acceso: 11 Marzo 2022].

[20] Canonical Ltd, «Explore OpenStack components and set up an OpenStack client | Ubuntu,» 2022. [En línea]. Available: https://ubuntu.com/tutorials/explore-openstack-components-and-set-up-an-openstack-client#2-learn-about-openstack-components. [Último acceso: 11 Marzo 2022].

[21] Canonical Ltd., «Microstack - Openstack in a snap,» Canonical Ltd., [En línea]. Available: https://microstack.run/. [Último acceso: 2 Octubre 2021].

[22] VMware Latinoamérica, «¿Qué es una máquina virtual? | Glosario de VMware | LATAM,» 2022. [En línea]. Available: https://www.vmware.com/latam/topics/glossary/content/virtual-machine.html. [Último acceso: 27 Septiembre 2022].

[23] VMware Latinoamérica, «¿Qué es un hipervisor? | Glorario de VMware | LATAM,» 2022. [En línea]. Available: https://www.vmware.com/latam/topics/glossary/content/hypervisor.html. [Último acceso: 29 Septiembre 2022].

[24] Universidad Pedagógica Experimental Libertador, Manualde Trabajos de Gradode Especialización y Maestría y Tesis Doctorales, Caracas: Fondo Editorial de la Universidad Pedagógica Experimental Libertador, 2016.

[25] MicroStack, «OpenStack for the edge, micro clouds and developers | Single-Node,» 2022. [En línea]. Available: https://microstack.run/docs/single-node. [Último acceso: 14 Junio 2022].

[26] MicroStack, «OpenStack for the edge, micro clouds and developers | Multi-node,» 2022. [En línea]. Available: https://microstack.run/docs/multi-node. [Último acceso: 14 Junio 2022].

[27] Canonical Ltd, «Use the concept of domains, roles, users and groups to manage identities | Ubuntu,» 2022. [En línea]. Available: https://ubuntu.com/tutorials/use-the-concept-of-domains-roles-users-and-groups-to-manage-identities#2-manage-domains. [Último acceso: 13 Marzo 2022].

[28] Canonical Ltd, «Manage instances templates, including images and flavors | Ubuntu,» 2022. [En línea]. Available: https://ubuntu.com/tutorials/manage-instance-templates-including-images-and-flavors#2-manage-images. [Último acceso: 13 Marzo 2022].

[29] Canonical Limited, «Ubuntu Cloud Images - the official Ubuntu images for public clouds, Openstack, KVM and LXD,» 2022. [En línea]. Available: https://cloud-images.ubuntu.com/. [Último acceso: 27 Septiembre 2022].

[30] Canonical Ltd, «Learn how OpenStack manages various virtual network resources | Ubuntu,» 2022. [En línea]. Available: https://ubuntu.com/tutorials/learn-how-openstack-manages-various-virtual-network-resources#6-manage-floating-ips. [Último acceso: 13 Marzo 2022].

[31] OpenStack, «Get images --- Virtual Machine Image Guide documentation,» 2022. [En línea]. Available: Get images --- Virtual Machine Image Guide documentation. [Último acceso: 30 Agosto 2022].

[32] Debian, «Index of /cdimage/openstack,» 2022. [En línea]. Available: http://cdimage.debian.org/cdimage/openstack/. [Último acceso: 30 Agosto 2022].

[33] CentOS, «CentOS Cloud images,» 2022. [En línea]. Available: https://cloud.centos.org/centos/9-stream/. [Último acceso: 30 Agosto 2022].

[34] Fedora, «Fedora Cloud,» 2022. [En línea]. Available: https://alt.fedoraproject.org/cloud/. [Último acceso: 30 Agosto 2022].

[35] Microsoft Corporation, «Download Windows 10,» 2022. [En línea]. Available: https://www.microsoft.com/en-us/*software*-download/windows10. [Último acceso: 30 Agosto 2022].

