Despliegue modular de la plataforma de *cloud computing* OpenStack en la red de laboratorios del Departamento de Electrónica de la Universidad del Valle de Guatemala

Francisco Javier López Turcios

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Despliegue modular de la plataforma de *cloud computing* OpenStack en la red de laboratorios del Departamento de Electrónica de la Universidad del Valle de Guatemala

Trabajo de graduación presentado por Francisco Javier López Turcios
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2025

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Despliegue modular de la plataforma de *cloud computing* OpenStack en la red de laboratorios del Departamento de Electrónica de la Universidad del Valle de Guatemala

Trabajo de graduación presentado por Francisco Javier López Turcios
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2025

Vo.Bo.:

(f) _____

M. Sc. Carlos Esquit

Tribunal Examinador:

(f) _____

M.Sc. Carlos Esquit

(f) _____

Dr. Luis Alberto Rivera Estrada

(f) _____

Ing. Kurt Emmanuel Kellner

Fecha de aprobación: Guatemala, 13 de febrero de 2025.

Prefacio

Quiero expresar mi más sincero agradecimiento a mi familia, por su apoyo incondicional, y por haberme formado como persona, inculcando en mí la importancia del estudio. A mis amigos, por su respaldo constante. A la Fundación Juan Bautista Gutiérrez, por confiar en mis sueños y brindarme su apoyo a lo largo de mi trayectoria universitaria. Al Centro Universitario Ciudad Vieja, por acogerme y hacerme sentir en casa durante mis estudios. Y finalmente, a la Universidad del Valle de Guatemala y a sus catedráticos, por su valiosa contribución en mi formación académica.

Índice

Lista de figuras

Lista de cuadros

Resumen

El objetivo de este proyecto fue desplegar y documentar la implementación de una plataforma de computación en la nube utilizando la plataforma de código abierto OpenStack en la red de laboratorios del Departamento de Electrónica de la Universidad del Valle de Guatemala (UVG). La plataforma fue diseñada para optimizar el uso de una computadora de alto rendimiento (HPC) del departamento, la cual se encontraba subutilizada, habilitándola para ejecutar y gestionar múltiples instancias virtualizadas.

El despliegue se realizó de forma modular, implementando cada componente de OpenStack de manera independiente y utilizando una combinación de infraestructura física y virtual. Los servicios de OpenStack instalados incluyeron Keystone para la gestión de identidad, Glance para el almacenamiento de imágenes de máquinas virtuales, Placement para la asignación de recursos, Nova para la creación de instancias, Neutron para la conectividad de red, Cinder para el almacenamiento en bloque y Horizon como interfaz gráfica.

Como resultado, se obtuvo un entorno capaz de proveer Infraestructura como Servicio (IaaS) que no solo optimiza los recursos del departamento, sino que también ofrece una exploración de la configuración y funcionamiento de un ambiente de nube. Esta implementación permite a estudiantes y personal de la UVG familiarizarse con tecnologías de *cloud computing*, promoviendo un entorno de aprendizaje práctico y relevante. Un reto significativo que se encontró fue que el hipervisor KVM tiene soporte nativo únicamente para los formatos de disco "qcow2" y "raw", lo que limitó la compatibilidad con otros formatos de imágenes que se intentó utilizar, este problema puede ser evitado usando herramientas externas para la conversión de formatos de disco hacia "qcow2".

Entre las áreas que podrían explorarse en futuros trabajos se incluyen la configuración de más nodos de cómputo y almacenamiento para ampliar los recursos disponibles de la plataforma; la incorporación de servicios adicionales de OpenStack para expandir sus funcionalidades; la implementación de las modificaciones necesarias para transformar la prueba de concepto actual en un entorno seguro y listo para la producción de una nube profesional; y la exploración de otras alternativas de despliegue de OpenStack con el fin de contrastar las ventajas y desventajas de cada enfoque.

Abstract

The objective of this project was to deploy and document the implementation of a cloud computing platform using the open-source solution OpenStack within the lab network of the Electronics Department at Universidad del Valle de Guatemala (UVG). The platform was designed to optimize the use of a high-performance computer (HPC) in the department, which was previously underutilized, by enabling it to run and manage multiple virtualized instances.

The deployment was carried out in a modular manner, independently implementing each OpenStack component and using a combination of physical and virtual infrastructure. The installed OpenStack services included Keystone for identity management, Glance for virtual machine image storage, Placement for resource allocation, Nova for instance creation, Neutron for network connectivity, Cinder for block storage, and Horizon as the graphical interface.

As a result, an environment capable of providing Infrastructure as a Service (IaaS) was obtained. This not only optimizes the department's resources but also offers an in-depth exploration of the configuration and operation of a cloud environment. This implementation enables UVG students and staff to familiarize themselves with cloud computing technologies, fostering a practical and relevant learning environment. A significant challenge encountered was that the KVM hypervisor natively supports only "qcow2" and "raw" disk formats, which limited compatibility with other image formats attempted during the process. This issue can be mitigated by using external tools to convert disk formats to "qcow2."

Future work could include configuring additional compute and storage nodes to expand the platform's available resources; incorporating additional OpenStack services to extend its functionalities; implementing the necessary modifications to transform the current proof of concept into a secure, production-ready professional cloud environment; and exploring other OpenStack deployment alternatives to contrast the advantages and disadvantages of each approach.

CAPÍTULO 1

Introducción

La industria de centros de procesamiento de datos –*data centers*– ha experimentado un crecimiento exponencial en los últimos años, transformando la manera en que las organizaciones administran y procesan grandes volúmenes de información. Esta expansión responde a una demanda global de servicios de computación en la nube, que permite a empresas y usuarios acceder a recursos informáticos robustos, escalables y eficientes. Aunque el uso de plataformas de computación en la nube es común en muchas organizaciones, el conocimiento sobre su despliegue y administración sigue siendo en gran parte propiedad de empresas privadas, limitando así el acceso de otras instituciones, como las académicas, a una comprensión más técnica y profunda de estas tecnologías.

En este contexto, resulta relevante que una institución académica no solo utilice, sino también investigue y replique el despliegue de plataformas de *cloud computing*. Además de contribuir al conocimiento académico, esta iniciativa responde a una necesidad interna de optimización de recursos. El Departamento de Electrónica de la Universidad del Valle de Guatemala cuenta con una computadora de alto rendimiento (HPC), cuya capacidad actual se encuentra subutilizada. La implementación de una plataforma de nube privada permitiría aprovechar este recurso para brindar servicios virtualizados, lo cual impulsaría la eficiencia en el uso de la infraestructura y, al mismo tiempo, ofrecería un entorno educativo de alto valor para estudiantes y personal de la universidad.

Este proyecto se propone como un ejercicio práctico y teórico para abordar dicha problemática. Su objetivo principal es implementar y documentar un despliegue modular de la plataforma OpenStack en la red de laboratorios del Departamento de Electrónica, empleando tanto infraestructura física como virtual. La solución habilitará la creación de máquinas virtuales y otros servicios en la HPC, facilitando un aprovechamiento más completo de sus capacidades y promoviendo el uso de tecnologías de nube dentro del ámbito académico.

CAPÍTULO 2

Antecedentes

Siendo OpenStack un software ampliamente utilizado, existen muchos casos exitosos de su implementación en una gran cantidad de sectores diferentes como lo son las finanzas, *cloud hosting*, telecomunicaciones, tecnologías de la información, entre otras. Debido a su versatilidad y modularidad, cada sector y empresa le da un enfoque distinto, lo cual propicia que exista una amplia variedad de casos de uso distintos.

De entre los sectores que usan de manera cotidiana estas herramientas, se pueden mencionar como los más relevantes para el propósito de este proyecto los ambientes académicos y de investigación; al igual que los despliegues realizados por empresas dedicadas a telecomunicaciones y *cloud hosting*.

Entre los amientes académicos y de investigación el caso de la Organización Europea para la Investigación Nuclear (mejor conocida como CERN) es el más relevante como guía. El CERN hace uso de los servicios de OpenStack desde 2013 [1], y utiliza máquinas proveedoras de recursos de computación con CentOS 7 para dotar de recursos virtuales a toda su red. El CERN empezó implementando servicios de virtualización de máquinas y tecnologías de volúmenes complementarios a estas máquinas y paulatinamente con el paso de los años han implementado nuevos módulos y funcionalidades hasta llegar a abarcar servicios más complejos como orquestaciones de contenedores, *software defined network*s, y una importante mejora en las capacidades de automatización de procesos y *workflows*. Para su implementación hacen uso de los módulos: Horizon, Nova, Cinder, Glance, Keystone y Neutron, cuyo uso también se contempla en la realización de este proyecto.

Otros casos valiosos enfocados al área de telecomunicaciones incluyen a empresas líderes del sector como AT&T, T-Mobile, Verizon, Deutsche Telekom y China Mobile. China Mobile es un caso particularmente valioso debido a que son la empresa de telecomunicaciones con más suscriptores del mundo, y hace un uso profundo de los servicios de OpenStack en cargas de trabajo como el despliegue de tecnologías 5G y junto a Ericsson están construyendo la mayor virtualización de funciones de red (NFV) basada en OpenStack del mundo [2].

CAPÍTULO 3

Justificación

A lo largo de los últimos años se ha dado una rápida adopción de tecnologías basadas en la nube, cerca del 94 % de las organizaciones en los Estados Unidos hacen uso de algún tipo de *cloud computing* y se prevee que el crecimiento de este mercado alcance un 360 % entre 2023 y 2030 [3]. La demanda de este tipo de servicios hace una necesidad el aprendizaje e investigación de su campo.

OpenStack es el *open source* software de *cloud computing* más utilizado en el mundo, por lo que es un candidato ideal para explorar las posibilidades que ofrece el *cloud computing* en un ambiente académico, gratuito y bien documentado.

Sumado a todo esto, el Departamento de Electrónica de la UVG cuenta con una computadora de alto rendimiento (HPC) que es utilizada para la realización de simulaciones y cálculos complejos, sin embargo, no es utilizada bajo una demanda constante. Esta computadora es un recurso valioso que no se aprovecha en su máxima capacidad, y por lo tanto, su transformación en un *cloud* privado sería un paso lógico para mejorar la eficiencia del manejo de recursos de la universidad al abrir la posibilidad de virtualizar una gran cantidad de funcionalidades adicionales capaces de coexistir en un mismo hardware, sin reemplazar sus capacidades actuales.

CAPÍTULO 4

Objetivos

4.1. Objetivo General

Desplegar empleando dos *hosts* —uno físico y otro virtualizado— la plataforma de *cloud computing* OpenStack, con las capacidades necesarias para ejecutar instancias de máquinas virtuales precargadas y manejar los recursos básicos que tienen asignados.

4.2. Objetivos Específicos

- Documentar todo el proceso de instalación para facilitar la replicación completa de la plataforma.
- Implementar y configurar el OpenStack *identity service* Keystone.
- Implementar y configurar el OpenStack *image service* Glance.
- Implementar y configurar el OpenStack *placement service* Placement.
- Implementar y configurar el OpenStack *networking service* Neutron.
- Implementar y configurar el OpenStack *compute service* Nova.
- Implementar y configurar el OpenStack *dashboard service* Horizon.
- Implementar y configurar el OpenStack *block storage service* Cinder.

CAPÍTULO 5

Alcance

El proyecto consiste en desplegar la plataforma de *cloud computing* OpenStack en la red de laboratorios del Departamento de Electrónica de la Universidad del Valle de Guatemala entre las fechas de enero a noviembre del año 2024. Este trabajo hace uso de la computadora de alto rendimiento (HPC) del departamento como principal nodo sobre el cual recae la demanda de recursos de la plataforma, y se emplea una máquina virtual alojada en otra computadora de la misma red para el despliegue de un nodo de control. El despliegue de OpenStack es de forma modular, por lo que es necesario instalar cada uno de los servicios que lo integran de manera individual, sin hacer uso de *frameworks* para el proceso de instalación; los servicios instalados son: Keystone, Glance, Placement, Nova, Neutron, Cinder y Horizon. Con estos componentes OpenStack es capaz de proveer infraestructura como servicio (IaaS) a los usuarios de la Universidad, permitiendo la creación de máquinas virtuales y un mejor aprovechamiento de los recursos de la HPC. Además, como entregable del proyecto se presenta una guía completa del proceso de instalación y configuración de OpenStack.

CAPÍTULO 6

Marco teórico

Cloud computing

Definición

La computación en la nube, es un modelo que permite el acceso bajo demanda a un conjunto compartido de recursos informáticos configurables (como redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser provisionados y liberados rápidamente con un esfuerzo de gestión mínimo o interacción del proveedor de servicios. En términos simples, permite a los usuarios utilizar recursos de TI a través de internet, pagando solo por lo que consumen, en lugar de tener que poseer y gestionar la infraestructura física [4]. Este modelo ofrece varias ventajas, como la escalabilidad, flexibilidad y eficiencia de costos. Los usuarios pueden acceder a los recursos desde cualquier lugar y en cualquier momento, siempre que tengan una conexión a la red. Además, los proveedores de servicios en la nube suelen encargarse del mantenimiento y la seguridad de la infraestructura subyacente, permitiendo a las organizaciones centrarse en sus competencias principales.

Modelos de servicio

Cloud computing se estructura en tres modelos principales de servicio [5] [6], cada uno sirviendo propósitos distintos:

- Infraestructura como Servicio (IaaS): IaaS proporciona recursos de computación virtualizados a través de internet. Los usuarios pueden alquilar servidores, almacenamiento y redes, y tienen control sobre el sistema operativo y las aplicaciones instaladas.
- Plataforma como Servicio (PaaS): PaaS ofrece una plataforma sobre la cual los desarrolladores pueden construir, desplegar y gestionar aplicaciones sin preocuparse por la infraestructura subyacente. Este modelo incluye sistemas operativos, entornos de desarrollo, bases de datos y servidores web.

- Software como Servicio (SaaS): SaaS entrega aplicaciones a través de internet. Los usuarios acceden a las aplicaciones mediante un navegador web, y no necesitan gestionar la infraestructura subyacente ni preocuparse por las actualizaciones de software.

Modelos de implementación

Los modelos de implementación de cloud computing se dividen en cuatro categorías principales [7], cada una con diferentes niveles de control, flexibilidad y costo:

- Nube pública: los recursos de computación son propiedad y están operados por un proveedor de servicios en la nube, que ofrece estos recursos al público en general a través de internet. Los usuarios comparten la infraestructura con otros clientes, lo que puede reducir los costos.
- Nube privada: está dedicada a una sola organización. Puede ser gestionada internamente o por un tercero, y puede estar ubicada en las instalaciones de la organización o en un centro de datos externo. Este modelo ofrece mayor control y seguridad, pero puede ser más costoso y complejo de gestionar. Las nubes privadas son ideales para organizaciones con estrictos requisitos de seguridad y cumplimiento.
- Nube híbrida: combina las nubes públicas y privadas, permitiendo que los datos y aplicaciones se compartan entre ellas. Este modelo proporciona mayor flexibilidad y opciones de despliegue, permitiendo a las organizaciones escalar sus recursos según sea necesario y mantener la seguridad y control sobre sus datos sensibles. Las nubes híbridas son útiles para equilibrar la carga de trabajo y optimizar costos.
- Nube comunitaria: la infraestructura es compartida por varias organizaciones con intereses o requisitos comunes, como seguridad, cumplimiento o políticas específicas. Este modelo puede ser gestionado por las organizaciones participantes o por un proveedor de servicios.

Estos modelos de implementación permiten a las organizaciones elegir la combinación de recursos de computación que mejor se adapte a sus necesidades específicas, optimizando costos y mejorando la eficiencia operativa [8].

Segmentación de redes en la Nube

Los proveedores de servicios de cloud computing atienden a miles de clientes que requieren servicios de diversa índole, para gestionar las redes de cada cliente de manera efectiva, las plataformas deben poder proporcionar un alto grado de independencia entre ellas, garantizando la seguridad y facilitando la administración de cada red [9]. Para lograr esto, se utiliza una variedad de protocolos e implementaciones distintas. Entre las implementaciones de red más comunes se pueden encontrar:

- Flat networks: permiten que todas las instancias residan en la misma red, la cual puede ser compartida con los hosts. En este modelo, no se utilizan etiquetas de protocolos

adicionales, ni ningún otro tipo de segregación de red. Las redes planas son útiles para configuraciones simples donde no se necesita aislamiento de red entre instancias o entre proyectos. Este tipo de red facilita la conexión directa de las instancias a la red externa, lo que puede ser ventajoso en entornos donde la simplicidad y la conectividad directa son prioritarias.

- VLAN networks: Las redes VLAN (Virtual Local Area Networks) permiten la creación de múltiples redes de mediante el uso de identificadores VLAN (802.1Q). Este protocolo permite el aislamiento de las instancias en diferentes segmentos de red —hasta 4096— dentro de la misma infraestructura física de capa 2, por medio de etiquetas que se añaden al encabezado ethernet, facilitando la comunicación entre ellas sin interferir con otras redes. Las redes VLAN son ideales para entornos donde se requiere una separación clara y segura del tráfico de red, así como la posibilidad de integración con infraestructuras de red físicas ya existentes [10].
- VXLAN networks: Las redes VXLAN (Virtual Extensible LAN) utilizan superposiciones de red para proporcionar comunicación privada entre instancias en una manera muy similar a las VLAN. A diferencia de las VLAN, VXLAN permite la creación de redes lógicas escalables en infraestructuras de nube —hasta 16 millones— y hacen uso tanto de la capa 2 como de la capa 3; el protocolo VXLAN se desarrolló con el propósito de superar las limitaciones de VLAN entornos de nube a gran escala. [11].

Virtualización

Hipervisores

Un hipervisor, también conocido como monitor de máquina virtual (VMM), es un software fundamental en la tecnología de virtualización que crea y gestiona máquinas virtuales (VM). Este software actúa como una capa de abstracción entre el hardware físico del servidor y las máquinas virtuales, permitiendo que múltiples sistemas operativos y aplicaciones se ejecuten simultáneamente en un solo hardware físico [12]. Los hipervisores se dividen en dos categorías principales:

- Tipo 1 (*bare-metal*): se ejecutan directamente sobre el hardware físico del servidor sin necesidad de un sistema operativo anfitrión. Este tipo de hipervisor es conocido por su alto rendimiento y eficiencia, ya que puede acceder directamente a los recursos de hardware. Estos hipervisores son comúnmente utilizados en entornos de centros de datos y servidores empresariales debido a su capacidad para manejar grandes cargas de trabajo y ofrecer alta disponibilidad [13].
- Tipo 2 (*hosted*): se ejecutan sobre un sistema operativo anfitrión que proporciona servicios de gestión de recursos y soporte para las VM. Aunque son más fáciles de instalar y usar, tienden a ser menos eficientes en términos de rendimiento en comparación con los hipervisores de tipo 1 debido a la capa adicional de software entre el hardware y las VM. Estos hipervisores son más comunes en entornos de desarrollo y pruebas, donde la facilidad de uso y la flexibilidad son más importantes que el rendimiento puro [13].

Los hipervisores permiten la consolidación de servidores, lo que reduce la necesidad de hardware físico adicional y mejora la utilización de los recursos existentes. Además, proporcionan una infraestructura más flexible y escalable, permitiendo a las organizaciones responder rápidamente a las demandas cambiantes de recursos y cargas de trabajo.

Máquinas virtuales

Una máquina virtual (VM) es una emulación de un sistema informático real que proporciona la funcionalidad de una computadora física [14]. Las VM se ejecutan sobre un hipervisor y comparten los recursos físicos del servidor, como CPU, memoria, almacenamiento y interfaces de red, mientras permanecen aisladas unas de otras. Las características clave de las máquinas virtuales incluyen:

- Aislamiento: cada VM opera de forma independiente y aislada de las demás, lo que significa que los problemas en una VM no afectan a las otras. Esto es crucial para la seguridad y la estabilidad, ya que permite que múltiples entornos operen en el mismo hardware sin interferencias.
- Flexibilidad: las VM pueden ejecutarse en cualquier hardware compatible con el hipervisor, lo que facilita la migración y el despliegue de aplicaciones. Esta portabilidad permite mover aplicaciones entre diferentes servidores y centros de datos sin necesidad de modificaciones significativas.
- Escalabilidad: las VM permiten la creación y eliminación rápida de instancias según sea necesario, facilitando la adaptación a las demandas cambiantes de carga de trabajo. Esta capacidad de escalar hacia arriba o hacia abajo según la demanda es esencial para optimizar el uso de recursos y responder a las fluctuaciones del mercado o de la carga de trabajo.
- Seguridad: el aislamiento entre VM ayuda a mantener la seguridad, ya que cada VM puede tener su propio sistema operativo y aplicaciones separados de otros entornos. Esto significa que una brecha de seguridad en una VM no afecta a las demás, proporcionando un nivel adicional de seguridad en entornos multiusuario.

Además, las VM pueden utilizarse para una variedad de propósitos, incluyendo el desarrollo y prueba de software, la ejecución de aplicaciones empresariales y la consolidación de servidores. Al utilizar VM, las organizaciones pueden reducir los costos de hardware, mejorar la eficiencia operativa y proporcionar un entorno más flexible y seguro para sus aplicaciones y datos.

OpenStack

Funcionamiento general

El proyecto OpenStack es una plataforma de computación en la nube de código abierto para todo tipo de nubes, que tiene como objetivo ser simple de implementar, altamente escalable y rica en funcionalidades. [15] OpenStack proporciona una solución de Infraestructura

como Servicio (IaaS) a través de un conjunto de servicios interrelacionados. Cada servicio ofrece una interfaz de programación de aplicaciones (API) que facilita su integración. Dependiendo de las necesidades, se pueden instalar algunos o todos los servicios. El sistema OpenStack consta de varios servicios clave que se instalan por separado. Estos servicios trabajan juntos según las necesidades de cada nube Se puede instalar cualquiera de estos proyectos por separado y configurarlos de forma independiente o como entidades conectadas. OpenStack es una plataforma versátil y adaptable que proporciona una base sólida para construir y gestionar diversos tipos de infraestructura en la nube.