[36] e. a. Kunihiro Ishiguro, «Quagga - A routing *software* package for TCP/IP networks,» [En línea]. Available: https://cs.uns.edu.ar/~ldm/data/rc/info/quagga.pdf. [Último acceso: 9 Septiembre 2022].

[37] FRRouting Project, «FRRouting,» 2022. [En línea]. Available: https://frrouting.org/. [Último acceso: 16 Septiembre 2022].

[38] International Business Machines Corporation, «OSPF (Open Shortest Path First) - Documentación de IBM,» 2022. [En línea]. Available: https://www.ibm.com/docs/es/i/7.2?topic=routing-open-shortest-path-first. [Último acceso: 16 Septiembre 2022].

[39] K. Koishigawa, «sudo apt-get update vs update - What is the difference?,» 2 Mayo 2022. [En línea]. Available: https://www.freecodecamp.org/news/sudo-apt-get-update-vs-upgrade-what-is-the-difference/#:~:text=The%20sudo%20apt%2Dget%20upgrade,want%20to%20perform%20the%20upgrades.. [Último acceso: 17 Septiembre 2022].

[40] FRRouting Project, «VTY shell -- FRR latest documentation,» 2022. [En línea]. Available: http://docs.frrouting.org/en/latest/vtysh.html#vty-shell. [Último acceso: 17 Septiembre 2022].

[41] Universidad Católica Andrés Bello, «Malla Curricular Escuela de Ingeniería de Telecomunicaciones Octubre 2021,» 20 Septiembre 2021. [En línea]. Available:

http://w2.ucab.edu.ve/tl_files/Ingenieriatelecom/202215/Grafo%20202215.pdf. [Último acceso: 30 Septiembre 2021].

[42] Canonical Limited, «Install OpenStack on your workstation and launch your first instance | Ubuntu,» 2022. [En línea]. Available: https://ubuntu.com/tutorials/install-openstack-on-your-workstation-and-launch-your-first-instance#1-overview. [Último acceso: 10 Julio 2022].

[43] OpenStack, «Welcome to neutron-dynamic-routing's documentation! --- neutron-dynamic-routing 21.1.0.dev2 documentation,» 2022. [En línea]. Available: https://docs.openstack.org/neutron-dynamic-routing/latest/index.html. [Último acceso: 18 Septiembre 2022].

[44] OpenStack, «BGP dynamic routing --- Neutron 21.1.0.dev2 documentation,» 2022. [En línea]. Available: https://docs.openstack.org/neutron/latest/admin/config-bgp-dynamic-routing.html. [Último acceso: 18 Septiembre 2022].

[45] Canonical Ltd, «Navigate through the OpenStack dashboard menu | Ubuntu,» 2022. [En línea]. Available: https://ubuntu.com/tutorials/navigate-through-the-openstack-dashboard-menu#2-change-admin-user-password. [Último acceso: 13 Marzo 2022].

[46] OpenNebula Systems, «OpenNebula – Open Source Cloud & Edge Computing Platform,» [En línea]. Available: https://opennebula.io/. [Último acceso: 08 Julio 2022].

[47] Stackscale, «30 distribuciones de Linux populares [Lista],» 28 Julio 2022. [En línea]. Available: https://www.stackscale.com/es/blog/distribuciones-linux-populares/. [Último acceso: 17 Septiembre 2022].

[48] W3Techs, «Historical yearly trends in the usage stadistics of Linux subcategories, September 2022,» Septiembre 2022. [En línea]. Available: https://w3techs.com/technologies/history_details/os-linux/all/y. [Último acceso: 17 Septiembre 2022].