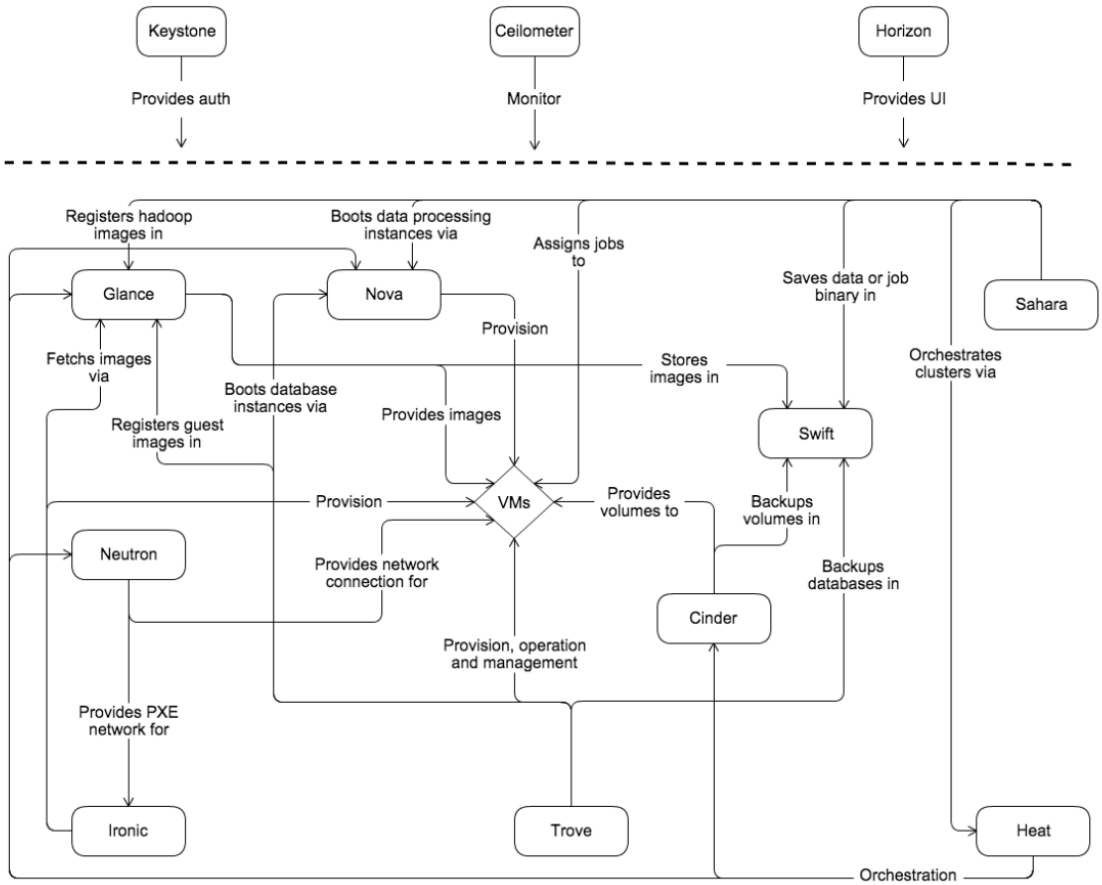

Figura 1: Diagrama que describe la relación entre los servicios de OpenStack

Módulos

- **Keystone**: es el servicio de identidad de OpenStack, el primero con el que interactúan los usuarios. Permite la autenticación y autorización de usuarios, que luego pueden acceder a otros servicios de OpenStack. También puede integrarse con sistemas externos de gestión de usuarios. [16] Gestiona el catálogo de servicios, una colección de todos los servicios disponibles en un despliegue de OpenStack, cada uno con múltiples *endpoints*: admin, interno y público. Estos endpoints pueden estar en redes separadas por razones de seguridad. Los componentes de Keystone son:
  - Servidor: proporciona servicios de autenticación y autorización mediante una in-

terfaz RESTful.

- Drivers: permiten acceder a la información de identidad en repositorios externos, como bases de datos SQL o servidores LDAP.
- Módulos: *middleware* que intercepta solicitudes de servicio, extrae credenciales de usuario y las envía al servidor para autorización, utilizando la WSGI.

- **Glance**: es el servicio de imágenes de OpenStack que permite a los usuarios descubrir, registrar y recuperar imágenes de máquinas virtuales a través de una API REST [17]. Las imágenes se pueden almacenar en diversos lugares, desde sistemas de archivos hasta sistemas de almacenamiento de objetos. Este módulo es crucial para la Infraestructura como Servicio (IaaS) en OpenStack. Los componentes de Glance son:
  - Glance-api: maneja llamadas de la API para el descubrimiento, recuperación y almacenamiento de imágenes.
  - Base de Datos: almacena los metadatos de las imágenes.
  - Repositorio de Imágenes: soporta varios tipos de almacenamiento como sistemas de archivos, almacenamiento de objetos y más.
  - Servicio de Metadatos: permite definir metadatos personalizados para recursos.
- **Placement**: permite a otros proyectos de OpenStack rastrear sus propios recursos. Estos proyectos pueden registrar o eliminar sus recursos a través de la API HTTP de Placement. Placement se originó en el proyecto Nova y su funcionalidad fue diseñada para ser genérica, de modo que cualquier servicio que necesite gestionar la selección y consumo de recursos pueda utilizarlo [18]. Usos que Nova hace de Placement incluyen:
  - Nova-compute: utiliza el rastreador de recursos para crear registros de proveedores de recursos, establecer inventarios de recursos disponibles (como VCPU) y establecer características cualitativas (como STORAGE_DISK_SSD).
  - Nova-scheduler: selecciona los hosts de destino adecuados para una carga de trabajo, formulando solicitudes a Placement para obtener una lista de candidatos de asignación según requisitos cuantitativos y cualitativos. Luego, asigna recursos a un proveedor de recursos específico, consumiendo parte del inventario establecido.
- **Neutron**: es el proyecto de OpenStack que proporciona conectividad de red como servicio entre dispositivos de interfaz (como vNICs) gestionados por otros servicios de OpenStack (como Nova). Neutron interactúa principalmente con Nova para proporcionar redes y conectividad a las instancias de cómputo [19]. Componentes de Neutron:
  - Neutron-server: acepta y enruta las solicitudes API a los complementos de red correspondientes.
  - Complementos y agentes de red: conectan/desconectan puertos, crean redes/subredes y asignan direcciones IP. Soporta varios complementos y agentes para tecnologías como Cisco, Open vSwitch, y VMware NSX.
  - Cola de mensajes: enruta información entre Neutron-server y agentes, y actúa como base de datos para el estado de la red.
- **Nova**: es el proyecto de OpenStack que permite aprovisionar instancias de cómputo (servidores virtuales). Soporta la creación de máquinas virtuales, servidores *bare-metal*

(a través de Ironic) y tiene soporte limitado para contenedores de sistema. Nova se ejecuta como un conjunto de *daemons* sobre servidores Linux existentes [20].

- **Horizon**: es la implementación canónica del Panel de Control de OpenStack, que proporciona una interfaz de usuario basada en web para los servicios de OpenStack. Permite la creación y gestión de proyectos, usuarios y roles; proporciona acceso a servicios como Nova (computación), Neutron (redes), Cinder (almacenamiento en bloque) y Glance (imágenes); y permite monitorizar el estado y rendimiento de los recursos, así como ejecutar tareas administrativas [21].
- **Cinder**: es el servicio de almacenamiento en bloque de OpenStack que proporciona volúmenes a máquinas virtuales de Nova, hosts *bare-metal* de Ironic, contenedores y más. Cinder cumple un papel fundamental en la infraestructura de OpenStack al proporcionar un almacenamiento en bloque confiable y escalable para diversas cargas de trabajo [22]. Sus objetivos principales son:
  - Arquitectura basada en componentes: Permite agregar rápidamente nuevos comportamientos al sistema.
  - Alta disponibilidad: Escala para cargas de trabajo muy exigentes.
  - Tolerancia a fallos: Procesos aislados evitan fallos en cascada.
  - Recuperable: Los fallos deben ser fáciles de diagnosticar, depurar y corregir.
  - Estándares abiertos: Ser una implementación de referencia para una API impulsada por la comunidad.

Arquitectura

Una arquitectura de OpenStack se refiere a la disposición y organización de los nodos y servicios que forman parte de una implementación de OpenStack. Esta arquitectura es fundamental para la creación y gestión eficiente de recursos en la nube. A menudo, una arquitectura básica de OpenStack comprende nodos controladores y nodos de cómputo, cada uno con funciones específicas y servicios asociados [15].

- Nodo controlador: es responsable de gestionar la infraestructura de OpenStack y ejecutar varios servicios esenciales, como el *identity service*, el *image service* y el *placement service*. También gestiona partes críticas del *compute service*, además de ejecutar diversos agentes del *networking service* para garantizar la conectividad. El *dashboard* también se encuentra alojado en este nodo, proporcionando una interfaz gráfica para la gestión de recursos. Además de estos servicios principales, el nodo controlador requiere servicios de soporte, como una base de datos SQL, una cola de mensajes y un servidor de tiempo. Dependiendo de los requisitos del entorno, el nodo controlador puede opcionalmente ejecutar partes de otros servicios como orquestación y telemetría. El nodo controlador es el núcleo de la infraestructura de OpenStack y requiere un mínimo de dos interfaces de red.
- Nodo de cómputo: se centra en ejecutar las instancias de máquinas virtuales y proporcionar recursos de cómputo para las cargas de trabajo. Este nodo ejecuta un hipervisor,

como KVM de forma predeterminada, que opera las instancias de máquinas virtuales. Además, el nodo de cómputo ejecuta un agente de servicio de Networking, que se encarga de conectar las instancias a las redes virtuales y proporcionar servicios de *firewall* a través de grupos de seguridad. Se puede implementar más de un nodo de cómputo según las necesidades de la carga de trabajo. Cada nodo de cómputo necesita un mínimo de dos interfaces de red para garantizar la conectividad y el rendimiento de las instancias virtuales.

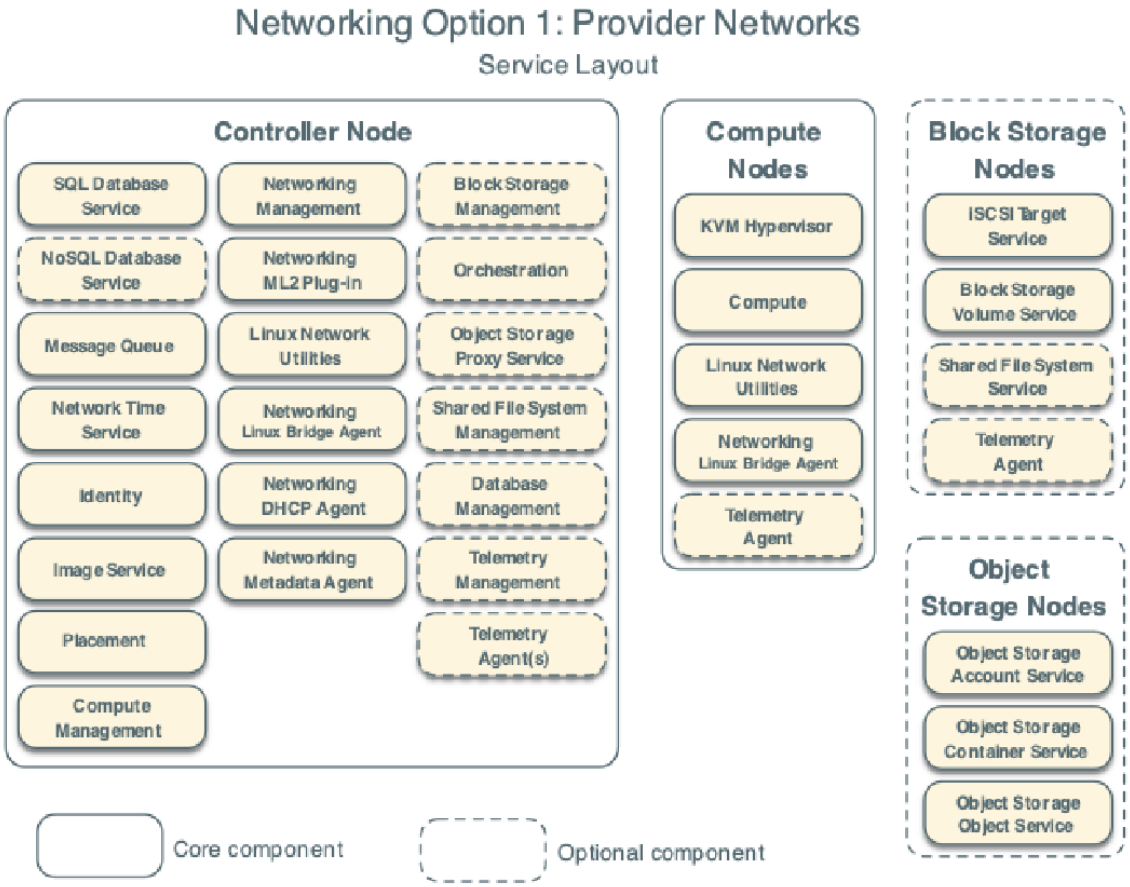

Figura 2: Diagrama que describe una arquitectura mínima de OpenStack

CAPÍTULO 7

Guía de instalación de OpenStack

Este despliegue de OpenStack fue diseñado como una prueba de concepto para explorar la implementación de la plataforma. Dado que el entorno está alojado en una red privada segura y protegida por un firewall estricto, se omitieron algunas características de seguridad propias de un entorno de nube de producción, como el manejo avanzado de contraseñas. En este despliegue, todas las contraseñas utilizadas durante el proceso de instalación fueron unificadas con el valor: "onap". Esto significa que, independientemente de la contraseña mencionada en la guía, en esta configuración siempre se reemplazó por "onap". A lo largo de la guía, se resalta con color `rosado` múltiples instancias donde aparecen contraseñas para resaltarlas. Es importante entender los diferentes tipos de contraseñas que aparecen en OpenStack para evitar confusiones:

- Contraseñas de bases de datos de los servicios de OpenStack: Estas contraseñas están asociadas a las bases de datos creadas para cada servicio. Siguen el formato `nombre_DBPASS`. Por ejemplo, `GLANCE_DBPASS`. Estas claves se definen al momento de otorgar permisos a las bases de datos con el comando `GRANT ALL PRIVILEGES ON`.
- Contraseñas de usuarios de los servicios de OpenStack: Estas contraseñas se utilizan para autenticar usuarios creados en OpenStack y siguen el formato `nombre_PASS`. Por ejemplo, `GLANCE_PASS`. Estas claves se asignan durante la creación de usuarios con el comando `openstack user create`.
- Otras contraseñas: Estas están relacionadas con aplicaciones que proveen servicios a los componentes de OpenStack. Su formato puede variar. Por ejemplo, la contraseña de RabbitMQ se define como `PASSWORD_PLACEHOLDER` durante la configuración inicial, pero también puede referenciarse como `RABBIT_PASS` en pasos posteriores de la guía.

7.1. Establecer ambiente básico de instalación

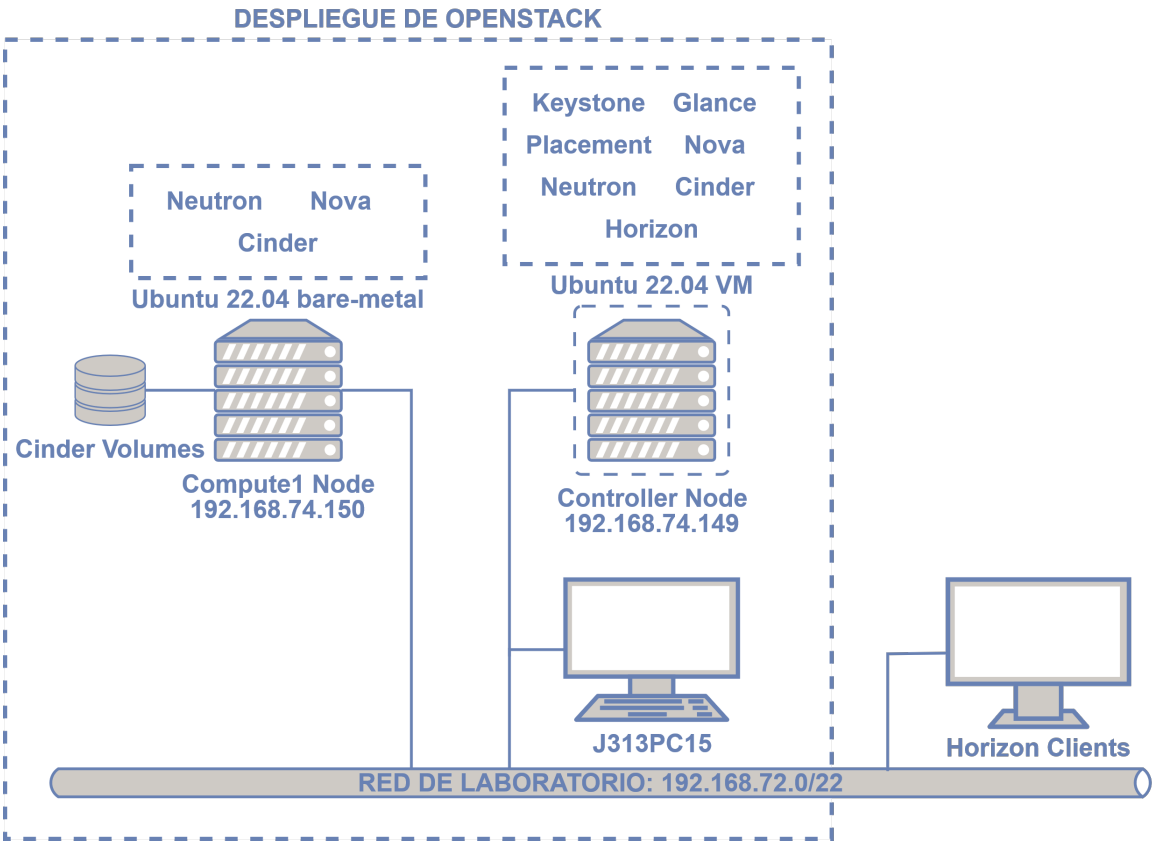

Figura 3: Diagrama que describe la topología del despliegue de OpenStack configurado

7.1.1. Instalación de sistemas operativos

OpenStack es una plataforma que opera de manera distribuida, haciendo uso de varios sistemas que se coordinan para realizar sus operaciones. Los equipos que alojan los diferentes servicios de OpenStack se denominan "nodo", y cada tipo de nodo cumple funciones específicas dentro de la arquitectura de la plataforma. Existen diferentes configuraciones para desplegar OpenStack, dependiendo de los requisitos de funcionalidad, rendimiento, disponibilidad y escalabilidad. En este despliegue, se han utilizado los siguientes nodos:

- *Controller Node*: Este nodo administra y coordina los servicios principales de OpenStack, como la autenticación, la gestión de imágenes, la programación de recursos y la red. Además, aloja las bases de datos y maneja las colas de mensajería que utilizan los otros nodos para comunicarse. Para este caso está implementado como una máquina virtual, virtualizada mediante Oracle VirtualBox y alojada en el equipo identificado como "J313PC1".
  - *Hostname*: openstackcontroller
  - *Username*: openstackcontroller

    - *Password*: onap

- *Compute Node*: Este nodo se encarga de la virtualización y la ejecución de las máquinas virtuales (instancias). También gestiona los recursos de cómputo, incluyendo CPU, memoria y almacenamiento de instancia temporal. Este nodo es una máquina física (*bare-metal*), está instalado en uno de los SSD de la HPC ubicada en el salón CIT118A.

    - *Hostname*: openstackcompute1
    - *Username*: onap
    - *Password*: onap

- *Storage Node*: Este nodo almacena y gestiona los volúmenes de almacenamiento persistente que se pueden asociar a las instancias. Aunque puede operar de manera completamente independiente, para este caso se implementaron los servicios de un *storage node* en el mismo hardware que el *compute node*.

OpenStack cuenta con soporte oficial para varias distribuciones de Linux, entre ellas SUSE, RHEL, CentOS y Ubuntu. En este despliegue se ha optado por Ubuntu Server 22.04 como sistema operativo para todos los nodos, ya que es una de las opciones más populares y ampliamente documentadas para implementar OpenStack. Esta versión LTS de Ubuntu no solo proporciona estabilidad, sino que además cada lanzamiento LTS de Ubuntu se alinea con una versión de OpenStack de compatibilidad completa, con paquetes optimizados y soporte de largo plazo. En el caso de Ubuntu 22.04, la versión de OpenStack correspondiente es Yoga, que será la utilizada en esta implementación, garantizando así una integración fluida y eficiente de los componentes de la plataforma.

Instalación en el *Controller Node*

Dado que el nodo controlador es una máquina virtual, el primer paso es configurar la interfaz de red de la máquina en modo *bridge*, para que se conecte de manera directa a la red de laboratorios del Departamento y configurar el `promiscuous mode = allow all`, para garantizar un tráfico fluido de paquetes. El siguiente paso es dejar de manera estática la IP de la máquina, ya que es necesario para las configuraciones posteriores de OpenStack que las IPs de todos sus nodos no varíen.

```
$ cd /etc/netplan/
$ sudo nano 00-installer-config.yaml
```

Cuadro 1: Apertura de archivo de configuración de red

```
network:
    ethernets:
        enp0s3:
        dhcp4: no
        addresses:
            - 192.168.74.149/22
        nameservers:
            addresses: [192.168.8.205,8.8.8.8]
        routes:
            - to: default
              via: 192.168.72.1
    version: 2
```

Cuadro 2: Modificación de archivo de configuración de red

```
$ sudo netplan apply
```

Cuadro 3: Aplicar la configuración del archivo de red

Instalación en el *Compute1 Node*

De la misma forma, también hay que configurar una IP estática para el nodo de cómputo.

```
$ cd /etc/netplan/
$ sudo nano 00-installer-config.yaml
```

Cuadro 4: Apertura de archivo de configuración de red

```
network:
    ethernets:
        enp0s3:
        dhcp4: no
        addresses:
            - 192.168.74.150/22
        nameservers:
            addresses: [192.168.8.205,8.8.8.8]
        routes:
            - to: default
            via: 192.168.72.1
        version: 2
```

Cuadro 5: Modificación de archivo de configuración de red

```
$ sudo netplan apply
```

Cuadro 6: Aplicar la configuración del archivo de red

Instalación y verificación de funcionalidad en ambos nodos

Modificar los *hostnames* para que ambos nodos resuelvan de manera adecuada los términos clave `controller` y `compute1` a sus respectivas IPs.

```
$ sudo nano /etc/hosts
```

Cuadro 7: Apertura de archivo de configuración de *hosts*

```
# Add definitions
127.0.0.1 localhost
192.168.74.149 controller
192.168.74.150 compute1
```

Cuadro 8: Modificación de archivo de configuración de *hosts*

Verificar conectividad entre los nodos y hacia internet desde ambos *hosts*.

```
$ ping controller
$ ping compute1
$ ping docs.openstack.org
```

Cuadro 9: Pings para la verificación de conectividad

7.1.2. Instalación de NTP

El protocolo NTP (*Network Time Protocol*) permite sincronizar la hora de los sistemas de una red de manera precisa y confiable. Su función es ajustar el reloj de los dispositivos de la red para que todos mantengan una hora uniforme, algo importante en entornos distribuidos. NTP utiliza una jerarquía de niveles de tiempo conocidos como "estratos", en los cuales los servidores de nivel más bajo, como los estrato 0 (relojes de referencia), proporcionan la hora a servidores de niveles superiores (estratos 1, 2, etc.). Los dispositivos cliente pueden comunicarse con servidores NTP para ajustar sus relojes en función de un tiempo universal coordinado (UTC), usando algoritmos para compensar cualquier variación de latencia en la red y mejorar la precisión.