CAPÍTULO VII. ANEXOS

VII.1. Manual de usuario de la nube

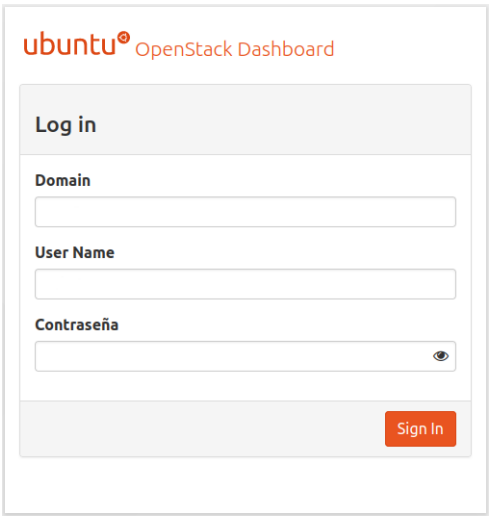
En la presente sección se plantea una guía de uso de la interfaz gráfica de usuario del tablero o *Dashboard* de *OPENSTACK*, con el objetivo de familiarizar y facilitar al lector el uso.

VII.1.1. Acceso al Dashboard

Para acceder al tablero principal de *OPENSTACK*, se debe hacer uso de una aplicación de navegación a Internet e ingresar la dirección IP de la interfaz virtual del nodo de control, la 10.20.20.1, dependiendo del navegador que se utilice, puede aparecer un mensaje de ingreso inseguro, el cual debe ser ignorado y solicitar continuar con el acceso. En otro tipo de implementación del módulo *Horizon* de forma independiente, esta dirección IP puede cambiar.

```
http://10.20.20.1/
```

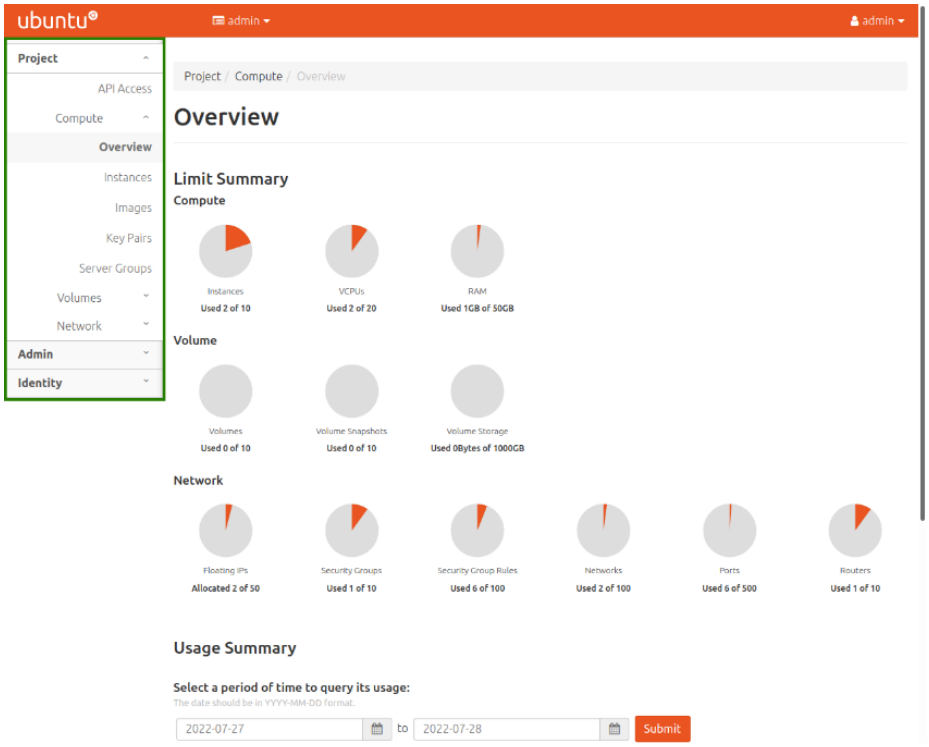
Luego de ingresar a esta dirección, en seguida, se mostrará la plataforma de acceso al *dashboard*, en la que deben ingresar las credenciales de acceso compuestas por Dominio, usuario y contraseña, según sea el caso.



Al ingresar, se muestran un sumario de uso general de los recursos en la nube dentro del proyecto consultado y en el lateral izquierdo.