En OpenStack NTP es fundamental para mantener la consistencia y sincronización en todos los nodos de la infraestructura. Los servicios de OpenStack, dependen de una hora precisa para coordinar sus operaciones y mantener la integridad de los registros, auditorías y la coherencia de datos entre los diferentes nodos. Sincronizar el tiempo ayuda a prevenir problemas como los errores de autenticación y asegura la correcta ejecución de tareas programadas, balanceo de carga y otras actividades críticas en una nube distribuida.

La instalación de protocolo NTP usa al *controller node* como servidor local para los

demás nodos desplegados, y que lo utilicen como referencia.

Instalación en el *Controller Node*

Modificar el horario del sistema a Guatemala.

```
$ sudo timedatectl set-timezone America/Guatemala
```

Cuadro 10: Modificación de zona horaria del sistema

Instalar y configurar **Chrony**. Modificar los servidores que se usarán para sincronizar el reloj del sistema por un conjunto de servidores en el mismo huso horario que Guatemala (actualmente Guatemala no tiene servidores de NTP activos), para este caso, se eligen servidores de Costa Rica; es posible encontrar los servidores en la página: ntppool. También permitir que cualquier dispositivo perteneciente a la sub red de la red de laboratorios del Departamento pueda utilizar al *controller node* como un servidor NTP local con la opción `allow`.

```
$ sudo apt install chrony
$ sudo nano /etc/chrony/chrony.conf
```

Cuadro 11: Instalar y abrir archivo de configuración de Chrony

```
# Replace all other server/pool options
server 3.cr.pool.ntp.org iburst
server 0.north-america.pool.ntp.org iburst
server 3.north-america.pool.ntp.org iburst

# Add at the end
allow 192.168.72.0/22
```

Cuadro 12: Modificación de archivo de configuración de Chrony

```
$ sudo service chrony restart
```

Cuadro 13: Reiniciar servicio de Chrony

Instalación en el *Compute1 Node*

Modificar el horario del sistema a Guatemala.

```
$ sudo timedatectl set-timezone America/Guatemala
```

Cuadro 14: Modificación de zona horaria del sistema

Instalar y configurar Chrony. Modificar los servidores que se usarán para sincronizar el reloj del sistema por el controller node, usándolo como referencia de `server`.

```
$ sudo apt install chrony
$ sudo nano /etc/chrony/chrony.conf
```

Cuadro 15: Instalar y abrir archivo de configuración de Chrony

```
# Comment all "pool/server ..." lines, replace with controller
server controller iburst
```

Cuadro 16: Modificación de archivo de configuración de Chrony

```
$ sudo service chrony restart
```

Cuadro 17: Reiniciar servicio de Chrony

Verificación de funcionalidad en ambos nodos

Al ejecutar el siguiente comando desde el *controller node*, se deben referenciar los servidores de NTP, mientras que desde el *compute1 node* se referencia al *controller node*.

```
$ chronyc sources
```

Cuadro 18: Listar fuentes de reloj de Chrony

```
openstackcontroller@openstackcontroller:~$ chronyc sources
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^? ntp3.fratec.net               0   7     0     -     +0ns[   +0ns]
    +/-    0ns
^- 64.ip-54-39-23.net            3   6    17    62  -2114us[-2114us]
    +/-   34ms
^* vps5.drown.org                2   6    17    63    -64us[ -503us]
    +/-   33ms
```

Cuadro 19: Mostrar fuentes de reloj de chrony en *controller node*

```
onap@openstackcompute1:~$ chronyc sources
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* controller                    3   7   377   120    +11us[  +12us]
    +/-   47ms
```

Cuadro 20: Mostrar fuentes de reloj de Chrony en *compute1 node*

7.1.3. Instalación de OpenStack Packages

Instalación de paquetes básicos necesarios para el despliegue de OpenStack (no incluye servicios).

Instalación en ambos nodos

Actualizar todos los paquetes en ambos nodos y desactivar actualizaciones automáticas. Paso necesario para que posibles actualizaciones no afecten el funcionamiento de OpenStack una vez sea desplegado.

```
# Actualización de paquetes
$ sudo apt update
$ sudo apt upgrade
# Suspensión de actualizaciones
$ sudo systemctl stop unattended-upgrades
$ sudo systemctl disable unattended-upgrades
# Revisar que no hayan quedado actualizaciones pendientes
$ sudo apt update
$ sudo apt list --upgradable
```

Cuadro 21: Comandos para manejo de paquetes en el sistema

Gracias a que se está utilizando Ubuntu 22.04, y la versión de OpenStack por usar es Yoga; no es necesario hacer una instalación adicional de paquetes, pues ya están instalados por defecto.

```
# Dado que se está usando Ubuntu 22.04 LTS por defecto se contienen
    los paquetes del release Yoga de OpenStack
$ sudo apt install python3-openstackclient
```

Cuadro 22: Instalación de paquetes adicionales de OpenStack

7.1.4. Instalación de MariaDB

MariaDB es un sistema de gestión de bases de datos relacional, derivado de MySQL. Utiliza el lenguaje SQL (*Structured Query Language*) para realizar operaciones de consulta, inserción, modificación y eliminación de datos en sus tablas. MariaDB es compatible con una gran cantidad de motores de almacenamiento, lo que permite ajustar su rendimiento y características a las necesidades específicas de las aplicaciones. En OpenStack, MariaDB se emplea como la base de datos central para almacenar la información de configuración y estado de sus servicios principales. Todos estos servicios necesitan registrar información crítica y acceder a ella de manera eficiente para coordinar sus funciones en un entorno distribuido.

Instalación en el *Controller Node*

Instalación de los paquetes de MariaDB

```
$ sudo apt install mariadb-server python3-pymysql
```

Cuadro 23: Instalar paquetes para MariaDB

Crear archivo de configuración para MariaDB y modificar su contenido. Es importante ajustar el parámetro `bind-address` para que tenga la IP del nodo controlador.

```
$ cd /etc/mysql/mariadb.conf.d/
$ sudo nano 99-openstack.cnf
```

Cuadro 24: Crear archivo de configuración de MariaDB

```
[mysqld]
# ...
bind-address = 192.168.74.149
default-storage-engine = innodb
innodb_file_per_table = on
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
```

Cuadro 25: Modificación de archivo de configuración de MariaDB

Una vez finalizado el archivo de configuración, es necesario reiniciar el servicio de MySQL.

```
$ sudo service mysql restart
```

Cuadro 26: Reiniciar servicio de MariaDB

Luego ejecutar el siguiente comando, que es un script de seguridad que se utiliza para mejorar la configuración inicial de seguridad de una instalación de MariaDB. No es indispensable responder de la misma manera la configuración, no obstante, estas fueron las opciones elegidas.

```
$ sudo mysql_secure_installation
```

Cuadro 27: Instalar paquetes complementarios para MariaDB

```
Enter current password for root (enter for none): onap
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that
   nobody can log into the MariaDB root user without the proper
   authorisation.
You already have your root account protected, so you can safely
   answer 'n'.
Switch to unix_socket authentication [Y/n] n
... skipping.

You already have your root account protected, so you can safely
   answer 'n'.
Change the root password? [Y/n] n
... skipping.

By default, a MariaDB installation has an anonymous user, allowing
   anyone to log into MariaDB without having to have a user account
   created for them. This is intended only for testing, and to make
   the installation go a bit smoother. You should remove them before
    moving into a production environment.
Remove anonymous users? [Y/n] y
... Success!

Normally, root should only be allowed to connect from 'localhost'.
   This ensures that someone cannot guess at the root password from
   the network.
Disallow root login remotely? [Y/n] y
... Success!

By default, MariaDB comes with a database named 'test' that anyone
   can access. This is also intended only for testing, and should be
    removed before moving into a production environment.
Remove test database and access to it? [Y/n] y
Dropping test database...
Success!
Removing privileges on test database... Success!

Reloading the privilege tables will ensure that all changes made so
   far will take effect immediately.
Reload privilege tables now? [Y/n] y
... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MariaDB
   installation should now be secure.
Thanks for using MariaDB!
```

Cuadro 28: Configuración de seguridad de MariaDB

7.1.5. Instalación de RabbitMQ

Instalación de un servicio tipo *message queue* para la coordinación de operaciones e intercambiar información del estado de los servicios.

RabbitMQ es un sistema de mensajería de código abierto que permite la comunicación entre aplicaciones mediante el intercambio de mensajes. Basado en el protocolo AMQP (*Advanced Message Queuing Protocol*), RabbitMQ facilita que los diferentes servicios en una arquitectura distribuida intercambien datos de forma asincrónica. Este servicio actúa como un intermediario que recibe, almacena y distribuye mensajes a los servicios o aplicaciones que los necesitan, asegurando que los datos se entreguen de manera confiable, incluso cuando algunos componentes están temporalmente fuera de línea. En OpenStack, RabbitMQ juega un papel fundamental al facilitar la comunicación entre sus distintos servicios. Los servicios de OpenStack envían y reciben mensajes mediante RabbitMQ para coordinar acciones, actualizar estados y compartir información de manera continua y sin bloqueos. Al usar RabbitMQ, OpenStack asegura una comunicación eficiente y escalable entre sus componentes, lo que es crucial para el funcionamiento de la plataforma y para la orquestación de recursos en tiempo real.

Instalación en el *Controller Node*

Instalar RabbitMQ, luego añadir un usuario openstack con su contraseña respectiva (`PASSWORD_PLACEHOLDER = onap`) y brindarle permisos de lectura, escritura y configuración a este usuario.

```
$ sudo apt install rabbitmq-server
$ sudo rabbitmqctl add_user openstack PASSWORD_PLACEHOLDER
$ sudo rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

Cuadro 29: Instalación y configuración de RabbitMQ

7.1.6. Instalación de Memcached

Memcached es un sistema de almacenamiento en memoria distribuido que se utiliza para acelerar aplicaciones web al almacenar datos en la memoria RAM. Este servicio actúa como una caché en la que se almacenan temporalmente resultados de consultas o información que se solicita frecuentemente, reduciendo la carga en las bases de datos y mejorando el rendimiento general. Memcached utiliza un sistema de almacenamiento en *key-value*, lo que significa que los datos se almacenan en memoria con una clave única que facilita su acceso y recuperación de forma rápida. Este enfoque es ideal para mejorar la eficiencia de aplicaciones y servicios que requieren respuestas rápidas y procesamiento de grandes volúmenes de datos.

En OpenStack, Memcached se utiliza principalmente para acelerar las operaciones de autenticación y para almacenar en caché los *tokens* de usuario generados por el servicio Keystone. Cuando un usuario o servicio solicita acceso a OpenStack, Keystone genera un *token* que se almacena en Memcached, permitiendo que los servicios lo recuperen rápida-

mente sin necesidad de consultar la base de datos en cada petición. Este almacenamiento en caché de *tokens* es fundamental para mantener la escalabilidad y la velocidad en un entorno de nube distribuido, ya que reduce la latencia y evita cuellos de botella en la autenticación, mejorando la capacidad de respuesta y el rendimiento general de OpenStack.

Instalación en el *Controller Node*

Instalar los paquetes de Memcached. Luego en el archivo de configuración es necesario reemplazar el parámetro `-l 127.0.0.1` por `-l 192.168.74.149`, o dicho de otra forma, por la IP de gestión del `controller node`; este cambio hace que Memcached escuche de esta IP los *requests* entrantes. Finalmente reiniciar el servicio para que el cambio entre en efecto.

```
$ sudo apt install memcached python3-memcache
$ sudo nano /etc/memcached.conf
```

Cuadro 30: Instalar y abrir archivo de configuración de Memcached

```
# Replace -l 127.0.0.1 with the new IP parameter
-l 192.168.74.149
```

Cuadro 31: Modificación de archivo de configuración de Memcached

```
$ sudo service memcached restart
```

Cuadro 32: Reiniciar servicio de Memcached

7.1.7. Instalación de Etcd

Etcd es un almacén de *key-value* distribuido, diseñado para almacenar datos de configuración y coordinar servicios en entornos de red de manera confiable. Etcd asegura la consistencia y disponibilidad de los datos en entornos distribuidos, utilizando el algoritmo de consenso Raft para sincronizar los datos entre múltiples nodos. Esto permite que las aplicaciones y servicios compartan datos de configuración, detecten cambios y coordinen acciones sin riesgo de datos obsoletos o conflictos. Cada dato en Etcd está representado por una clave única y un valor asociado, lo que permite acceder rápidamente a configuraciones de forma segura, incluso en sistemas con alta concurrencia.

En OpenStack, Etcd se usa principalmente para almacenar información de red y datos de configuración que necesitan estar siempre actualizados en todos los nodos. Esto es especialmente útil en el servicio de Neutron, ya que permite coordinar la configuración de redes virtuales, subredes y puertos de manera sincronizada en los diferentes nodos de la nube. Etcd también ayuda en la gestión y detección de servicios en tiempo real, permitiendo que cada nodo sepa qué servicios están activos o en qué estado se encuentran, facilitando así la escalabilidad y estabilidad de la plataforma OpenStack.

Instalación en el *Controller Node*

Instalar los paquetes de Etcd. Luego en el archivo de configuración es necesario reemplazar todas las IPs de los siguientes parámetros, por la IP de gestión del *controller node* `192.168.74.149`. Los puertos `2379` y `2380` son usados para atender solicitudes externas al servicio, y para atender la comunicación entre nodos conectados por Etcd, respectivamente. Finalmente reiniciar el servicio para que el cambio entre en efecto y habilitarlo para que se ejecute con el inicio del sistema.

```
$ sudo apt install etcd
$ sudo nano /etc/default/etcd
```

Cuadro 33: Instalar y abrir archivo de configuración de Etcd

```
ETCD_NAME="controller"
ETCD_DATA_DIR="/var/lib/etcd"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
ETCD_INITIAL_CLUSTER="controller=http://192.168.74.149:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://192.168.74.149:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://192.168.74.149:2379"
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
ETCD_LISTEN_CLIENT_URLS="http://192.168.74.149:2379"
```

Cuadro 34: Modificación de archivo de configuración de Etcd

```
$ sudo systemctl enable etcd
$ sudo systemctl restart etcd
```

Cuadro 35: Habilitar desde el arranque del sistema y reiniciar servicio de Etcd

7.2. Instalación de Keystone

El *identity service* de OpenStack —nombre código "Keystone"— provee un punto de integración para el manejo de autenticación, autorización y un catálogo de servicios implementados. El *identity service* es de manera típica el primer servicio con el que interactúan los usuarios. Una vez son autenticados, los usuarios pueden acceder a los demás servicios de OpenStack; otros servicios de OpenStack también hacen uso de Keystone para autenticar usuarios previo a su acceso. Tanto los usuarios, como otros servicios hacen uso de Keystone para el descubrimiento de servicios a través de su catálogo de servicios.

Cada servicio puede tener uno o más *endpoints*, y cada *endpoint* puede ser uno de tres tipos: *admin*, *internal* o *public*. En ambientes de producción los distintos *endpoints* pueden residir en redes distintas y estar expuestos únicamente a ciertos usuarios por motivos de seguridad. Por ejemplo, la *public* API puede ser visible a través de internet, la *admin*

API puede estar restringida a operadores dentro de la organización que manejan la infraestructura, y la *internal* API puede estar disponible únicamente para los nodos que alojan servicios de OpenStack. OpenStack también maneja el concepto de *regions*, separaciones a usualmente a nivel de centros de datos, para la escalabilidad.

En conjunto, las regiones, servicios y *endpoints* conforman dentro de Keystone el catálogo de servicios. Cada servicio de OpenStack implementado necesita entradas con sus consecuentes *endpoints* almacenados en el *identity service*.

Keystone contiene los siguientes tres componentes:

- Servidor: Un servidor centralizado que provee servicios de autenticación y autorización utilizando una interfaz RESTful.
- Controladores: *Drivers* o *back end* integrado al servidor. Son utilizados para acceder a la información en repositorios externos a OpenStack (servicios como bases de datos SQL o servidores LDAP).
- Módulos: Existen módulos de *middleware* que corren en el espacio de red de los componentes de OpenStack que usan Keystone; estos módulos interceptan *requests*, extraen las credenciales del usuario y las envían al servidor para verificar su autorización. La integración entre estos módulos de *middleware* y los componentes de OpenStack se da usando el Python Web Server Gateway Interface (WSGI).

7.2.1. Instalación en el *Controller Node*

Crear base de datos SQL llamada `keystone` utilizando MariaDB, y brindar los permisos correspondientes; para este caso se le dan todos los privilegios al usuario de base de datos `keystone` sobre la base de datos `keystone`. El usuario `keystone` puede acceder desde localhost o desde cualquier IP identificándose con su `KEYSTONE_DBPASS`, para este caso, `onap`.

```
$ sudo mysql
```

Cuadro 36: Acceder a la consola de MySQL

```
MariaDB [(none)]> CREATE DATABASE keystone;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'
    localhost' IDENTIFIED BY 'KEYSTONE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@
    '%' IDENTIFIED BY 'KEYSTONE_DBPASS';
```

Cuadro 37: Creación y modificación de permisos de la base de datos de Keystone

Esta guía utiliza un servidor de HTTP Apache junto a mod_wsgi para manejar *requests* al *identity service* en el puerto 5000. Por defecto Keystone escucha a este puerto. Los pa-

quetes manejan toda la configuración de Apache (incluyendo la activación de mod_wsgi y la configuración de Keystone dentro de él).

Instalación de paquetes y configuración de archivos. Referenciar el acceso a la base de datos de Keystone y reemplazar la contraseña `KEYSTONE_DBPASS`. Configurar en la sección de `[token]` a `Fernet` como `provider`.

```
$ sudo apt install keystone
$ sudo nano /etc/keystone/keystone.conf
```

Cuadro 38: Instalación y apertura de archivo de configuración de Keystone

```
[database]
# Remove everything else from this section
connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/
    keystone

[token]
# ...
provider = fernet
```

Cuadro 39: Modificación de archivo de configuración de Keystone

Popular la base de datos del servicio de identidad. Este comando cambia al usuario `keystone` para ejecutar el comando `keystone-manage db_sync` desde la terminal, el cual cual crea el *schema* de la base de datos de Keystone.

```
$ sudo su
$ su -s /bin/sh -c "keystone-manage db_sync" keystone
```

Cuadro 40: Configuraciones de la base de datos de Keystone

Fernet es un método de autenticación y encriptación que genera *tokens* firmados. Estos *tokens* se utilizan para representar y validar la autenticación en OpenStack. Los siguientes comandos son utilizados para inicializar los repositorios de claves de Fernet a partir del usuario `keystone`.

```
$ sudo keystone-manage fernet_setup --keystone-user keystone --
    keystone-group keystone
$ sudo keystone-manage credential_setup --keystone-user keystone --
    keystone-group keystone
```

Cuadro 41: Manejo de claves de Fernet

Realizar el *bootstrap* del *identity service*, definir la contraseña de admin con `ADMIN_PASS`, crear los *endpoints* de Keystone y definir su región.

```
$ sudo keystone-manage bootstrap --bootstrap-password ADMIN_PASS --
   bootstrap-admin-url http://controller:5000/v3/ --bootstrap-
   internal-url http://controller:5000/v3/ --bootstrap-public-url
   http://controller:5000/v3/ --bootstrap-region-id RegionOne
```

Cuadro 42: *Bootstrap* de Keystone

Configurar el servidor Apache HTTP, referenciar al *hostname* del nodo controlador y reiniciar el servicio para que entren en efecto las modificaciones.

```
$ sudo nano /etc/apache2/apache2.conf
```

Cuadro 43: Apertura de archivo de configuración de Apache

```
# Add at the end if it doesn't previously exist
ServerName controller
```

Cuadro 44: Modificación de archivo de configuración de Apache

```
$ sudo service apache2 restart
```

Cuadro 45: Reiniciar servicio de Apache

Crear archivo con variables del sistema, esto es importante debido a que contiene las variables de la cuenta admin. Este archivo se ejecutará de manera habitual a lo largo del resto de la instalación de OpenStack. El archivo se almacena en el directorio `openstack_files/`, sin embargo, puede ser almacenado en cualquier sitio de fácil acceso. Reemplazar `ADMIN_PASS` por una contraseña.

```
$ mkdir openstack_files
$ nano openstack_files/admin-openrc
```

Cuadro 46: Crear y abrir archivo de variables del sistema

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

Cuadro 47: Modificación de archivo de variables del sistema

Ejecutar el archivo con variables de entorno del sistema para ganar los permisos para continuar con los siguientes pasos.

```
$ . openstack_files/admin-openrc
```

Cuadro 48: Cargar variables del sistema

Ejemplos de creación de dominios, proyectos, usuarios y roles. Utilizar la salida de los comandos como verificación del funcionamiento.

```
$ openstack domain create --description "An Example Domain" example
```

Cuadro 49: Creación de dominio de ejemplo

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | An Example Domain                |
| enabled     | True                             |
| id          | 415a5c0e688f4d4caa8731a77a0111d6 |
| name        | example                          |
| options     | {}                               |
| tags        | []                               |
+-------------+----------------------------------+
```

Cuadro 50: Salida esperada del comando de creación de dominio

```
$ openstack project create --domain default --description "Service
   Project" service
```

Cuadro 51: Creación de proyecto de servicio

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Service Project                  |
| domain_id   | default                          |
| enabled     | True                             |
| id          | 1f5ee61500ae4db38aaf4794b06255b6 |
| is_domain   | False                            |
| name        | service                          |
| options     | {}                               |
| parent_id   | default                          |
| tags        | []                               |
+-------------+----------------------------------+
```

Cuadro 52: Salida esperada del comando de creación de proyecto

```
$ openstack project create --domain default --description "Demo
   Project" myproject
```

Cuadro 53: Creación de proyecto de demostración

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Demo Project                     |
| domain_id   | default                          |
| enabled     | True                             |
| id          | 8a75a6b5169c48c8bf660d5e62f84fa1 |
| is_domain   | False                            |
| name        | myproject                        |
| options     | {}                               |
| parent_id   | default                          |
| tags        | []                               |
+-------------+----------------------------------+
```

Cuadro 54: Salida esperada del comando de creación de proyecto

```
$ openstack user create --domain default --password-prompt myuser
```

Cuadro 55: Creación de usuario de demostración

```
User Password: onap
Repeat User Password: onap
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 81650bdebd9c45d7bcd1a6fbc4fa0a9c |
| name                | myuser                           |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Cuadro 56: Salida esperada del comando de creación de usuario

```
$ openstack role create myrole
```

Cuadro 57: Creación de rol de demostración

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | None                             |
| domain_id   | None                             |
| id          | b516efc7dbbf468bb7a85a60822b9da1 |
| name        | myrole                           |
| options     | {}                               |
+-------------+----------------------------------+
```

Cuadro 58: Salida esperada del comando de creación de rol

```
$ openstack role add --project myproject --user myuser myrole
```

Cuadro 59: Asignación de rol a usuario y proyecto

7.2.2. Verificación de funcionalidad desde el *Controller Node*

Hacer `unset` de las variables de entorno `OS_AUTH_URL` y `OS_PASSWORD`

```
$ unset OS_AUTH_URL OS_PASSWORD
```

Cuadro 60: Eliminar variables de estado

Hacer *request* de un *token* de autenticación como usuario `admin`.

```
$ openstack --os-auth-url http://controller:5000/v3 --os-project-
   domain-name Default --os-user-domain-name Default --os-project-
   name admin --os-username admin token issue
```

Cuadro 61: Solicitar *token* de autenticación

```
+------------+-----------------------------------------------------+
| Field      | Value                                               |
+------------+-----------------------------------------------------+
| expires    | 2024-07-24T20:42:36+0000                            |
| id         | gAAAAABmoVKSRIqMcJIZasCk92-G1HX9KDbaWRL8Dnh1-otBh5WF|
|            | N1dWx3xcx0s9cBh-eMIX3woRx8cG0cFeX7L3z8wpYXx5ALq4fh50|
|            | qV11z5q5c6um5KJE2881PAYRWAS8CVWaeo7j8p9mHWb4zAkm2112|
| project_id | bbb262cc62f94e67b3f8d672ce66e6e8                    |
| user_id    | 374e31aabb84462d95b076dce29973df                    |
+------------+-----------------------------------------------------+
```

Cuadro 62: Salida esperada del comando de solicitud de *token*

Como el usuario `myuser` creado previamente, hacer *request* de *token* de autenticación.

```
$ openstack --os-auth-url http://controller:5000/v3 --os-project-
   domain-name Default --os-user-domain-name Default --os-project-
   name myproject --os-username myuser token issue
```

Cuadro 63: Soliciar *token* de autenticación

```
Password: onap
+------------+-----------------------------------------------------+
| Field      | Value                                               |
+------------+-----------------------------------------------------+
| expires    | 2024-07-24T20:43:15+0000                            |
| id         | gAAAAABmoV1TVXSURPTt90nZJawfRvA7mc085_Kjfyspuzg-Cd1X|
|            | _L1dAHQji9YaagKCjEDNYVPjSLOMPJbks_ftuZJD5xiIV5YZByTX|
|            | VZF5Iyv8_r28xC_b2Eybjdwb1ohB-5RuaqiDCfRx5LZXKwPNjaMa|
| project_id | 8a75a6b5169c48c8bf660d5e62f84fa1                    |
| user_id    | 81650bdebd9c45d7bcd1a6fbc4fa0a9c                    |
+------------+-----------------------------------------------------+
```

Cuadro 64: Salida esperada del comando de solicitud de *token*

Revisar el funcionamiento general reinsertando todas las variables de entorno.

```
$ . openstack_files/admin-openrc
$ openstack token issue
```

Cuadro 65: Cargar variables de entorno y solicitar *token* de autenticación

```
+------------+-----------------------------------------------------+
| Field      | Value                                               |
+------------+-----------------------------------------------------+
| expires    | 2024-07-24T20:45:42+0000                            |
| id         | gAAAAABmoVnmNHN1YBHTEuxzdwlxzmRVItih-9IH4qnzDYKcjY11|
|            | ox-N1H29Exyh_W1JZ_wSCAvhZkNds_YXZVOxTkUbg2adRvc_d9Lk|
|            | vcy60c2WE+2WqPG0naLyhcN60awrEJAZHD3C2LFxfLz71C4Gu648|
| project_id | bbb262cc62f94e67b3f8d672ce66e6e8                    |
| user_id    | 374e31aabb84462d95b076dce29973df                    |
+------------+-----------------------------------------------------+
```

Cuadro 66: Salida esperada del comando de solicitud de *token*

7.3. Instalación de Glance

7.3.1. Instalación en el *Controller Node*

El *image service* de OpenStack —nombre código "Glance"— posibilita a los usuarios el descubrir, registrar y recuperar imágenes de máquinas virtuales. Glance ofrece una API

REST que permite hacer consultas de metadatos de imágenes de máquinas virtuales y recuperar dichas imágenes. El *image service* permite almacenar las imágenes en distintas ubicaciones, desde manera local (en el *controller node*) hasta en nodos dedicados de *object storage*; por simplicidad, en esta guía se siguen los pasos para almacenar las imágenes en el *controller node* en el directorio `/var/lib/glance/images/`.
Glance se conforma de los siguientes componentes:

- Glance-api: API encargada de descubrimiento, registro y recuperación de imágenes.
- Base de datos: Dedicada al almacenamiento de metadatatos de imágenes.
- Repositorio de almacenamiento para archivos de imágenes: Como dice en su nombre, se encarga del almacenamiento de archivos de las imágenes. Permite el uso de varios tipos de backends, como el *local file system*, *object storage* (OpenStack Swift), HTTP, *datastore* de VMware, entre otros.
- Servicio de definición de metadatos: API que permite la definición de metadatos personalizados que pueden ser aplicados a distintos recursos como imágenes, volúmenes o sabores.

Crear base de datos SQL llamada `glance` utilizando MariaDB, y brindar los permisos correspondientes; para este caso se le dan todos los privilegios al usuario de base de datos `glance` sobre la base de datos `glance`. El usuario `glance` puede acceder desde localhost o desde cualquier IP identificándose con su `GLANCE_DBPASS`, para este caso, `onap`.

```
$ sudo mysql
```

Cuadro 67: Acceder a la consola de MySQL

```
MariaDB [(none)]> CREATE DATABASE glance;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'
    localhost' IDENTIFIED BY 'GLANCE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%'
    IDENTIFIED BY 'GLANCE_DBPASS';
```

Cuadro 68: Creación y modificación de permisos de la base de datos de Glance

Cargar variables de entorno para ganar acceso a commandos exclusivos del administrador desde la CLI.

```
$ . openstack_files/admin-openrc
```

Cuadro 69: Cargar archivo de variables de entorno del sistema

Crear un usuario `glance` que se utiliza para la integración del *image service* con Keystone y los demás servicios, designar como servicio a este usuario y agregarle permisos de administrador.

```
$ openstack user create --domain default --password-prompt glance
```

Cuadro 70: Creación de usuario de Glance

```
User Password: onap
Repeat User Password: onap
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 3f4e777c4062483ab8d9edd7dff829df |
| name                | glance                           |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Cuadro 71: Salida esperada del comando de creación de usuario

```
$ openstack role add --project service --user glance admin
```

Cuadro 72: Modificar propiedades de usuario glance

Añadir el servicio de tipo `image` para Glance dentro del catálogo de servicios de Keystone.

```
$ openstack service create --name glance --description "OpenStack
    Image" image
```

Cuadro 73: Crear servicio de Glance

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Image                  |
| enabled     | True                             |
| id          | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| name        | glance                           |
| type        | image                            |
+-------------+----------------------------------+
```

Cuadro 74: Salida esperada del comando de creación de servicio

Crear los *endpoints* de Glance y definir su región.

```
$ openstack endpoint create --region RegionOne image public http://
   controller:9292
$ openstack endpoint create --region RegionOne image internal http
   ://controller:9292
$ openstack endpoint create --region RegionOne image admin http://
   controller:9292
```

Cuadro 75: Creación de *endpoints* de Glance

```
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 340be3625e9b4239a6415d034e98aace |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | a6e4b153c2ae4c919eccfdbb7dceb5d2 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 0c37ed58103f4300a84ff125a539032d |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```

Cuadro 76: Salida esperada del comando de creación de *endpoints*

Instalación de paquetes de Glance y configuraciones adicionales a los archivos instalados. Es particularmente importante la sección `[glance_store]`, debido a que trata en detalle el comportamiento de glance, más allá de su integración al despliegue de OpenStack y su acople a otros servicios como Keystone. Los atributos de dicha sección definen:

- `stores = file,http`: Define los tipos de almacenamiento que Glance soportará. En este caso:
  - `file`: Almacenamiento en un sistema de archivos local.
  - `http`: Permite acceder a imágenes usando URLs.
- `default_store = file`: Define `file` como el almacenamiento predeterminado donde se guardarán las imágenes.
- `filesystem_store_datadir = /var/lib/glance/images/`: Especifica la ruta en el sistema de archivos donde Glance almacenará las imágenes si se utiliza `file` como backend de almacenamiento.

```
$ sudo apt install glance
$ sudo nano /etc/glance/glance-api.conf
```

Cuadro 77: Instalación y apertura de archivo de configuración de Glance

```
[database]
# ...
connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance

[keystone_authtoken]
# Remove everything else from this section
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = GLANCE_PASS

[paste_deploy]
# ...
flavor = keystone

[glance_store]
# ...
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/

[oslo_limit]
# Add whole section if it doesn't exist
auth_url = http://controller:5000
auth_type = password
user_domain_id = default
username = glance
system_scope = all
password = MY_PASSWORD
region_name = RegionOne

[DEFAULT]
# ...
# use_keystone_quotas = True
```

Cuadro 78: Modificación de archivo de configuración de Glance

Antes de continuar con el siguiente paso, es necesario añadir permisos de lectura globales al usuario `glance`, es decir, aumentar su alcance de proyecto a nivel de sistema al acceso de lectura; no es necesario —ni recomendado— añadir permisos adicionales de escritura ni ejecución.

```
$ openstack role add --user glance --user-domain Default --system
   all reader
```

Cuadro 79: Cambiar permisos de usuario de Glance

Popular la base de datos del *image service*. Este comando cambia al usuario `glance` para ejecutar el comando `glance-manage db_sync` desde la terminal, el cual cual crea el *schema* de la base de datos de Glance.

```
$ sudo su
$ su -s /bin/sh -c "glance-manage db_sync" glance
```

Cuadro 80: Configuraciones de la base de datos de Glance

Finalizar instalación y reiniciar glance para que todos los cambios surtan efecto.

```
$ sudo service glance-api restart
```

Cuadro 81: Reiniciar servicio de Glance

7.3.2. Verificación de funcionalidad desde el *Controller Node*

Desde el *controller node*, primero cargar variables de entorno para poder ejecutar comandos como administrador, luego descargar de la web una imagen de Cirros, un sistema operativo liviano utilizado como prueba de concepto en ambientes de nube. Después que se finalice la descarga, añadir a glance la imagen y finalmente listar todas las imágenes registradas en Glance para garantizar que todo el funcionamiento sea el esperado.

```
$ . openstack_files/admin-openrc
$ wget http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-
    disk.img
$ glance image-create --name "cirros" --file cirros-0.4.0-x86_64-
    disk.img --disk-format qcow2 --container-format bare --visibility
    =public
```

Cuadro 82: Cargar variables de entorno y descargar imagen de Cirros

```
+------------------+------------------------------------------------+
| Field            | Value                                          |
+------------------+------------------------------------------------+
| checksum         | 133eae9fb1c98f45894a4e60d8736619               |
| container_format | bare                                           |
| created_at       | 2015-03-26T16:52:10Z                           |
| disk_format      | qcow2                                          |
| file             | /v2/images/cc5c6982-4910-471e-b864-101b5/file  |
| id               | cc5c6982-4910-471e-b864-1098015901b5           |
| min_disk         | 0                                              |
| min_ram          | 0                                              |
| name             | cirros                                         |
| owner            | ae7a98326b9c455588edd2656d723b9d               |
| protected        | False                                          |
| schema           | /v2/schemas/image                              |
| size             | 13200896                                       |
| status           | active                                         |
| tags             |                                                |
| updated_at       | 2015-03-26T16:52:10Z                           |
| virtual_size     | None                                           |
| visibility       | public                                         |
+------------------+------------------------------------------------+
```

Cuadro 83: Salida esperada del comando de creación de imagen

```
1 $ glance image-list
```

Cuadro 84: Solicitar lista de imágenes registradas en Glance

```
+--------------------------------------+--------+--------+
| ID                                   | Name   | Status |
+--------------------------------------+--------+--------+
| 38047887-61a7-41ea-9b49-27987d5e8bb9 | cirros | active |
+--------------------------------------+--------+--------+
```

Cuadro 85: Salida esperada del comando de lista de imágenes

7.4. Instalación de Placement

El servicio de Placement en OpenStack permite gestionar y asignar eficientemente los recursos físicos (CPU, RAM, almacenamiento, etc.) de los nodos a las instancias de máquinas virtuales y otros recursos de infraestructura. Placement actúa como un intermediario centralizado que proporciona a otros servicios, como Nova, una visión clara y precisa del inventario de recursos disponibles en el entorno. Esto permite que los recursos se asignen de manera óptima, evitando conflictos y maximizando el uso de los recursos físicos.

Placement se basa en una base de datos centralizada y una API RESTful para registrar y consultar inventarios de recursos, reservas y asignaciones. Al recibir solicitudes de recursos,

Placement evalúa la disponibilidad en función de los criterios especificados (como requisitos de hardware y capacidades de almacenamiento) y devuelve una lista de opciones viables. Placement facilita la toma de decisiones informadas y ayuda a evitar el aprovisionamiento insuficiente o excesivo en la infraestructura.

7.4.1. Instalación en el *Controller Node*

Crear base de datos SQL llamada `placement` utilizando MariaDB, y brindar los permisos correspondientes; para este caso se le dan todos los privilegios al usuario de base de datos `placement` sobre la base de datos `placement`. El usuario `placement` puede acceder desde localhost o desde cualquier IP identificándose con su `PLACEMENT_DBPASS`, para este caso, `onap`.

```
$ sudo mysql
```

Cuadro 86: Acceder a la consola de MySQL

```
MariaDB [(none)]> CREATE DATABASE placement;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'
    @'localhost' IDENTIFIED BY 'PLACEMENT_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'
    @'%' IDENTIFIED BY 'PLACEMENT_DBPASS';
```

Cuadro 87: Creación y modificación de permisos de la base de datos de Placement

Cargar variables de entorno para ganar acceso a commandos exclusivos del administrador desde la CLI.

```
$ . openstack_files/admin-openrc
```

Cuadro 88: Cargar archivo de variables de entorno del sistema

Crear un usuario `placement` que se utiliza para la integración del placement service con Keystone y los demás servicios, designar como servicio a este usuario y agregarle permisos de administrador. Finalmente añadirlo al catálogo de servicios.

```
$ openstack user create --domain default --password-prompt placement
```

Cuadro 89: Creación de usuario de Placement

```
User Password: onap
Repeat User Password: onap
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | fa742015a6494a949f67629884fc7ec8 |
| name                | placement                        |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Cuadro 90: Salida esperada del comando de creación de usuario

```
$ openstack role add --project service --user placement admin
```

Cuadro 91: Modificar propiedades de usuario de Placement

Añadir el servicio de tipo placement para Placement dentro del catálogo de servicios de Keystone.

```
$ openstack service create --name placement --description "Placement
    API" placement
```

Cuadro 92: Crear servicio de Placement

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Placement API                    |
| enabled     | True                             |
| id          | 2d1a27022e6e4185b86adac4444c495f |
| name        | placement                        |
| type        | placement                        |
+-------------+----------------------------------+
```

Cuadro 93: Salida esperada del comando de creación de servicio

Crear los *endpoints* de Placement y definir su región.

```
$ openstack endpoint create --region RegionOne placement public http
   ://controller:8778
$ openstack endpoint create --region RegionOne placement internal
   http://controller:8778
$ openstack endpoint create --region RegionOne placement admin http
   ://controller:8778
```

Cuadro 94: Creación de *endpoints* de Placement

```
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 2b1b2637908b4137a9c2e0470487cbc0 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 2d1a27022e6e4185b86adac4444c495f |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://controller:8778           |
+--------------+----------------------------------+

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 02bcda9a150a4bd7993ff4879df971ab |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 2d1a27022e6e4185b86adac4444c495f |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://controller:8778           |
+--------------+----------------------------------+

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 3d71177b9e0f406f98cbff198d74b182 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 2d1a27022e6e4185b86adac4444c495f |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://controller:8778           |
+--------------+----------------------------------+
```

Cuadro 95: Salida esperada del comando de creación de *endpoints*

Instalación de paquetes de Placement y configuraciones adicionales a los archivos instalados.

```
$ sudo apt install placement-api
$ sudo nano /etc/placement/placement.conf
```

Cuadro 96: Instalación y apertura de archivo de configuración de Placement

```
[placement_database]
# ...
connection = mysql+pymysql://placement:PLACEMENT_DBPASS@controller/
    placement

[api]
# ...
auth_strategy = keystone

[keystone_authtoken]
# Remove everything else from this section
auth_url = http://controller:5000/v3
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = placement
password = PLACEMENT_PASS
```

Cuadro 97: Modificación de archivo de configuración de Placement

Popular la base de datos del *placement service*. Este comando cambia al usuario `placement` para ejecutar el comando `placement-manage db_sync` desde la terminal, el cual cual crea el *schema* de la base de datos de Placement.

```
$ sudo su
$ su -s /bin/sh -c "placement-manage db sync" placement
```

Cuadro 98: Configuraciones de la base de datos de Placement

Añadir el usuario del sistema al grupo de Placement, esto es debido a que, de no hacerlo, fallará el comando `placement-status upgrade check`. Después de ejecutar el comando, es necesario hacer *logout* del usuario actual para que surta efecto este comando. La explicación del comando es la siguiente:

- `usermod`: Modificar las propiedades de un usuario en el sistema, como su nombre de usuario, grupo principal, grupos adicionales, o su shell de inicio.
- `-a -G`: `a` significa *append* (agregar) y se usa junto con `G` para agregar el usuario a un grupo sin eliminarlo de otros grupos a los que ya pertenece.

- `YOUR_USERNAME`: Reemplazar por el nombre del usuario que se desea añadir al grupo `placement`. Este es el usuario que recibirá los permisos del grupo especificado. El usuario por elegir debe ser el usuario habitual del sistema (`openstackcontroller`, para este caso).

```
$ sudo usermod -a -G placement YOUR_USERNAME
$ exit
```

Cuadro 99: Modificación de usuario Placement

Finalizar instalación y reiniciar el servidor web para que todos los cambios surtan efecto.

```
$ sudo service apache2 restart
```

Cuadro 100: Reiniciar servicio de apache2

7.4.2. Verificación de funcionalidad desde el *Controller Node*

Primero cargar variables de entorno para poder ejecutar comandos como administrador.

```
$ . openstack_files/admin-openrc
```

Cuadro 101: Cargar archivo de variables de entorno del sistema

Revisar el estado de la base de datos y de la instalación del servicio de Placement y revisar si está preparado para una actualización.

```
$ placement-status upgrade check
```

Cuadro 102: Revisar estado del servicio de Placement

```
+----------------------------------------+
| Upgrade Check Results                  |
+----------------------------------------+
| Check: Missing Root Provider IDs       |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
| Check: Incomplete Consumers            |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
| Check: Policy File JSON to YAML Migration |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
```

Cuadro 103: Salida esperada del comando de revisión de estado

Instalar el *plugin* para el openstackclient CLI llamada `osc-placement`, la cuál es útil para la inspección y manipulación de recursos dentro de Placement. Después, utilizando este *plugin*, listar todas las clases de recursos disponibles para placement y luego todos los *traits* o atributos asignables para los recursos. Revisar que la salida de estos comandos sea apropiada.