VII.1.2. Pestañas de funciones

El *dashboard* de *OPENSTACK*, está divido en 3 macro grupos principales: *Project*, *Admin* e *Identity*. Cada uno de ellos, maneja un conjunto de opciones y funciones correspondientes al proyecto general de nube en que se encuentra, visualización global desde el punto de vista del administrador, y la gestión de identidades nuevas para la nube, respectivamente.



VII.1.2.1. Grupo Project

El grupo *Project* es el responsable de la gestión de los recursos del proyecto actual de la infraestructura de nube [45], consta de cuatro subgrupos: *API Access*, *Compute*, *Volumes* y *Network*, que serán descritos a continuación:

VII.1.2.1.1. Pestaña API Access

Proporciona información sobre los puntos finales (*endpoints*) de la Interfaz de Programación de Aplicaciones (*API*) de los servicios de *OPENSTACK* [45].

VII.1.2.1.2. Pestaña Compute

Proporciona acceso a los recursos informáticos del proyecto en cuestión, entre los que están las instancias, imágenes, llaves de acceso, y grupos de seguridad [45].

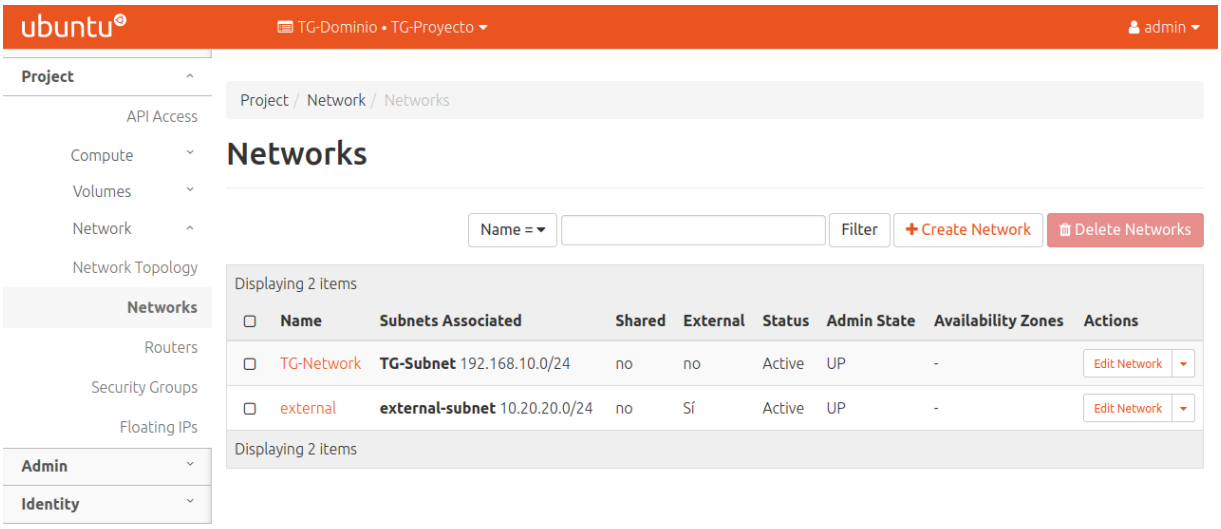
- ***Instances*:** Este menú permite la interacción directa con las instancias, permitiendo su creación, borrado, encendido y apagado, asignación de direcciones IP, entre otros múltiples parámetros de edición.
- ***Images*:** Desde este menú se gestionan las imágenes de sistemas operativos para las instancias, su creación, eliminación y disponibilidad entre los proyectos de la nube.
- ***Key Pairs*:** Gestión de las llaves de acceso remoto a las instancias vía SSH.
- ***Server Groups*:** Permite crear grupos de instancias de acuerdo a lo que el usuario requiera.