```
$ sudo apt install python3-osc-placement
```

Cuadro 104: Instalación de paquetes adicionales

```
$ openstack --os-placement-api-version 1.2 resource class list --
   sort-column name
```

Cuadro 105: Listar clases de recursos disponibles

```
+----------------------------------------+
| name                                   |
+----------------------------------------+
| DISK_GB                                |
| FPGA                                   |
| IPV4_ADDRESS                           |
| MEMORY_MB                              |
| MEM_ENCRYPTION_CONTEXT                 |
| NET_BW_EGR_KILOBIT_PER_SEC             |
| NET_BW_IGR_KILOBIT_PER_SEC             |
| ...                                    |
```

Cuadro 106: Salida esperada del comando de listar clases de recursos

```
$ openstack --os-placement-api-version 1.6 trait list --sort-column
   name
```

Cuadro 107: Listar atributos asignables a recursos

```
+---------------------------------------+
| name                                  |
+---------------------------------------+
| COMPUTE_ACCELERATORS                  |
| COMPUTE_ARCH_AARCH64                  |
| COMPUTE_ARCH_MIPSEL                   |
| COMPUTE_ARCH_PPC64LE                  |
| COMPUTE_ARCH_RISCV64                  |
| COMPUTE_ARCH_S390X                    |
| COMPUTE_ARCH_X86_64                   |
| ...                                   |
```

Cuadro 108: Salida esperada del comando de listar atributos asignables

7.5. Instalación de Nova

El *compute service* de OpenStack —nombre código "Nova"— es el encargado de proporcionar un método para aprovisionar instancias de cómputo. A través de Nova, es posible crear máquinas virtuales, servidores *bare-metal* (con el uso de Ironic), e incluso tiene un soporte limitado para contenedores de sistema. Nova está diseñado para ejecutarse como un conjunto de procesos o *daemons*, permitiendo administrar la infraestructura de cómputo en un entorno de nube.

Nova sigue una arquitectura *shared nothing* (sin componentes compartidos) basada en mensajería y altamente distribuida, lo que permite escalar horizontalmente sus servicios. Cada uno de sus componentes realiza funciones específicas y se comunica a través de mensajería RPC usando la biblioteca oslo.messaging, que actúa como capa de abstracción sobre aplicativos de *message queueing* como RabbitMQ.

- API: Es la interfaz que recibe las solicitudes REST desde el exterior. Este componente procesa peticiones HTTP, como crear, listar, detener o eliminar instancias, y las convierte en comandos internos que se envían a otros servicios de Nova a través de las colas de mensajería. Las solicitudes a menudo implican lecturas y escrituras en la base de datos y pueden desencadenar mensajes RPC para otros servicios de Nova.
- Nova-scheduler: Es el componente encargado de decidir en qué nodo se ejecutará cada instancia. Para tomar esta decisión, el *scheduler* consulta la información de recursos disponibles proporcionada por el servicio Placement. Esto permite optimizar la asignación de instancias, basándose en factores como disponibilidad de CPU, memoria y almacenamiento en cada nodo.
- Nova-compute: Es el servicio que administra la comunicación con el hipervisor en cada

nodo de cómputo, permitiendo el aprovisionamiento y gestión del ciclo de vida de las instancias (crear, eliminar, suspender, etc.).

- Nova-conductor: Actúa como intermediario entre el *compute service* y la base de datos. Nova-conductor maneja solicitudes que requieren coordinación, como la creación y redimensionamiento de instancias. Este componente también funciona como un *proxy* de base de datos, ya que los nodos de cómputo no acceden directamente a la base de datos para garantizar una mayor seguridad y estabilidad.
- Base de datos: Nova utiliza una base de datos SQL compartida por sus múltiples componentes para almacenar datos importantes sobre las instancias, como su estado, asignación de recursos y configuración. La base de datos también permite que los componentes se mantengan en sincronía y coordinen el estado de las instancias en toda la infraestructura.
- Placement: Es un servicio independiente pero esencial para Nova, ya que realiza el seguimiento del inventario de recursos disponibles (CPU, memoria, almacenamiento) en los nodos de cómputo. Al recibir solicitudes de recursos, Placement proporciona la información que necesita el *scheduler* para asignar instancias a los nodos de cómputo de manera eficiente, maximizando el uso de los recursos.

7.5.1. Instalación en el *Controller Node*

Crear tres bases de datos SQL, en cada una de ellas se brindan todos los privilegios al usuario de base de datos. El usuario puede acceder desde localhost o desde cualquier IP identificándose con su `NOVA_DBPASS`, para este caso, `onap`. Las bases de datos tienen las siguientes funciones:

- `nova_api`: Almacena los datos relacionados con las solicitudes de API de Nova, como el inicio, parada y consulta de instancias. Esta separación reduce la carga sobre la base de datos principal y mejora la escalabilidad del servicio.
- `nova`: Contiene la información crítica sobre las instancias, incluyendo su estado y recursos asignados. Es la base de datos principal de Nova para gestionar el ciclo de vida de las máquinas virtuales.
- `nova_cell0`: Es utilizada en la arquitectura de celdas de Nova para registrar instancias fallidas o inactivas. Facilita la administración y diagnóstico de fallos sin afectar las celdas activas.

```
1 $ sudo mysql
```

Cuadro 109: Acceder a la consola de MySQL

```
MariaDB [(none)]> CREATE DATABASE nova_api;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'
    localhost' IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%'
    IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> CREATE DATABASE nova;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'
    localhost' IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%'
    IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> CREATE DATABASE nova_cell0;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'
    localhost' IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%'
     IDENTIFIED BY 'NOVA_DBPASS';
```

Cuadro 110: Creación y modificación de permisos de las bases de datos de Nova

Cargar variables de entorno para ganar acceso a commandos exclusivos del administrador desde la CLI.

```
$ . openstack_files/admin-openrc
```

Cuadro 111: Cargar archivo de variables de entorno del sistema

Crear un usuario `nova` que se utiliza para la integración del *compute service* con Keystone y los demás servicios, designar como servicio a este usuario y agregarle permisos de administrador.

```
$ openstack user create --domain default --password-prompt nova
```

Cuadro 112: Creación de usuario de Nova

```
User Password: onap
Repeat User Password: onap
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 8a7dbf5279404537b1c7b86c033620fe |
| name                | nova                             |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Cuadro 113: Salida esperada del comando de creación de usuario

```
$ openstack role add --project service --user nova admin
```

Cuadro 114: Modificar propiedades de usuario Nova

Añadir el servicio de tipo `compute` para Nova dentro del catálogo de servicios de Keystone.

```
$ openstack service create --name nova --description "OpenStack
   Compute" compute
```

Cuadro 115: Crear servicio de Nova

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Compute                |
| enabled     | True                             |
| id          | 060d59eac51b4594815603d75a00aba2 |
| name        | nova                             |
| type        | compute                          |
+-------------+----------------------------------+
```

Cuadro 116: Salida esperada del comando de creación de servicio

Crear los *endpoints* de Nova y definir su región.

```
$ openstack endpoint create --region RegionOne compute public http
   ://controller:8774/v2.1
$ openstack endpoint create --region RegionOne compute internal http
   ://controller:8774/v2.1
$ openstack endpoint create --region RegionOne compute admin http://
   controller:8774/v2.1
```

Cuadro 117: Creación de *endpoints* de Nova

```
+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 3c1caa473bfe4390a11e7177894bcc7b         |
| interface    | public                                   |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 060d59eac51b4594815603d75a00aba2         |
| service_name | nova                                     |
| service_type | compute                                  |
| url          | http://controller:8774/v2.1              |
+--------------+------------------------------------------+

+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | e3c918de680746a586eac1f2d9bc10ab         |
| interface    | internal                                 |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 060d59eac51b4594815603d75a00aba2         |
| service_name | nova                                     |
| service_type | compute                                  |
| url          | http://controller:8774/v2.1              |
+--------------+------------------------------------------+

+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 38f7af91666a47cfb97b4dc790b94424         |
| interface    | admin                                    |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 060d59eac51b4594815603d75a00aba2         |
| service_name | nova                                     |
| service_type | compute                                  |
| url          | http://controller:8774/v2.1              |
+--------------+------------------------------------------+
```

Cuadro 118: Salida esperada del comando de creación de *endpoints*

Instalación de paquetes del servicio de cómputo.

```
1 $ sudo apt install nova-api nova-conductor nova-novncproxy nova-
    scheduler
```

Cuadro 119: Instalación de paquetes de Nova

Nota, antes de realizar la configuración de los servicios de nova, seguir a la guía de instalación del *networking service* Neutron y completar todos los pasos hasta la instalación de sus componentes con el comando:
`sudo apt install neutron-server ... neutron-metadata-agent`.

Configuración básica de Nova. Para este archivo de configuración es de particular interés que se llene de manera correcta la sección de `[neutron]`, debido a que son servicios co-dependientes. También es de interés configurar la sección `[vnc]` de tal manera que quede habilitado.

```
$ sudo nano /etc/nova/nova.conf
```

Cuadro 120: Apertura de archivo de configuración de Nova

```
[api_database]
# ...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api

[database]
# ...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova

[DEFAULT]
# ...
# Due to a packaging bug, remove the log_dir option from the section
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672/
my_ip = 192.168.74.149

[api]
# ...
auth_strategy = keystone

[keystone_authtoken]
# Remove all other options
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS

[service_user]
# ...
send_service_user_token = true
auth_url = http://controller:5000/identity
auth_strategy = keystone
auth_type = password
project_domain_name = Default
project_name = service
```

```
user_domain_name = Default
username = nova
password = NOVA_PASS

[neutron]
# ...
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = onap
service_metadata_proxy = true
metadata_proxy_shared_secret = METADATA_SECRET

[vnc]
# ...
enabled = true
server_listen = 192.168.74.149
server_proxyclient_address = 192.168.74.149

[glance]
# ...
api_servers = http://controller:9292

[oslo_concurrency]
# ...
lock_path = /var/lib/nova/tmp

[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS

[scheduler]
discover_hosts_in_cells_interval = 300
```

Cuadro 121: Modificación de archivo de configuración de Nova

```
$ sudo nano /etc/neutron/metadata_agent.ini
```

Cuadro 122: Apertura de archivo de configuración de Neutron

```
[DEFAULT]
# ...
nova_metadata_host = controller
metadata_proxy_shared_secret = METADATA_SECRET
```

Cuadro 123: Modificación de archivo de configuración de Neutron

Popular las bases de datos del *compute service*. Este comando cambia al usuario `nova` para ejecutar los siguientes comandos desde la terminal:

- `nova-manage api_db sync`: Crea el *schema* de la base de datos para gestionar solicitudes de la API de nova.
- `nova-manage cell_v2 map_cell0`: Se agregan datos de configuración en `nova_api` para que Nova pueda localizar y usar `cell0` en caso de errores en la asignación de instancias.
- `nova-manage cell_v2 create_cell -name=cell1 -verbose`: Se insertan registros en la base de datos `nova_api`, pero esta vez para crear `cell1`, que es una celda activa y funcional.
- `nova-manage db sync`: Crea el *schema* de la base de datos de las instancias de nova.

```
$ sudo su
$ su -s /bin/sh -c "nova-manage api_db sync" nova
$ su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova
$ su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --
    verbose" nova
$ su -s /bin/sh -c "nova-manage db sync" nova
```

Cuadro 124: Configuraciones de las bases de datos de Nova

Verificar que las celdas de Nova fueron añadidas correctamente.

```
$ su -s /bin/sh -c "nova-manage cell_v2 list_cells" nova
```

Cuadro 125: Listar celdas de Nova

```
+-------+------+---------------+----------------------+----------+
|  Name | UUID | Transport URL | Database Connection | Disabled |
+-------+------+---------------+----------------------+----------+
| cell0 | 0000 |    none:/     | mysql+pymysql://*** |  False   |
| cell1 | 60a9 | rabbit://**** | mysql+pymysql://*** |  False   |
+-------+------+---------------+----------------------+----------+
```

Cuadro 126: Salida esperada del comando de listar celdas

Finalizar instalación en nodo controlador, reiniciar todos los servicios cuya configuración fue modificada.

```
$ service nova-api restart
$ service nova-scheduler restart
$ service nova-conductor restart
$ service nova-novncproxy restart
```

Cuadro 127: Reiniciar servicios de Nova

7.5.2. Instalación en el *Compute1 Node*

Instalar paquetes de nova sobre el nodo de cómputo y configurar sus archivos. Para este caso la sección más importante es `[vnc]` para poder definir correctamente el acceso por medio de una GUI a las instancias de máquinas virtuales.

- `enabled = true`: Habilitar el soporte para VNC.
- `server_listen = 0.0.0.0`: Permitir conexiones desde cualquier dirección de red.
- `server_proxyclient_address = 192.168.74.150`: Especificar la dirección del propio *compute1 node* que se utilizará para comunicarse con el *controller node*, ya que es la dirección que se usará acceder a la consola de las instancias.
- `novncproxy_base_url = http://192.168.74.149:6080/vnc_auto.html`: Definir la dirección URL que se utiliza para el acceso web de las instancias.

```
$ sudo apt install nova-compute
$ sudo nano /etc/nova/nova.conf
```

Cuadro 128: Instalación y apertura de archivo de configuración de Nova

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
my_ip = 192.168.6.252

[api]
# ...
auth_strategy = keystone

[keystone_authtoken]
# remove all other options from this section
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
```

```
project_name = service
username = nova
password = NOVA_PASS

[service_user]
send_service_user_token = true
auth_url = http://controller:5000/identity
auth_strategy = keystone
auth_type = password
project_domain_name = Default
project_name = service
user_domain_name = Default
username = nova
password = NOVA_PASS

[neutron]
# ...
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = onap

[vnc]
# ...
enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = 192.168.74.150
novncproxy_base_url = http://192.168.74.149:6080/vnc_auto.html

[glance]
# ...
api_servers = http://controller:9292

[oslo_concurrency]
# ...
lock_path = /var/lib/nova/tmp

[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS
```

Cuadro 129: Modificación de archivo de configuración de Nova

Revisar la compatibilidad con la aceleración de hardware para máquinas virtuales. Si el siguiente comando retorna un valor de 1 o más, quiere decir que sí es soportado y se puede proseguir al reinicio del servicio de `nova-compute` sin realizar más modificaciones; en caso no sea soportado, es necesario reemplazar el valor `virt_type = kvm` por `virt_type = qemu` como se muestra en las siguientes instrucciones.

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

Cuadro 130: Revisión de soporte de virtualización

```
$ sudo nano /etc/nova/nova-compute.conf
```

Cuadro 131: Apertura de archivo de configuración de Nova

```
[libvirt]
# ...
virt_type = qemu
```

Cuadro 132: Modificación de archivo de configuración de Nova

Reiniciar servicio de Nova después de finalizar los cambios.

```
$ sudo service nova-compute restart
```

Cuadro 133: Reiniciar servicio de Nova

7.5.3. Finalizar instalación en el *Controller Node*

Cargar variables de entorno para ejecutar los siguientes comandos. Primero listar los servicios de `nova-compute` activos y verificar que estén bien. Después de eso, se ejecuta el siguiente comando para el descubrimiento desde el *controller node* de todos los nodos tipo `compute` que estén disponibles, para así registrarlos en su base de datos. En caso estos comandos no funcionen adecuadamente, reiniciar los servicios y sistemas e intentarlo nuevamente.

```
$ . openstack_files/admin-openrc
$ openstack compute service list --service nova-compute
```

Cuadro 134: Cargar variables de entorno y listar servicios de nova-compute

```
+-----+--------------+--------------------+------+---------+-------+
| ID  | Binary       | Host               | Zone | Status  | State |
+-----+--------------+--------------------+------+---------+-------+
| ffb | nova-compute | openstackcompute1  | nova | enabled | up    |
+-----+--------------+--------------------+------+---------+-------+
```

Cuadro 135: Salida esperada del comando de listar servicios de nova-compute

```
$ sudo su
$ su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose"
   nova
```

Cuadro 136: Descubrimiento de nodos *compute*

```
Found 2 cell mappings.
Skipping cell0 since it does not contain hosts.
Getting computes from cell 'cell1': 4d3586c4-2884-4460-a527-
   d13258169a09
Found 0 unmapped computes in cell: 4d3586c4-2884-4460-a527-
   d13258169a09
```

Cuadro 137: Salida esperada del comando de descubrimiento de nodos *compute*

7.5.4. Verificación de funcionalidad desde el *Controller Node*

Cargar variables de entorno para ejecutar los siguientes comandos, primero listar todos los servicios de tipo compute reconocidos por OpenStack. Después, desplegar el catálogo con todos los servicios activos y sus *endpoints*. Verificar las imágenes disponibles para virtualizar, y para terminar desplegar el `upgrade check` de Nova.

```
$ . openstack_files/admin-openrc
$ openstack compute service list
```

Cuadro 138: Cargar variables de entorno y listar servicios de Nova

```
+----+----------------+------------+----------+---------+-------+
| ID | Binary         | Host       | Zone     | Status  | State |
+----+----------------+------------+----------+---------+-------+
| f6 | nova-scheduler | controller | internal | enabled | up    |
| 8d | nova-conductor | controller | internal | enabled | up    |
| 80 | nova-compute   | onap       | nova     | enabled | up    |
+----+----------------+------------+----------+---------+-------+
```

Cuadro 139: Salida esperada del comando de listar servicios de Nova

```
$ openstack catalog list
```

Cuadro 140: Listar catálogo de servicios de OpenStack

```
+-----------+-----------+-----------------------------------------+
| Name      | Type      | Endpoints                               |
+-----------+-----------+-----------------------------------------+
| neutron   | network   | RegionOne                               |
|           |           |   admin: http://controller:9696         |
|           |           | RegionOne                               |
|           |           |   public: http://controller:9696        |
|           |           | RegionOne                               |
|           |           |   internal: http://controller:9696      |
|           |           |                                         |
| nova      | compute   | RegionOne                               |
|           |           |   admin: http://controller:8774/v2.1    |
|           |           | RegionOne                               |
|           |           |   internal: http://controller:8774/v2.1 |
|           |           | RegionOne                               |
|           |           |   public: http://controller:8774/v2.1   |
|           |           |                                         |
| glance    | image     | RegionOne                               |
|           |           |   internal: http://controller:9292      |
|           |           | RegionOne                               |
|           |           |   public: http://controller:9292        |
|           |           | RegionOne                               |
|           |           |   admin: http://controller:9292         |
|           |           |                                         |
| keystone  | identity  | RegionOne                               |
|           |           |   public: http://controller:5000/v3/    |
|           |           | RegionOne                               |
|           |           |   admin: http://controller:5000/v3/     |
|           |           | RegionOne                               |
|           |           |   internal: http://controller:5000/v3/  |
|           |           |                                         |
| placement | placement | RegionOne                               |
|           |           |   admin: http://controller:8778         |
|           |           | RegionOne                               |
|           |           |   internal: http://controller:8778      |
|           |           | RegionOne                               |
|           |           |   public: http://controller:8778        |
|           |           |                                         |
+-----------+-----------+-----------------------------------------+
```

Cuadro 141: Salida esperada del comando de listar catálogo de servicios

```
$ openstack image list
```

Cuadro 142: Listar imágenes disponibles

```
+--------------------------------------+-----------------+--------+
| ID                                   | Name            | Status |
+--------------------------------------+-----------------+--------+
| 208c9f50-f086-4db5-983d-d638b888eba9 | cirros          | active |
+--------------------------------------+-----------------+--------+
```

Cuadro 143: Salida esperada del comando de listar imágenes

```
$ sudo nova-status upgrade check
```

Cuadro 144: Revisar estado de actualización de Nova

```
+----------------------------------------+
| Upgrade Check Results                  |
+----------------------------------------+
| Check: Cells v2                        |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
| Check: Placement API                   |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
| Check: Cinder API                      |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
| Check: Policy Scope-based Defaults     |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
| Check: Policy File JSON to YAML Migration |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
| Check: Older than N-1 computes         |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
| Check: hw_machine_type unset           |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
| Check: Service User Token Configuration |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
```

Cuadro 145: Salida esperada del comando de revisión de actualización

7.6. Instalación de Neutron

El *networking service* de OpenStack —nombre código "Neutron"— es el encargado de proporcionar una capa de infraestructura de red virtual en la nube y administrar la conectividad de las instancias. A través de Neutron, los proyectos pueden crear topologías de red avanzadas que incluyen servicios como *firewalls* y VPNs. Neutron proporciona un sistema flexible, permitiendo integrar diferentes equipos y *software* de red mediante el uso de *plugins*.

Funciones principales de Neutron:

- Gestión de redes y subredes: Neutron permite a los usuarios crear redes y subredes virtuales, que actúan como equivalentes virtuales de las redes físicas. Los usuarios pueden definir redes internas para sus instancias y, a través de subredes, asignar rangos de direcciones IP específicas para cada red.
- Enrutamiento y NAT: Neutron permite la creación de *routers* virtuales que conectan redes internas con redes externas, proporcionando salida a Internet mediante el protocolo NAT (*Network address translation*). Los *routers* también permiten que las subredes se comuniquen entre sí a través de sus interfaces, replicando la funcionalidad de un *router* físico.
- Redes externas: Una configuración de Neutron incluye al menos una red externa, que es una red física o virtualmente definida que permite que las instancias se comuniquen con el exterior del entorno OpenStack. Las redes externas están conectadas a los *routers* de Neutron, proporcionando una ventana hacia la red externa.
- Direcciones IP flotantes: Neutron permite asignar direcciones IP de redes externas a puertos en redes internas. Esto es esencial para exponer servicios específicos o dar acceso directo a las instancias desde redes externas o Internet. Las IP flotantes permiten que las instancias en redes privadas se accedan desde fuera del entorno de OpenStack, sin comprometer la seguridad de toda la red privada.
- Grupos de seguridad: Neutron soporta grupos de seguridad, que permiten definir reglas de *firewall* para controlar el tráfico entrante y saliente en las interfaces de red de las instancias. Los administradores pueden definir reglas para bloquear o permitir puertos, rangos de puertos o tipos de tráfico específicos, proporcionando un nivel de seguridad granular.

Componentes de Neutron:

- Neutron-server: Es el componente central que acepta y enruta las solicitudes de API hacia los *plugins* de red de Neutron, encargados de realizar las acciones específicas solicitadas. Neutron-server es el punto de entrada principal para crear y gestionar redes, subredes y puertos en el entorno de OpenStack.
- Plugins y agentes de Neutron: Neutron utiliza *plugins* y agentes para gestionar las operaciones de red. Los *plugins* controlan la creación de redes y subredes, conectan y desconectan puertos y asignan direcciones IP. Los *plugins* y agentes varían según el

proveedor y las tecnologías de red elegidas, ofreciendo soporte para múltiples equipos y soluciones como Cisco, Open vSwitch, Linux Bridge, y VMware NSX. Los agentes más comunes incluyen:

  - Agente L3 (Capa 3): Gestiona el enrutamiento y NAT para redes internas, permitiendo que las instancias accedan a redes externas.
  - Agente DHCP: Proporciona direcciones IP de manera dinámica a las instancias, facilitando la administración de direcciones en redes de autoservicio.
  - Agente del *plugin* de red: Administra las conexiones de red en cada nodo de cómputo, gestionando el tráfico de red en las instancias.

- Cola de mensajes: Neutron utiliza un aplicativo de cola de mensajes (como RabbitMQ) para coordinar la comunicación entre neutron-server y los diferentes agentes. La cola de mensajes permite que los cambios de estado en la red se transmitan de manera eficiente, asegurando la consistencia en el estado de la red.

7.6.1. Instalación en el *Controller Node*

Crear base de datos SQL llamada `neutron` utilizando MariaDB, y brindar los permisos correspondientes; para este caso se le dan todos los privilegios al usuario de base de datos `neutron` sobre la base de datos `neutron`. El usuario `neutron` puede acceder desde localhost o desde cualquier IP identificándose con su `NEUTRON_DBPASS`, para este caso, `onap`.

```
$ sudo mysql
```

Cuadro 146: Acceder a la consola de MySQL

```
MariaDB [(none)]> CREATE DATABASE neutron;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'
    localhost' IDENTIFIED BY 'NEUTRON_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%'
    IDENTIFIED BY 'NEUTRON_DBPASS';
```

Cuadro 147: Creación y modificación de permisos de la base de datos de Neutron

Cargar variables de entorno para ganar acceso a commandos exclusivos del administrador desde la CLI.

```
$ . openstack_files/admin-openrc
```

Cuadro 148: Cargar archivo de variables de entorno del sistema

Crear un usuario `neutron` que se utiliza para la integración del *networking service* con Keystone y los demás servicios, designar como servicio a este usuario y agregarle permisos de administrador.

```
$ openstack user create --domain default --password-prompt neutron
```

Cuadro 149: Creación de usuario de Neutron

```
User Password: onap
Repeat User Password: onap
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | fdb0f541e28141719b6a43c8944bf1fb |
| name                | neutron                          |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Cuadro 150: Salida esperada del comando de creación de usuario

```
$ openstack role add --project service --user neutron admin
```

Cuadro 151: Modificar propiedades de usuario neutron

Añadir el servicio de tipo `network` para Neutron dentro del catálogo de servicios de Keystone.

```
$ openstack service create --name neutron --description "OpenStack
    Networking" network
```

Cuadro 152: Crear servicio de Neutron

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Networking             |
| enabled     | True                             |
| id          | f71529314dab4a4d8eca427e701d209e |
| name        | neutron                          |
| type        | network                          |
+-------------+----------------------------------+
```

Cuadro 153: Salida esperada del comando de creación de servicio

Crear los `endpoints` de Neutron y definir su región.

```
$ openstack endpoint create --region RegionOne network public http
   ://controller:9696
$ openstack endpoint create --region RegionOne network internal http
   ://controller:9696
$ openstack endpoint create --region RegionOne network admin http://
   controller:9696
```

Cuadro 154: Creación de *endpoints* de Neutron

```
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 85d80a6d02fc4b7683f611d7fc1493a3 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 09753b537ac74422a68d2d791cf3714f |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+

+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 1ee14289c9374dffb5db92a5c112fc4e |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | f71529314dab4a4d8eca427e701d209e |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

Cuadro 155: Salida esperada del comando de creación de *endpoints*

Instalación de paquetes de Neutron, para este caso se empleó la configuración tipo *self-service networks*, en lugar de la opción *provider networks*; la opción tipo *self-service* tiene integradas todas las funcionalidades de *provider networks* sumado a más opciones. Para expandir en ambas opciones, visitar: guía de instalación de neutron.

Self-service networks permite a los usuarios crear redes virtuales privadas, completamente aisladas, dentro de su propio proyecto o *tenant*, además de proporcionar herramientas avanzadas para conectividad y acceso externo. Esta opción es ideal para entornos de *multi-tenancy* donde cada usuario o proyecto requiere un espacio de red aislado y la flexibilidad de controlar la conectividad de sus instancias. Los usuarios pueden:

- Crear redes privadas aisladas: Los usuarios pueden definir redes virtuales dentro de su proyecto que están aisladas de otras redes y proyectos. Este aislamiento total asegura que cada *tenant* tenga un entorno de red independiente, sin interferencias de otros usuarios o proyectos.
- Configurar *routers* virtuales y NAT: Para conectar redes internas a redes externas, *self-service networks* permite crear *routers* virtuales que facilitan la salida a internet. A través de estos *routers*, el tráfico de salida desde las redes privadas pasa por un proceso de Network Address Translation (NAT), permitiendo que las instancias en redes privadas accedan a redes externas sin estar expuestas directamente.
- Asignar direcciones IP flotantes: Las *self-service networks* permiten asignar direcciones IP flotantes a instancias específicas, permitiendo el acceso directo desde redes externas o internet a dichas instancias. Esto es útil para exponer ciertos servicios o aplicaciones sin comprometer la seguridad de toda la red privada.
- Gestionar redes internas y externas: Los usuarios pueden conectar múltiples redes internas dentro de su proyecto, y mediante *routers* virtuales, decidir cómo y cuándo conectar estas redes a redes externas, brindando mayor control sobre la segmentación y conectividad.

```
$ sudo apt install neutron-server neutron-plugin-ml2 neutron-
    linuxbridge-agent neutron-l3-agent neutron-dhcp-agent neutron-
    metadata-agent
```

Cuadro 156: Instalación de paquetes de Neutron

Configurar el *server component*. Es importante configurar de manera adecuada la sección de `[nova]` para que se integren de manera correcta los servicios de compute y networking. También es muy relevante para el funcionamiento la sección `[DEFAULT]` cuyos apartados son:

- `core_plugin = ml2`: Define el *plugin* principal de red que Neutron utilizará. En este caso, se usa ML2 (Modular Layer 2), que es un *plugin* modular de red que permite a Neutron gestionar redes de capa 2 y soporta múltiples tipos de tecnologías de red, como VLANs, VXLANs, y GRE.

- `service_plugins = router`: Especifica el *plugin* de servicio adicional que se cargará en Neutron. En este caso, el *plugin* `router` permitiendo crear y administrar *routers* virtuales, necesarios para la opción de *self-service networks*. Esto permite conectar redes privadas a redes externas y manejar NAT para la salida a internet.
- `allow_overlapping_ips = true`: Permite el uso de direcciones IP superpuestas en redes distintas. Esto es importante para entornos *multi-tenant* donde diferentes proyectos (*tenants*) pueden tener redes internas con el mismo rango de direcciones IP sin conflictos, ya que estas redes están aisladas entre sí.
- `notify_nova_on_port_status_changes = true`: Habilita la notificación a Nova cada vez que cambia el estado de un puerto de red (por ejemplo, cuando una instancia se conecta o desconecta de una red). Esta comunicación es importante para que Nova esté al tanto del estado de las conexiones de red de las instancias y actualice su estado de red en consecuencia.
- `notify_nova_on_port_data_changes = true`: Habilita la notificación a Nova cuando cambian los datos de un puerto en Neutron, como la asignación de direcciones IP. Esto permite que Nova mantenga la información de red actualizada para cada instancia, lo cual es crítico para la correcta administración de las redes y la conectividad de las instancias.

```
$ sudo nano /etc/neutron/neutron.conf
```

Cuadro 157: Apertura de archivo de configuración de Neutron

```
[database]
# remove all other connections
connection = mysql+pymysql://neutron:NEUTRON_DBPASS@controller/
    neutron

[DEFAULT]
# ...
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = true
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone
notify_nova_on_port_status_changes = true
notify_nova_on_port_data_changes = true

[keystone_authtoken]
# remove all other options in this section
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS

[nova]
# ...
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS

[oslo_concurrency]
# ...
lock_path = /var/lib/neutron/tmp
```

Cuadro 158: Modificación de archivo de configuración de Neutron

Configurar ML2 , elegido como `core_plugin` previamente. Sus apartados son:

- `[ml2]`
  - `type_drivers = flat,vlan,vxlan`: Define los tipos de redes que Neutron puede crear y gestionar. En este caso, están habilitados tres tipos:
    - `flat`: Red sin segmentación, que se mapea directamente a la red física sin aislamiento adicional. Útil para redes públicas o de acceso directo.

    - `vlan`: Red segmentada usando VLANs, que proporciona aislamiento de capa 2 mediante etiquetas VLAN en la red física.
    - `vxlan`: Protocolo de encapsulación que permite crear redes virtuales en una infraestructura de red IP. VXLAN es adecuado para redes privadas de *tenants*, ya que ofrece un aislamiento alto sin necesidad de VLANs en el hardware físico.
  - `tenant_network_types = vxlan`: Define el tipo de red predeterminado que Neutron asignará a las redes de los proyectos en OpenStack. Aquí se establece `vxlan`, lo cual permite crear redes privadas virtuales para cada *tenant*, ofreciendo aislamiento de red sin consumir VLANs físicas.
  - `mechanism_drivers = linuxbridge,l2population`: Especifica los controladores que Neutron usará para implementar la conectividad de red.
    - `linuxbridge`: Configura el uso de la herramienta Linux Bridge para gestionar el tráfico de red en el nivel de enlace (Capa 2) en los nodos de cómputo.
    - `l2population`: Optimiza el tráfico de multidifusión en redes VXLAN mediante la técnica de *layer 2 population*.
  - `extension_drivers = port_security`: Habilita el uso de la extensión `port_security`, que permite controlar el tráfico a nivel de puertos. Esta configuración es esencial para gestionar reglas de seguridad, como habilitar o deshabilitar el filtrado de paquetes de seguridad, permitiendo o bloqueando tráfico en los puertos según las políticas de seguridad de red.

- `[ml2_type_flat]`
  - `flat_networks = provider`: Especifica los nombres de las redes planas que Neutron puede usar. En este caso, se configura `provider`, que representa la red física mapeada para acceso directo sin encapsulación.

- `[ml2_type_vlan]`
  - `vni_ranges = 1:1000`: Define el rango de VNI (VXLAN Network Identifier) que Neutron puede asignar para las redes de los tenants. En este caso, permite crear hasta 1000 redes virtuales separadas, cada una con su propio VNI.

- `[security_group]`
  - `enable_ipset = true`: Habilita el uso de la herramienta `ipset` para optimizar el filtrado de reglas de seguridad en los grupos de seguridad. `ipset` permite manejar grandes cantidades de reglas de *firewall* de manera más eficiente, especialmente cuando hay muchas IPs o subredes a las cuales aplicar las reglas. Esta opción mejora el rendimiento y reduce el tiempo de procesamiento en configuraciones con un gran número de instancias y grupos de seguridad.

```
$ sudo nano /etc/neutron/plugins/ml2/ml2_conf.ini
```

Cuadro 159: Apertura de archivo de configuración de Neutron

```
[ml2]
# ...
type_drivers = flat,vlan,vxlan
tenant_network_types = vxlan
mechanism_drivers = linuxbridge,l2population
extension_drivers = port_security

[ml2_type_flat]
# ...
flat_networks = provider

[ml2_type_vxlan]
# ...
vni_ranges = 1:1000

[securitygroup]
# ...
enable_ipset = true
```

Cuadro 160: Modificación de archivo de configuración de Neutron

Configurar el agente de Linux Bridge descrito previamente.

- `[linux_bridge]`:
  - `physical_interface_mappings = provider:enp6s0`: Define la correspondencia entre las redes físicas en OpenStack y las interfaces de red del sistema operativo. En este caso `provider` es el nombre lógico que representa la red física dentro de OpenStack; mientras que `enp6s0` es la interfaz de red que tiene la IP de gestión del controller node (`192.168.74.149`).
- `[vxlan]`:
  - `enable_vxlan = true`: Habilita el uso de VXLAN en el agente de Linux Bridge.
  - `local_ip = 192.168.74.149`: Especifica la dirección IP local que se utilizará para el túnel de VXLAN. Esta IP es la dirección en el nodo controller a través de la cual se enviará y recibirá el tráfico encapsulado de VXLAN.
  - `l2_population = true`: Activa la característica de *L2 Population*, que optimiza el tráfico de difusión en redes VXLAN.
- `[securitygroup]`:
  - `enable_security_group = true`: Habilita el uso de grupos de seguridad en el agente de Linux Bridge. Los grupos de seguridad actúan como *firewalls* virtuales que controlan el tráfico entrante y saliente en las interfaces de red de las instancias, proporcionando un nivel adicional de seguridad.
  - `firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver`: Especifica el controlador de *firewall* que Neutron

usará para aplicar las reglas de los grupos de seguridad. En este caso, se usa `IptablesFirewallDriver`, que implementa el firewall mediante `iptables`, la herramienta de filtrado de paquetes en Linux. Este driver permite configurar reglas de *firewall* de manera granular para cada instancia, controlando qué tráfico se permite o se bloquea.

```
$ sudo nano /etc/neutron/plugins/ml2/linuxbridge_agent.ini
```

Cuadro 161: Apertura de archivo de configuración de Neutron

```
[linux_bridge]
physical_interface_mappings = provider:enp6s0

[vxlan]
enable_vxlan = true
local_ip = 192.168.74.149
l2_population = true

[securitygroup]
# ...
enable_security_group = true
firewall_driver = neutron.agent.linux.iptables_firewall.
   IptablesFirewallDriver
```

Cuadro 162: Modificación de archivo de configuración de Neutron

Verificar el estado de parámetros del sistema relacionados con el filtrado de tráfico en interfaces tipo *bridge*. En OpenStack, los nodos de red y los agentes de red (como el agente L3 y el agente de seguridad) dependen de `iptables` e `ip6tables` para aplicar reglas de *firewall* y gestionar el tráfico de red de manera segura. Configurar ambos parámetros con un valor de `1` es necesario para que el filtrado de tráfico sea efectivo en redes virtuales.

- `net.bridge.bridge-nf-call-iptables`: Este parámetro controla si el *kernel* debe pasar el tráfico IPv4 que atraviesa dispositivos *bridge* a través de las reglas de `iptables` para filtrado. Si el valor es `1`, significa que el tráfico en las interfaces *bridge* será sometido a las reglas de *firewall* definidas en `iptables`, permitiendo aplicar políticas de seguridad al tráfico de red.
- `net.bridge.bridge-nf-call-ip6tables`: Similar al anterior, este parámetro controla si el tráfico IPv6 en interfaces bridge pasa por las reglas de `ip6tables` (la versión de `iptables` para IPv6). Un valor de `1` indica que el tráfico IPv6 que atraviesa el *bridge* también se filtrará según las reglas definidas en `ip6tables`.

En caso no haya una salida igual a `1` después de ejecutar los primeros comandos, entonces ejecutar el comando `sudo modprobe br_netfilter` para solucionarlo. Una vez

realizado este cambio, ejecutar nuevamente las verificaciones para comprobar que ya fue solucionado.

```
$ sysctl net.bridge.bridge-nf-call-iptables
$ sysctl net.bridge.bridge-nf-call-ip6tables
$ sudo modprobe br_netfilter
```

Cuadro 163: Verificación de parámetros de filtrado de tráfico en interfaces *bridge*

Configurar el agente de capa 3. Declara como controlador `linuxbridge`.

```
$ sudo nano /etc/neutron/l3_agent.ini
```

Cuadro 164: Apertura de archivo de configuración de Neutron

```
[DEFAULT]
# ...
interface_driver = linuxbridge
```

Cuadro 165: Modificación de archivo de configuración de Neutron

Configurar el agente DHCP.

- `interface_driver = linuxbridge`: Configura el driver de interfaz que el agente DHCP usará, en este caso `linuxbridge`. Esto permite al agente DHCP gestionar interfaces de red virtuales utilizandolo.
- `dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq`: Define el driver DHCP que Neutron usará para asignar direcciones IP a las instancias. En este caso, se especifica `Dnsmasq`, una herramienta ligera de servidor DHCP que también puede manejar DNS y TFTP.
- `enable_isolated_metadata = true`: Habilita el acceso a metadatos para instancias en redes aisladas (que no están conectadas a una red externa).

```
$ sudo nano /etc/neutron/dhcp_agent.ini
```

Cuadro 166: Apertura de archivo de configuración de Neutron

```
[DEFAULT]
# ...
interface_driver = linuxbridge
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = true
```

Cuadro 167: Modificación de archivo de configuración de Neutron

Configurar el agente de metadatos en el nodo controlador.

```
$ sudo nano /etc/neutron/metadata_agent.ini
```

Cuadro 168: Apertura de archivo de configuración de Neutron

```
[DEFAULT]
# ...
nova_metadata_host = controller
metadata_proxy_shared_secret = METADATA_SECRET
```

Cuadro 169: Modificación de archivo de configuración de Neutron

Configurar el *compute service* para que esté coordinado con el *networking service*. Verificar que esté todo bien en la sección de Nova.

```
$ sudo nano /etc/nova/nova.conf
```

Cuadro 170: Apertura de archivo de configuración de Nova

```
[neutron]
# ...
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
service_metadata_proxy = true
metadata_proxy_shared_secret = METADATA_SECRET
```

Cuadro 171: Modificación de archivo de configuración de Nova

Finalizar la instalación. Ejecutar como usuario `neutron` el comando entre comillas, que inicializa o actualiza la estructura de la base de datos de Neutron de acuerdo con la configuración actual, creando las tablas necesarias y aplicando las últimas migraciones y reiniciar todos los servicios cuyas configuraciones fueron modificadas.

```
$ sudo su
$ su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/
    neutron.conf --config-file /etc/neutron/plugins/ml2/ml2_conf.ini
    upgrade head" neutron

$ service nova-api restart
$ service neutron-server restart
$ service neutron-linuxbridge-agent restart
$ service neutron-dhcp-agent restart
$ service neutron-metadata-agent restart
$ service neutron-l3-agent restart
```

Cuadro 172: Configuraciones finales de las bases de datos y reinicio de servicios modificados

7.6.2. Instalación en el *Compute1 Node*

Instalación de componentes de Neutron en el *compute1 node*, específicamente las herramientas relacionadas con Linux Bridge; después realizar configuraciones básicas para integrarlo al despliegue de OpenStack.

```
$ sudo apt install neutron-linuxbridge-agent
$ sudo nano /etc/neutron/neutron.conf
```

Cuadro 173: Instalación de paquetes de Neutron y apertura de archivo de configuración

```
[database]
# comment out any connection options

[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone

[keystone_authtoken]
# Comment out or remove any other options
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS

[oslo_concurrency]
# ...
lock_path = /var/lib/neutron/tmp
```

Cuadro 174: Modificación de archivo de configuración de Neutron

Configurar en el *compute service* el apartado de Neutron para que se integren ambos servicios correctamente.

```
$ sudo nano /etc/nova/nova.conf
```

Cuadro 175: Apertura de archivo de configuración de Nova

```
[neutron]
# ...
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
```

Cuadro 176: Modificación de archivo de configuración de Nova

Configurar el *bridge* para el nodo de cómputo.

- `[linux_bridge]`:

- `physical_interface_mappings = provider:enp2s0`: Define la correspondencia entre las redes físicas en OpenStack y las interfaces de red del sistema operativo. En este caso `provider` es el nombre lógico que representa la red física dentro de OpenStack; mientras que `enp2s0` es la interfaz de red que tiene la IP de gestión del *compute1 node* (`192.168.74.150`).

- `[vxlan]`:
  - `enable_vxlan = true`: Habilita el uso de VXLAN en el agente de Linux Bridge.
  - `local_ip = 192.168.74.150`: Especifica la dirección IP local que se utilizará para el túnel de VXLAN. Esta IP es la dirección en el *compute1 node* a través de la cual se enviará y recibirá el tráfico encapsulado de VXLAN.
  - `l2_population = true`: Activa la característica de `L2 Population`, que optimiza el tráfico de difusión en redes VXLAN.

- `[securitygroup]`:
  - `enable_security_group = true`: Habilita el uso de grupos de seguridad en el agente de Linux Bridge. Los grupos de seguridad actúan como *firewalls* que controlan el tráfico entrante y saliente en las interfaces de red de las instancias, proporcionando un nivel adicional de seguridad.
  - `firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver`: Especifica el controlador de *firewall* que Neutron usará para aplicar las reglas de los grupos de seguridad. En este caso, se implementa el firewall mediante `iptables`, la herramienta de filtrado de paquetes en Linux. Este driver permite configurar reglas de *firewall* de manera granular para cada instancia, controlando qué tráfico se permite o se bloquea.

```
$ sudo nano /etc/neutron/plugins/ml2/linuxbridge_agent.ini
```

Cuadro 177: Apertura de archivo de configuración de Neutron

```
[linux_bridge]
# ...
physical_interface_mappings = provider:enp2s0

[vxlan]
enable_vxlan = true
local_ip = 192.168.74.150
l2_population = true

[securitygroup]
# ...
enable_security_group = true
firewall_driver = neutron.agent.linux.iptables_firewall.
   IptablesFirewallDriver
```

Cuadro 178: Modificación de archivo de configuración de Neutron

De la misma manera que en el *controller node*, verificar que el estado de parámetros del sistema relacionados con el filtrado de tráfico en interfaces tipo *bridge* sea igual a `1`. En caso no haya una salida igual a `1` después de ejecutar los primeros comandos, entonces ejecutar el comando `sudo modprobe br_netfilter` para solucionarlo. Una vez realizado este cambio, ejecutar nuevamente las verificaciones para comprobar que ya fue solucionado.

```
$ net.bridge.bridge-nf-call-iptables
$ net.bridge.bridge-nf-call-ip6tables
$ sudo modprobe br_netfilter
```

Cuadro 179: Verificación de parámetros de filtrado de tráfico en interfaces *bridge*

Finalizar la instalación, reiniciando todos los servicios cuya configuración fue modificada.

```
$ sudo service nova-compute restart
$ sudo service neutron-linuxbridge-agent restart
```

Cuadro 180: Reiniciar servicios modificados

7.7. Instalación de Cinder

El *block storage service* de OpenStack —nombre código "Cinder"— , está diseñado para proporcionar volúmenes de almacenamiento a las máquinas virtuales gestionadas por Nova, a *hosts bare-metal* mediante Ironic, contenedores y otros entornos. Entre sus principales características, se destacan que tiene una arquitectura basada en componentes que le permite agregar nuevos comportamientos de manera rápida; una alta disponibilidad, pues está diseñado para escalar y manejar cargas de trabajo de gran envergadura; tolerancia a fallos debido a que sus procesos están aislados, evitando que los fallos se propaguen en cascada; y recuperación rápida a las fallas, ya que son fáciles de diagnosticar, depurar y corregir. Los componentes de Cinder son:

- Cinder-api: Acepta las solicitudes que provienen de usuarios o servicios y redirige estas solicitudes al servicio Cinder-volume para que se realicen las acciones correspondientes.
- Cinder-volume: Es el núcleo del servicio de almacenamiento, procesa las solicitudes relacionadas con los volúmenes mediante la interacción con otros servicios como Cinder-scheduler. También se comunica con los controladores específicos del almacenamiento físico subyacente (como LVM, iSCSI o almacenamiento en red) y utiliza una cola de mensajería para coordinar la comunicación entre los diferentes componentes.
- Cinder-scheduler *daemon*: Selecciona el nodo de almacenamiento óptimo donde se va a crear un volumen. Es similar al Nova-scheduler, que asigna recursos de computación para instancias.
- Cinder-backup *daemon*: Este servicio permite respaldar volúmenes hacia un proveedor de almacenamiento externo o especializado. Al igual que Cinder-volume, puede

trabajar con diferentes tipos de almacenamiento gracias a su arquitectura basada en *drivers*.

- Cola de mensajería: Coordina la comunicación entre los diferentes procesos de Cinder.

7.7.1. Instalación en el *Controller Node*

Crear base de datos SQL llamada `cinder` utilizando MariaDB, y brindar los permisos correspondientes; para este caso se le dan todos los privilegios al usuario de base de datos `cinder` sobre la base de datos `cinder`. El usuario `cinder` puede acceder desde localhost o desde cualquier IP identificándose con su `CINDER_DBPASS`, para este caso, `onap`.

```
$ sudo mysql
```

Cuadro 181: Acceder a la consola de MySQL

```
MariaDB [(none)]> CREATE DATABASE cinder;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'
    localhost' IDENTIFIED BY 'CINDER_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%'
    IDENTIFIED BY 'CINDER_DBPASS';
```

Cuadro 182: Creación y modificación de permisos de la base de datos de Cinder

Cargar variables de entorno para ganar acceso a commandos exclusivos del administrador desde la CLI.