VII.1.2.1.3. Pestaña Volumes

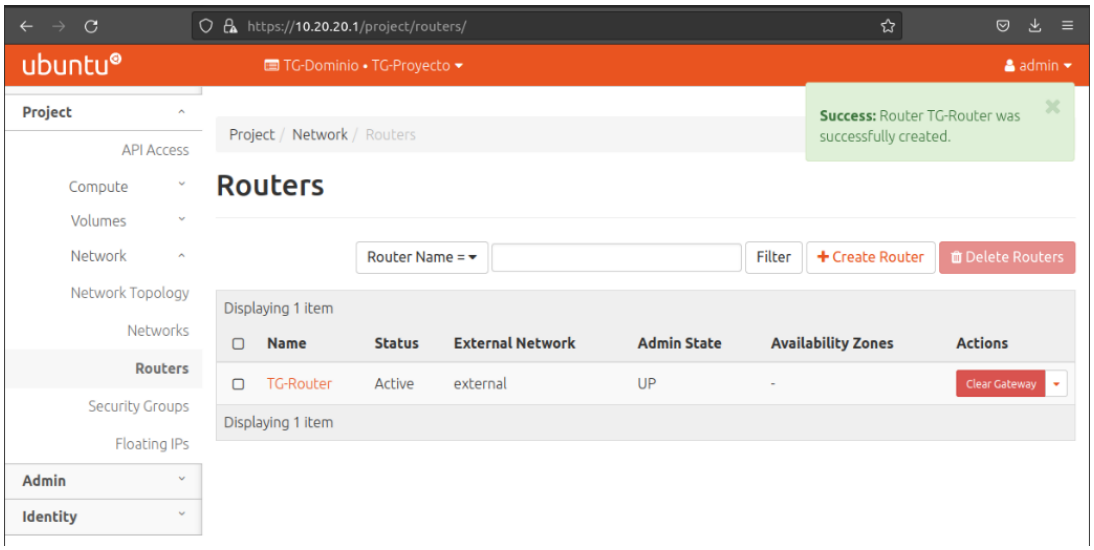
Proporciona acceso a los recursos de almacenamiento en disco del proyecto, como los volúmenes de almacenamiento en bloque persistentes [45].

VII.1.2.1.4. Pestaña Network

Proporciona acceso a los recursos y funciones de red del proyecto en cuestión, como las redes virtuales, topologías, *routers*, entre otras funciones [45].



- ***Network Topology*:** Desde acá se visualizan dos tipos de diagramas de la topología de red desplegada en el proyecto que se haya ingresado: Diagrama básico y diagrama gráfico.
- ***Networks*:** Desde este menú se gestionan las redes virtualizadas en el proyecto en el que se haya ingresado dentro del entorno de la nube. Pueden crearse redes y subredes de distinto direccionamiento IP.
- ***Routers*:** Gestionan los elementos enrutadores o *routers* virtualizados por *Neutron*, se pueden añadir y eliminar interfaces, asignar subredes a las mismas, direcciones IP y establecer rutas estáticas.