```
$ . openstack_files/admin-openrc
```

Cuadro 183: Cargar archivo de variables de entorno del sistema

Crear un usuario `cinder` que se utiliza para la integración del *block storage service* con Keystone y los demás servicios, designar como servicio a este usuario y agregarle permisos de administrador. Finalmente añadirlo al catálogo de servicios.

```
$ openstack user create --domain default --password-prompt cinder
```

Cuadro 184: Creación de usuario de Cinder

```
User Password: onap
Repeat User Password: onap
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 9d7e33de3e1a498390353819bc7d245d |
| name                | cinder                           |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Cuadro 185: Salida esperada del comando de creación de usuario

```
$ openstack role add --project service --user cinder admin
```

Cuadro 186: Modificar propiedades de usuario cinder

Añadir el servicio de tipo `volumev3` para Cinder dentro del catálogo de servicios de Keystone.

```
$ openstack service create --name cinderv3 --description "OpenStack
   Block Storage" volumev3
```

Cuadro 187: Crear servicio de Cinder

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Block Storage          |
| enabled     | True                             |
| id          | ab3bbbef780845a1a283490d281e7fda |
| name        | cinderv3                         |
| type        | volumev3                         |
+-------------+----------------------------------+
```

Cuadro 188: Salida esperada del comando de creación de servicio

Crear los *endpoints* de Cinder y definir su región.

```
$ openstack endpoint create --region RegionOne volumev3 public http
   ://controller:8776/v3/%\(project_id\)s
$ openstack endpoint create --region RegionOne volumev3 internal
   http://controller:8776/v3/%\(project_id\)s
$ openstack endpoint create --region RegionOne volumev3 admin http
   ://controller:8776/v3/%\(project_id\)s
```

Cuadro 189: Creación de *endpoints* de Cinder

```
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 03fa2c90153546c295bf30ca86b1344b        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | ab3bbbef780845a1a283490d281e7fda        |
| service_name | cinderv3                                |
| service_type | volumev3                                |
| url          | http://controller:8776/v3/%(project_id)s |
+--------------+-----------------------------------------+

+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 94f684395d1b41068c70e4ecb11364b2        |
| interface    | internal                                |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | ab3bbbef780845a1a283490d281e7fda        |
| service_name | cinderv3                                |
| service_type | volumev3                                |
| url          | http://controller:8776/v3/%(project_id)s |
+--------------+-----------------------------------------+

+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 4511c28a0f9840c78bacb25f10f62c98        |
| interface    | admin                                   |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | ab3bbbef780845a1a283490d281e7fda        |
| service_name | cinderv3                                |
| service_type | volumev3                                |
| url          | http://controller:8776/v3/%(project_id)s |
+--------------+-----------------------------------------+
```

Cuadro 190: Salida esperada del comando de creación de *endpointsendpoints*

Instalación de paquetes de Cinder y configuraciones adicionales a los archivos instalados. Se configura el acceso a la base de datos de cinder en el *controller node* en la sección `[database]`, las configuraciones generales en `[DEFAULT]`, la configuración de Keystone en `[keystone_authtoken]` y finalmente en `[oslo_concurrency]` el directorio que Cinder usa como `lockpath`, es decir, el manejo de archivos que pueden ser accedidos por múltiples procesos o hilos a la vez, y por la naturaleza delicada de su contenido —como la creación o manejo de volúmenes y *snapshots*— se debe asegurar que no exista ningún traslape en su manejo.

```
$ sudo apt install cinder-api cinder-scheduler
$ sudo nano /etc/cinder/cinder.conf
```

Cuadro 191: Instalación y apertura de archivo de configuración de Cinder

```
[database]
# ...
connection = mysql+pymysql://cinder:CINDER_DBPASS@controller/cinder

[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone
my_ip = 192.168.74.149

[keystone_authtoken]
# Remove everything else from this section
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = cinder
password = CINDER_PASS

[oslo_concurrency]
# ...
lock_path = /var/lib/cinder/tmp
```

Cuadro 192: Modificación de archivo de configuración de Cinder

Popular la base de datos del *block storage service*. Este comando cambia al usuario `cinder` para ejecutar el string `cinder-manage db_sync` desde la terminal, el cual cual crea el *schema* de la base de datos de cinder.

```
$ sudo su
$ su -s /bin/sh -c "cinder-manage db sync" cinder
```

Cuadro 193: Configuraciones de la base de datos de Cinder

7.7.2. Instalación en el *Compute1 Node*

Es necesario realizar una modificación en el archivo de configuración de Nova, para coordinar la interoperatividad de estos dos servicios, debido a que Nova hace uso de Cinder para realizar operaciones como:

- Adjuntar (*attach*) y desadjuntar (*detach*) volúmenes a instancias.
- Crear nuevas instancias usando volúmenes como disco raíz.
- Crear *snapshots* de volúmenes conectados a las máquinas virtuales.
- Proporcionar almacenamiento persistente para las instancias, que sigue existiendo aunque la instancia sea eliminada.

El parámetro `os_region_name` especifica la región en la que Nova buscará los servicios de Cinder. Las regiones en OpenStack son agrupaciones lógicas de servicios. Para este caso solo existe la región RegionOne.

```
$ sudo nano /etc/nova/nova.conf
```

Cuadro 194: Apertura de archivo de configuración de Nova

```
[cinder]
# ...
os_region_name = RegionOne
```

Cuadro 195: Modificación de archivo de configuración de Nova

7.7.3. Finalizar instalación en el *Controller Node*

Reiniciar los servicios de Nova y Cinder en el *controller node* para que surtan efecto las modificaciones realizadas.

```
$ sudo service nova-api restart
$ sudo service cinder-scheduler restart
$ sudo service apache2 restart
```

Cuadro 196: Reiniciar servicios modificados

7.7.4. Instalación en el *Storage Node*

NOTA: Para añadir un *block storage node* independiente, referirse a la guía de instalación de OpenStack: *Cinder installation, Yoga release*.

Antes de instalar y configurar el servicio de almacenamiento en bloques en el nodo deseado es necesario realizar algunos pasos previos como preparación. Para este caso, se eligió el mismo *compute1 node* como nodo dedicado a almacenamiento de bloques, por lo que los pasos a continuación se ejecutan en dicho dispositivo.

El primer paso es la instalación de paquetes con herramientas para gestionar volúmenes, lo que permite una gestión flexible de discos al crear y redimensionar particiones sin necesidad de desmontar los sistemas de archivos y para gestionar *thin-provisioning*, una técnica que permite asignar espacio de almacenamiento dinámicamente, utilizando solo el espacio necesario en lugar de reservar todo el almacenamiento por adelantado.

```
$ sudo apt install lvm2 thin-provisioning-tools
```

Cuadro 197: Instalación de paquetes adicionales para Cinder

También es necesario identificar el disco sobre el cual se manejarán los volúmenes, para esto se despliegan todas las unidades de almacenamiento, y se identifica el disco que se desea emplear. Para este caso `sda` es el disco donde está montado el sistema operativo, y `sdb` es el disco que será usado por Cinder.

```
$ ls -l /dev | grep "sd"
$ lsblk
```

Cuadro 198: Visualizar estructura de almacenamiento

```
sda                          8:0    0 931.5G  0 disk
|-sda1                       8:1    0     1G  0 part /boot/efi
|-sda2                       8:2    0     2G  0 part /boot
\-sda3                       8:3    0 928.5G  0 part
    \-ubuntu--vg-ubuntu--lv 253:0  0   100G  0 lvm  /
sdb                          8:16   0  10.9T  0 disk
nvme0n1                    259:0    0   1.9T  0 disk
```

Cuadro 199: Salida esperada del comando de visualización de almacenamiento

Para preparar el disco que manejará Cinder, primero es necesario eliminar todas las particiones existentes, y luego formatearlo para eliminar todos sus metadatos remanentes.

```
$ sudo fdisk /dev/sdb
```

Cuadro 200: Apertura de herramienta de particionado de disco

```
=> d # Eliminar partición
=> w # Escribir los cambios
```

Cuadro 201: Eliminar particiones de disco

```
$ sudo wipefs -a /dev/sdb
```

Cuadro 202: Eliminar metadatos de disco

Una vez la unidad esté completamente vacía, se inicializa la unidad como un volumen físico para ser utilizado por LVM y luego se crea un grupo de volúmenes llamado `cinder-volumes`, el cuál será el espacio de almacenamiento donde se crearán los volúmenes lógicos que Cinder gestionará.

```
$ sudo pvcreate /dev/sdb
$ sudo vgcreate cinder-volumes /dev/sdb
```

Cuadro 203: Preparación de disco para LVM

Por defecto, la herramienta de escaneo de LVM realiza análisis a toda la carpeta `dev/` para verificar si existen dispositivos de almacenamiento en bloques que contengan volúmenes. Si LVM detecta volúmenes ajenos a los discos relevantes pueden existir problemas con Cinder como corrupción de datos o fallos tanto en los volúmenes del sistema como en los gestionados por Cinder. Para evitar esto, se aplica un filtro en LVM para que se lean únicamente los volúmenes de Cinder; sin embargo, debido a que el sistema operativo de la máquina está montado sobre una partición de disco que también es gestionada por LVM, es necesario añadirlo de también a los dispositivos permitidos por el filtro. De esta forma solo estos dos dispositivos son gestionados por LVM y el resto son rechazados.

```
$ sudo nano /etc/lvm/lvm.conf
```

Cuadro 204: Apertura de archivo de configuración de LVM

```
devices {
    ...
    filter = [ "a/sda/", "a/sdb/", "r/.*/"]
```

Cuadro 205: Modificación de filtros de LVM

Instalación de paquetes de Cinder y configuraciones adicionales a los archivos instalados. Para este caso es importante la sección `[DEFAULT]` en `enabled_backends = lvm`, que es la herramienta LVM (*Logical Volume Manager*) del sistema operativo que será utilizada para la gestión de volúmenes. Guardando la consistencia, también se tiene que modificar el apartado de `[lvm]` el cuál es de suma importancia para el correcto funcionamiento de cinder:

- `volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver`: En primer lugar es necesario declarar el controlador que se utilizará para manejar LVM.
- `volume_group = cinder-volumes`: Luego es necesario colocar el nombre del `volume group` creado previamente sobre el disco `/dev/sdb`.
- `target_protocol = iscsi`: Para que las instancias puedan interactuar con los volúmenes como si fueran un sistema de archivos local es necesario emplear un protocolo que se encargue de esta operación, para este caso se usa `iSCSI`.
- `target_helper = tgtadm`: Como apoyo para iSCSI se utiliza `tgtadm`, la cual es una herramienta del iSCSI *target daemon* —tgt— que facilita la administración de objetivos iSCSI.

```
$ sudo apt install cinder-volume tgt
$ sudo nano /etc/cinder/cinder.conf
```

Cuadro 206: Instalación y apertura de archivo de configuración de Cinder

```
[database]
# ...
connection = mysql+pymysql://cinder:CINDER_DBPASS@controller/cinder

[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone
my_ip = 192.168.74.150
enabled_backends = lvm
glance_api_servers = http://controller:9292

[keystone_authtoken]
# Remove everything else from this section
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = cinder
password = CINDER_PASS

[lvm]
# ...
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_group = cinder-volumes
target_protocol = iscsi
target_helper = tgtadm

[oslo_concurrency]
# ...
lock_path = /var/lib/cinder/tmp
```

Cuadro 207: Modificación de archivo de configuración de Cinder

También es necesario realizar una configuración de `tgt`, en la cual se debe mencionar explícitamente al servicio tgtd que debe incluir todos los volúmenes gestionados por Cinder para exponerlos como `targets` iSCSI.

```
$ sudo nano /etc/tgt/targets.conf
```

Cuadro 208: Apertura de archivo de configuración de tgt

```
# ...
include /var/lib/cinder/volumes/*
```

Cuadro 209: Modificación de archivo de configuración de tgt

Finalmente es necesario reiniciar los servicios a los que se les aplicaron modificaciones para que todas entren en efecto.

```
$ sudo service tgt restart
$ sudo service cinder-volume restart
```

Cuadro 210: Reiniciar servicios modificados

7.8. Instalación de Horizon

El *dashboard service* de OpenStack —nombre código "Horizon"— proporciona una interfaz gráfica basada en HTTP que permite a los usuarios y administradores gestionar de manera intuitiva y centralizada los recursos de la nube. A través de Horizon, los usuarios pueden lanzar y administrar instancias de máquinas virtuales, redes, almacenamiento, usuarios y proyectos, entre otros. Este dashboard facilita la interacción con los servicios subyacentes de OpenStack, permitiendo a los usuarios llevar a cabo tareas que de otro modo requerirían comandos de API o CLI, todo desde un entorno accesible y visualmente amigable.

Horizon está compuesto por una serie de paneles y módulos que representan los distintos servicios de OpenStack (como Nova, Neutron, Cinder, etc.) y permite una experiencia de usuario unificada. La arquitectura de Horizon es modular, lo que permite la integración de *plugins* y la personalización de paneles para satisfacer necesidades específicas de despliegues o proyectos.

7.8.1. Instalación en el *Controller Node*

Como pre-requisitos para la instalación, es necesario asegurarse de tener instalado Python en versión mínima 3.6 o 3.7, por defecto debería estar instalado en el sistema, sin embargo se puede verificar rápidamente con el siguiente comando.

```
$ python3 -V
```

Cuadro 211: Verificación de versión de Python

```
Python 3.10.12
```

Cuadro 212: Salida esperada del comando de verificación de Python

También es necesario tener instalado el *framework* Django en versión mínima 3.2, a pesar de que ya hay nuevas versiones disponibles, se recomienda utilizar la versión 3.2 por ser la versión recomendada para Horizon en el release Yoga de OpenStack. Django fué instalado con ayuda de `pip`, por lo que primero se instaló dicho *package manager*.

```
$ sudo apt install python3-pip
$ pip3 install django==3.2
```

Cuadro 213: Instalación de paquetes de Python

Instalación y configuración de componentes de Horizon. Primero es necesario instalar todos los paquetes del *dashboard service*; debido a que al momento de instalar se dieron problemas con las IPs de `archive.ubuntu`, fue necesario ejecutar los comandos `-fix-missing` y `upgrade` para corregirlo.

```
$ sudo apt install openstack-dashboard
$ sudo apt update --fix-missing
$ sudo apt upgrade
```

Cuadro 214: Instalación de paquetes de Horizon

Dentro de la configuración general los parámetros más importantes son las definiciones de los servicios desplegados en OpenStack. Es necesario declarar todos los servicios activos, junto a la versión de su API para que Horizon logre detectarlos de manera adecuada y los integré, logrando así que todos se puedan gestionar desde el dashboard y así aprovechar las ventajas que brinda la interfaz gráfica.

```
$ sudo nano /etc/openstack-dashboard/local_settings.py
```

Cuadro 215: Apertura de archivo de configuración de Horizon

```
# Do not edit the ALLOWED_HOSTS parameter under the Ubuntu
    configuration section.
ALLOWED_HOSTS = ['*']

OPENSTACK_HOST = "controller"

CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.
            MemcachedCache',
        'LOCATION': 'controller:11211',
    }
}

# Comment out any other session storage configuration
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'

OPENSTACK_KEYSTONE_URL = "http://%s:5000/identity/v3" %
    OPENSTACK_HOST
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
    "placement": 1,
    "compute": 2,
    "network": 2,
    "volume": 3,
}
TIME_ZONE = "America/Guatemala"
```

Cuadro 216: Modificación de archivo de configuración de Horizon

```
$ sudo nano /etc/apache2/conf-available/openstack-dashboard.conf
```

Cuadro 217: Apertura de archivo de configuración de Apache

```
WSGIApplicationGroup %{GLOBAL}
```

Cuadro 218: Modificación de archivo de configuración de Apache

Reiniciar los servicios de Apache para que surtan efecto las modificaciones realizadas.

```
$ sudo systemctl reload apache2.service
```

Cuadro 219: Reiniciar servicio de Apache

7.8.2. Verificación de funcionalidad

Acceso a Horizon

Para verificar el funcionamiento, se debe acceder desde un navegador la siguiente URL, puede ser desde cualquier dispositivo pertenciente a la red de laboratorios donde OpenStack fue desplegado. De esta manera se puede acceder al servidor de Apache que alberga el dashboard. Al ingresar las credenciales, se debe poder ingresar a la plataforma.