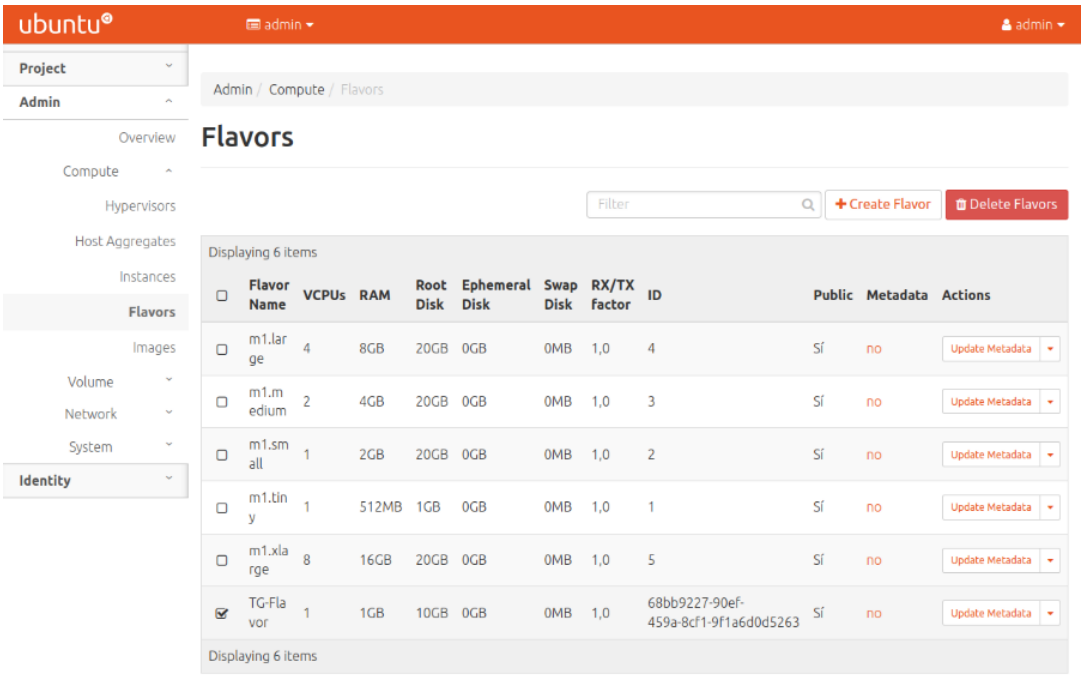


- ***Secrurity Groups*:** Permite la gestión de distintos parámetros relacionados a la seguridad de la red, como habilitación y bloqueo de protocolos de puertos, ingreso restrictivo de redes y acceso SSH.
- ***Floating IPs*:** Gestiona las direcciones IP flotantes, que permiten a las instancias ser identificadas fuera de la infraestructura de la nube. La traducción a las direcciones reales de las mismas las realiza el nodo de control con el protocolo NAT.

VII.1.2.2. Grupo Administrador

Proporciona acceso a todos los recursos globales que se utilizan en los distintos proyectos de la nube, y permite la gestión de los mismos a nivel de administrador [45]. A su vez, este se encuentra dividido en cinco pestañas:

- **Pestaña *Overview*:** Muestra todas las métricas de uso de cada uno de los proyectos creados [45].
- **Pestaña *Compute*:** Proporciona acceso a los recursos informáticos globales, incluyendo hipervisores [45]. También permite la gestión de todas las instancias, imágenes, *key pairs* de todos los proyectos, con permisos de administrador.
    - ***Flavors*:** Permite la gestión de los perfiles de configuración predefinida de las memorias para ser equipadas en las instancias.



- **Pestaña *Volume*:** Proporciona acceso a los recursos de almacenamiento globales, como los tipos de volumen [45]. Además, soporta la gestión de los mismos con permisos de administrador.
- **Pestaña *Network*:** Proporciona acceso a los recursos globales de las redes de todos los proyectos de nube, incluyendo el Control de Acceso Basado en Roles (*Role Based Access Control*, RBAC) [45]. Permite la gestión de los mismos a nivel de administrador.
- **Pestaña *System*:** Proporciona acceso a la configuración global de la infraestructura, como los límites de uso [45].

VII.1.2.3. Grupo Identity

Se encarga de proporcionar funciones de gestión de identidades para el manejo de la infraestructura de la nube. Permite la gestión de proyectos, usuarios, grupos, roles y credenciales de la aplicación. Para cada una de las funciones mencionadas, el grupo *Identity* tiene un subgrupo asociado:

- **Pestaña *Projects*:** Proporciona acceso a las cuentas de proyectos generales de nubes, a los que se les asocian instancias, redes y grupos de usuarios.
- **Pestaña *Users*:** Proporciona la gestión, creación y eliminación de cuentas de usuarios para los distintos proyectos.
- **Pestaña *Groups*:** Permite la gestión, creación y eliminación de grupos de usuarios, con opciones definidas para cada uno de los mismos.
- **Pestaña *Roles*:** Proporciona acceso a los roles de los usuarios.
- **Pestaña *Application credentials*:** Gestiona las credenciales de la aplicación de *OPENSTACK*, para el *dashboard* y para el cliente de consola *OPENSTACK Client*.