```
# URL
http://controller/horizon

# Credenciales
Usuario: Admin
Contraseña: onap
Dominio: Default
```

Cuadro 220: Acceso a dashboard de Horizon

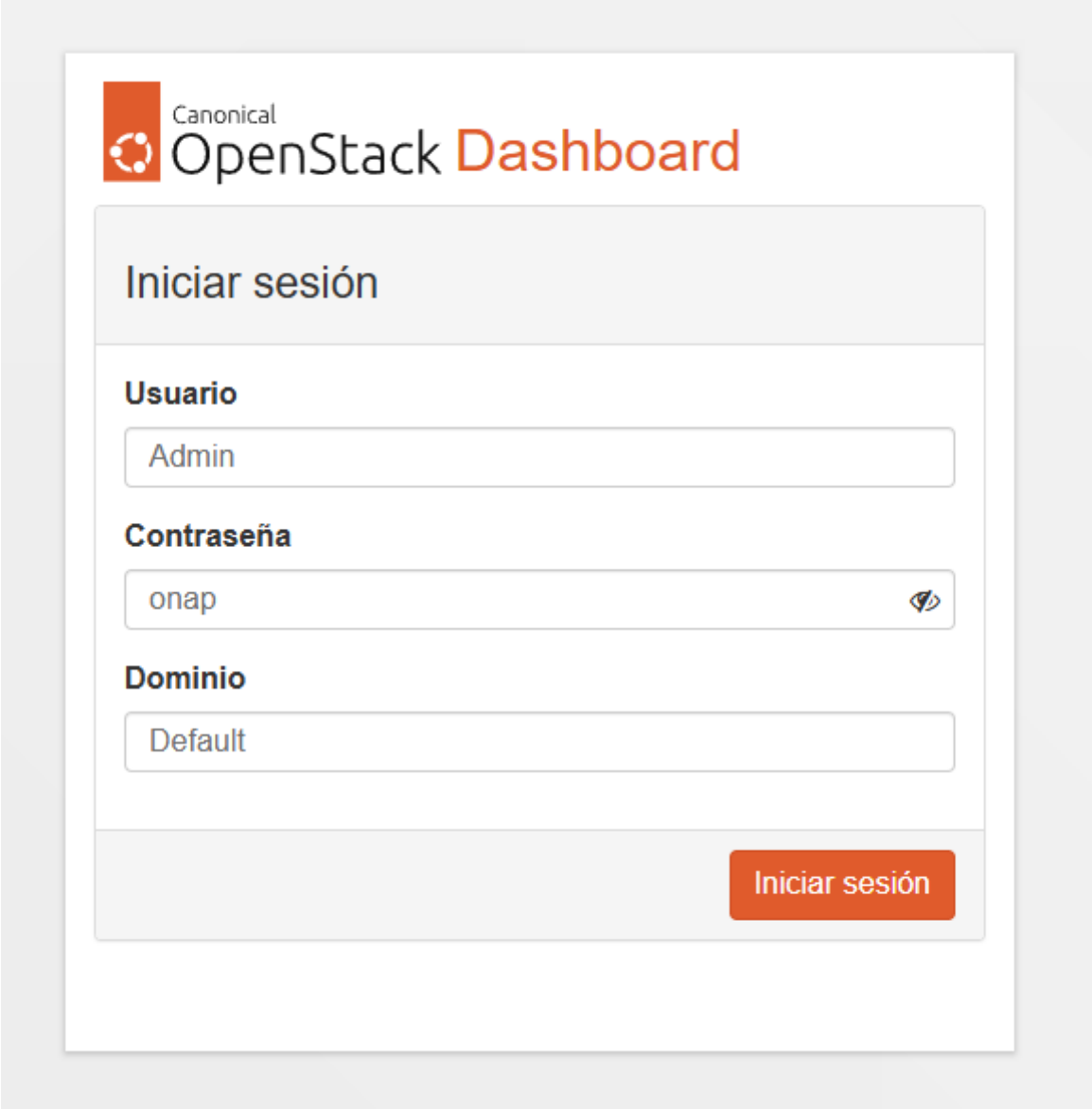

Figura 4: Menú de acceso esperado al ingresar a la página web de Horizon

Añadir imágenes al entorno con Glance

En el menú de navegación de Horizon, seleccionar la opción **Admin** > **Compute** > **Images**. Esta sección permite gestionar las imágenes disponibles en el entorno.

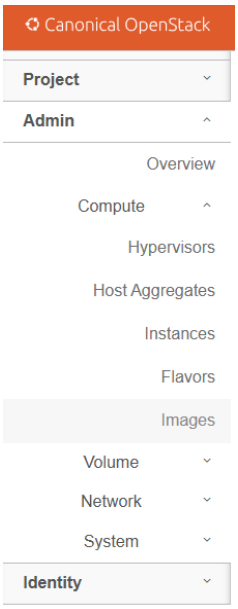

Figura 5: Ubicación del apartado de imágenes en el panel lateral de Horizon

La pantalla muestra todas las imágenes disponibles en el entorno. En esta vista, se pueden observar detalles como el nombre, estado, visibilidad y formato de las imágenes existentes. Para añadir una nueva imagen, seleccionar el botón **Create Image**.

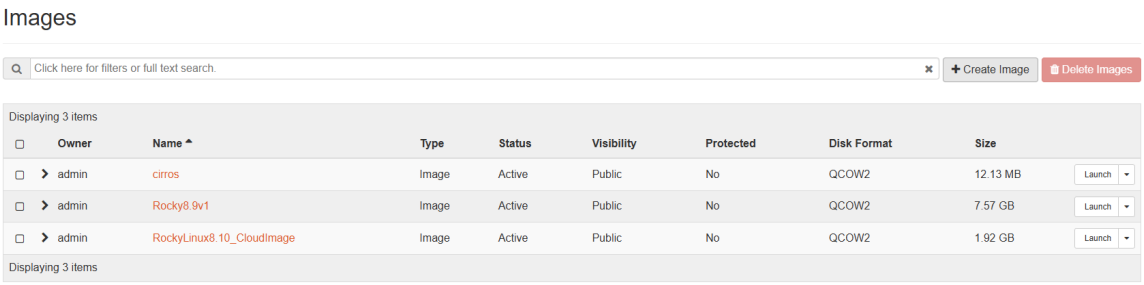

Figura 6: Listado de imágenes y opciones adicionales

Esta es la sección más importante para añadir una imagen al entorno. A continuación, se describen los puntos clave al rellenar este formulario:

- **Image Name**: Proporcionar un nombre descriptivo para identificar la imagen.
- **Image Description** (Opcional): Añadir una breve descripción de la imagen.
- **File**: Seleccionar el archivo de imagen desde el equipo. Asegurarse de que el archivo esté en un formato compatible, como QCOW2.
- **Format**: Seleccionar el formato del disco. Por ejemplo, QCOW2 para discos creados con QEMU/KVM.
- **Visibility**: Configurar la visibilidad de la imagen:

- **Private**: Solo el usuario que la creó puede verla.
- **Public**: Todos los usuarios pueden verla.
- **Shared**: Visible para usuarios específicos.
- **Community**: Visible para la comunidad de usuarios.

Después de completar todos los campos necesarios, hacer clic en **Create Image** para añadir la imagen al entorno.

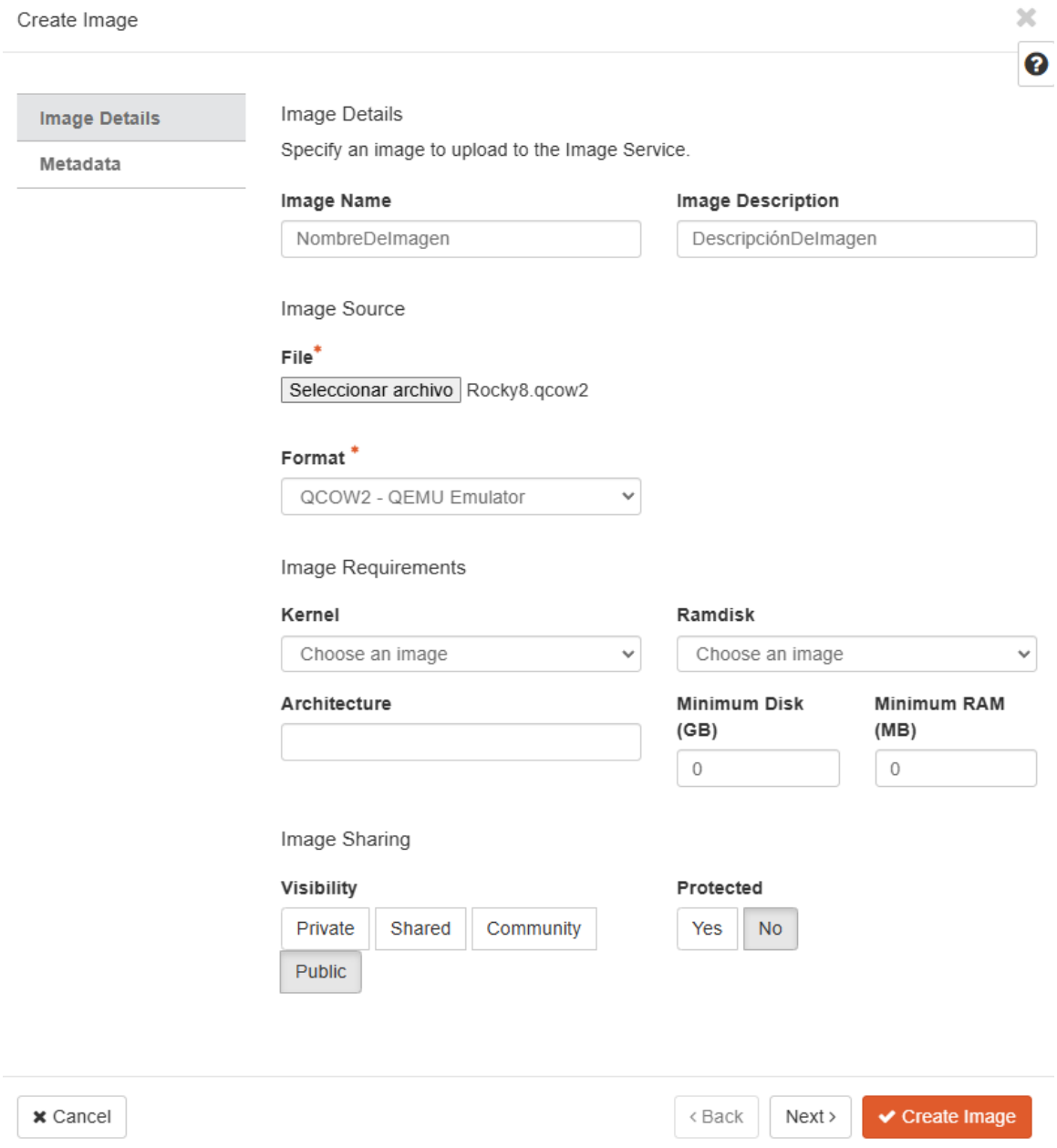

Figura 7: Menú de creación de imágenes

Crear volúmenes con Cinder

En el menú de navegación de Horizon, seleccionar la opción **Project** > **Compute** > **Volumes**. Esta sección permite gestionar los volúmenes disponibles en el entorno.

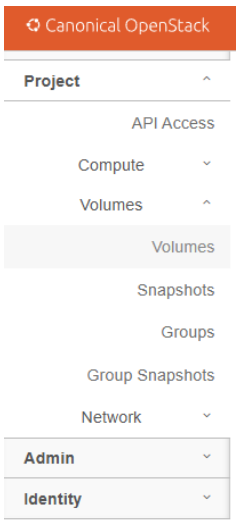

Figura 8: Ubicación del apartado de volúmenes en el panel lateral de Horizon

La pantalla muestra todos los volúmenes disponibles en el entorno. En esta vista, se pueden observar detalles como el nombre, descripción, tamaño, estado y zona de disponibilidad. Para crear un nuevo volumen, seleccionar el botón **Create Volume**.

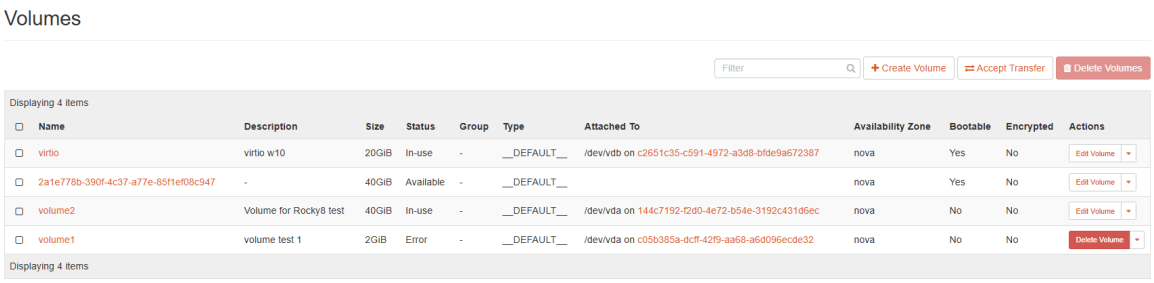

Figura 9: Listado de volúmenes y opciones adicionales

Esta es la sección más importante para crear un volumen en el entorno. A continuación, se describen los puntos clave al rellenar este formulario:

- **Volume Name**: Proporcionar un nombre descriptivo para identificar el volumen.
- **Description** (Opcional): Añadir una breve descripción del volumen.
- **Volume Source**: Seleccionar la fuente del volumen. Puede ser un volumen vacío (*No source, empty volume*) o basado en una imagen, volumen existente o instantánea.
- **Size (GiB)**: Definir el tamaño del volumen en GiB.

Después de completar todos los campos necesarios, hacer clic en **Create Volume** para añadir el volumen al entorno.

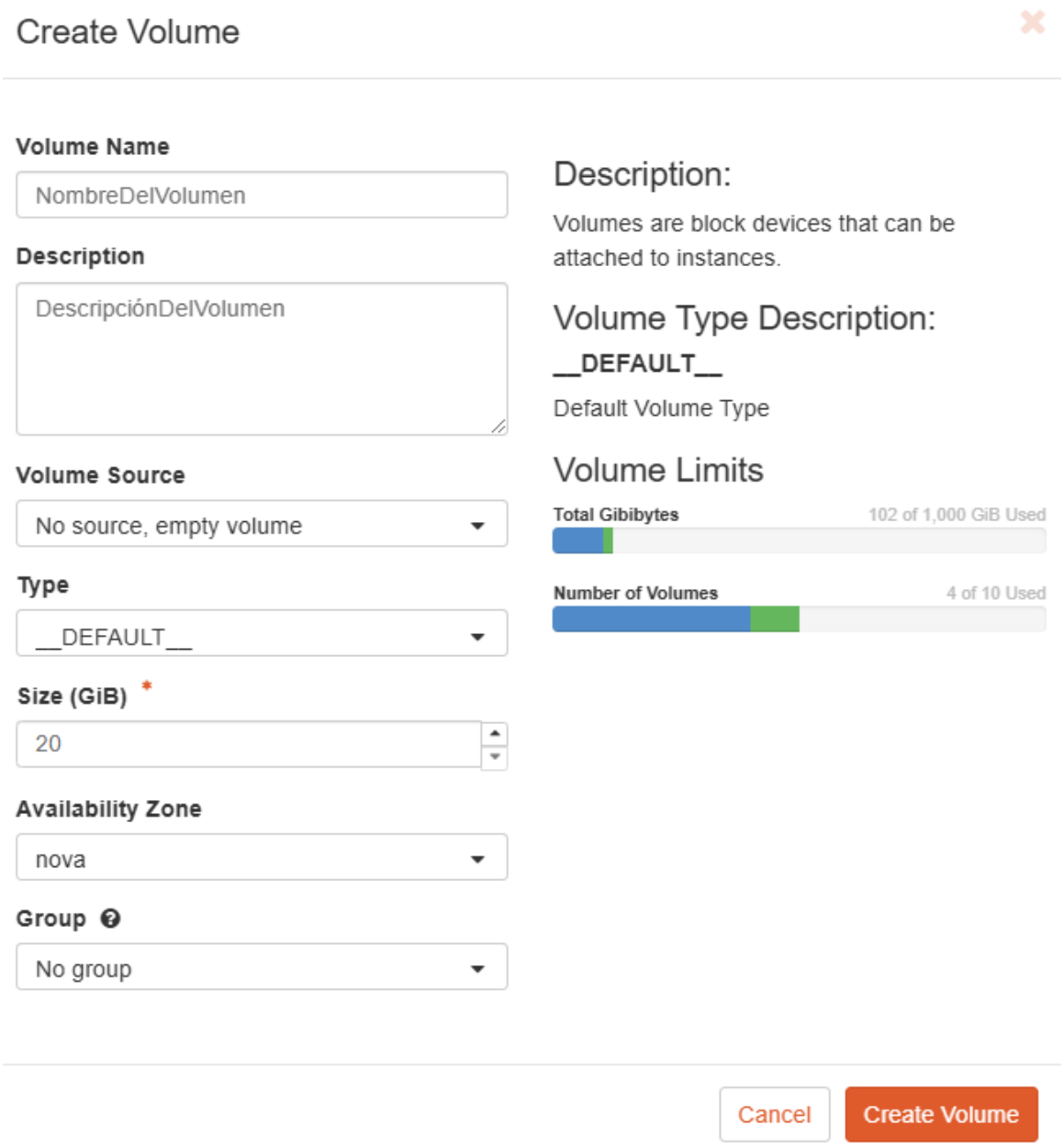

Figura 10: Menú de creación de volúmenes

Crear redes y *routers* virtuales con Neutron

En el menú de navegación de Horizon, seleccionar la opción **Project** > **Network** > **Networks**. Esta sección permite gestionar las redes disponibles en el entorno.

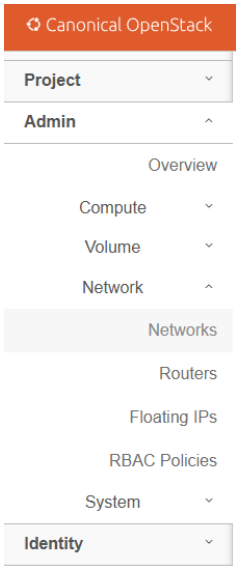

Figura 11: Ubicación del apartado de redes en el panel lateral de Horizon

La pantalla muestra todas las redes disponibles en el entorno. Para crear una nueva red, seleccionar el botón **Create Network**.

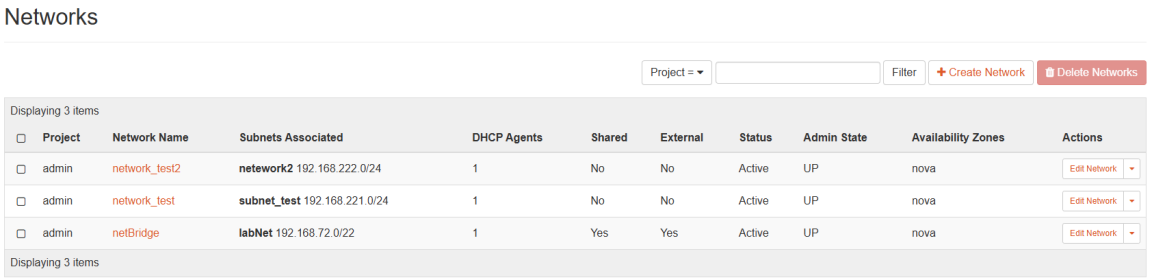

Figura 12: Listado de redes y opciones adicionales

Al crear una red, se deben considerar los siguientes campos en la pestaña **Network**:

- **Name**: Definir un nombre para identificar la red.
- **Project**: Seleccionar el proyecto al que se asociará la red.
- **Provider Network Type**: Especificar el tipo de red. Las posibles opciones son:
  - **Local**: Indica que la red no está asociada a ningún backend físico. Esta opción es adecuada para redes aisladas internas al nodo de computación.
  - **Flat**: Utiliza una red física subyacente existente (*bridge*, tomar en cuenta opciones **Shared** y **External Network**), compartiendo el mismo rango de direcciones IP sin VLAN ni segmentación adicional.
  - **VLAN**: Implementa redes virtuales etiquetadas mediante identificadores VLAN. Esto permite múltiples redes lógicas dentro de la misma infraestructura física, separando el tráfico entre ellas.

    - **VXLAN**: Utiliza un esquema de encapsulación para crear redes virtuales lógicas sobre redes físicas. Es ideal para infraestructuras grandes donde se necesita escalar más allá de las limitaciones de las VLAN tradicionales.

- **Enable Admin State**: Habilitar el estado administrativo.
- **Shared**: Configurar si la red será compartida.
- **External Network**: Marcar si la red será externa.
- **Create Subnet**: Activar para crear una subred asociada.

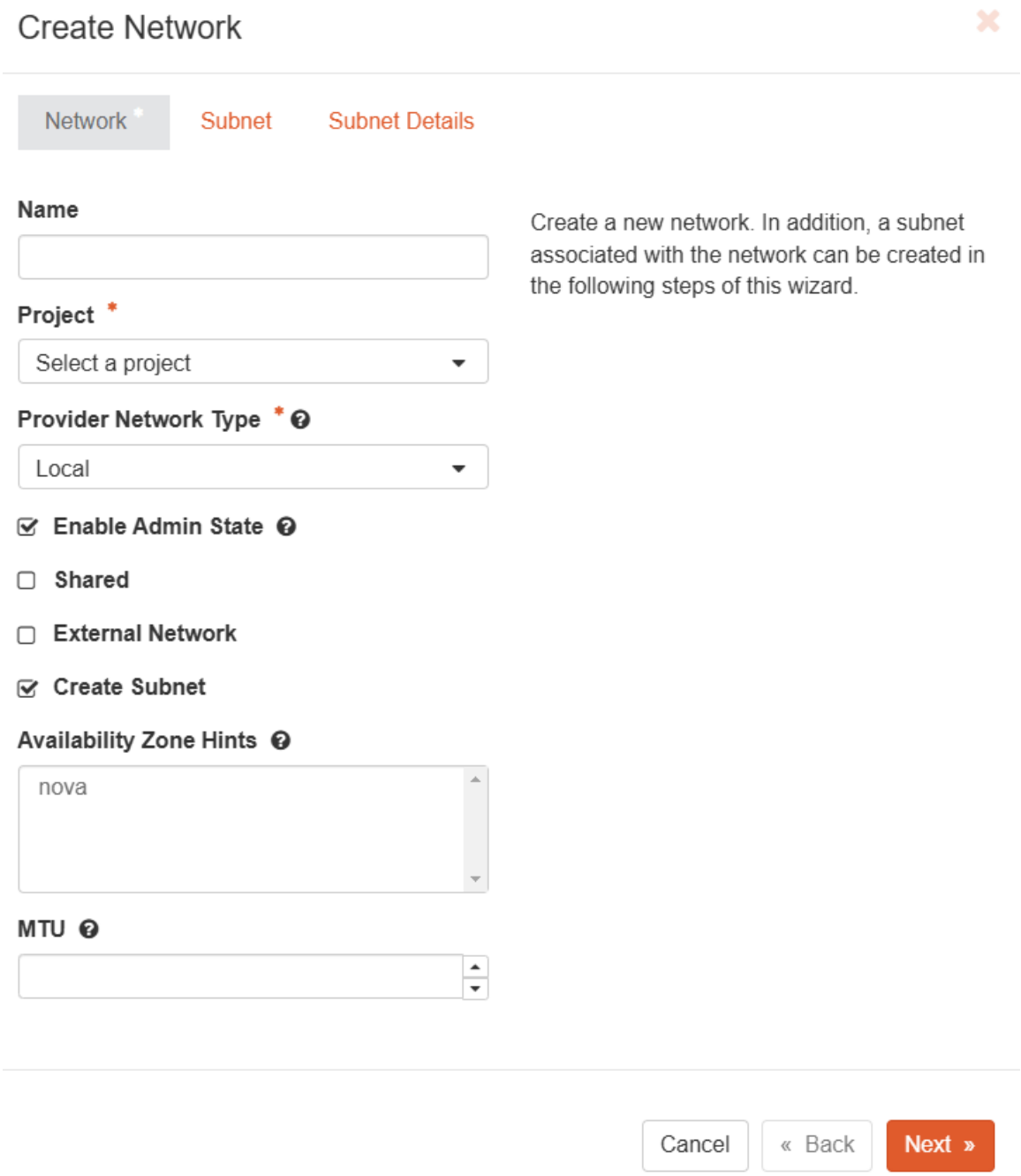

Figura 13: Menú de creación de redes, apartado de opciones generales

En la pestaña **Subnet**, se deben configurar los siguientes campos:

- **Subnet Name**: Definir un nombre para la subred.
- **Network Address**: Especificar la dirección de red en formato CIDR.
- **Gateway IP** (Opcional): Configurar la dirección IP del *gateway*.

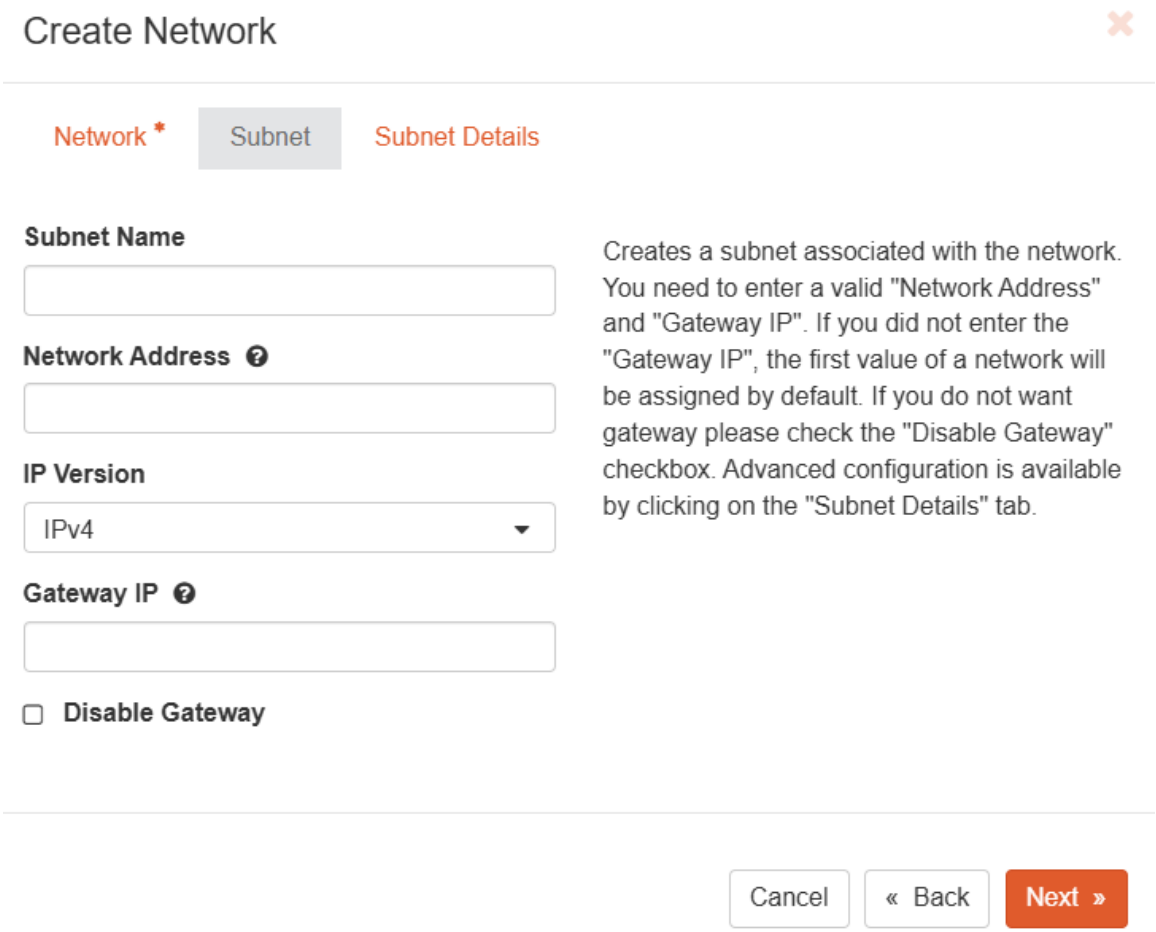

Figura 14: Menú de creación de redes, apartado de subred

En la pestaña **Subnet Details**, se pueden configurar las siguientes opciones avanzadas:

- **Allocation Pools** (Opcional): Definir los rangos de direcciones IP que se asignarán dinámicamente.
- **DNS Name Servers** (Opcional): Especificar los servidores DNS.
- **Host Routes** (Opcional): Configurar rutas estáticas para los hosts.

Al finalizar la configuración, hacer clic en **Create** para crear la red.

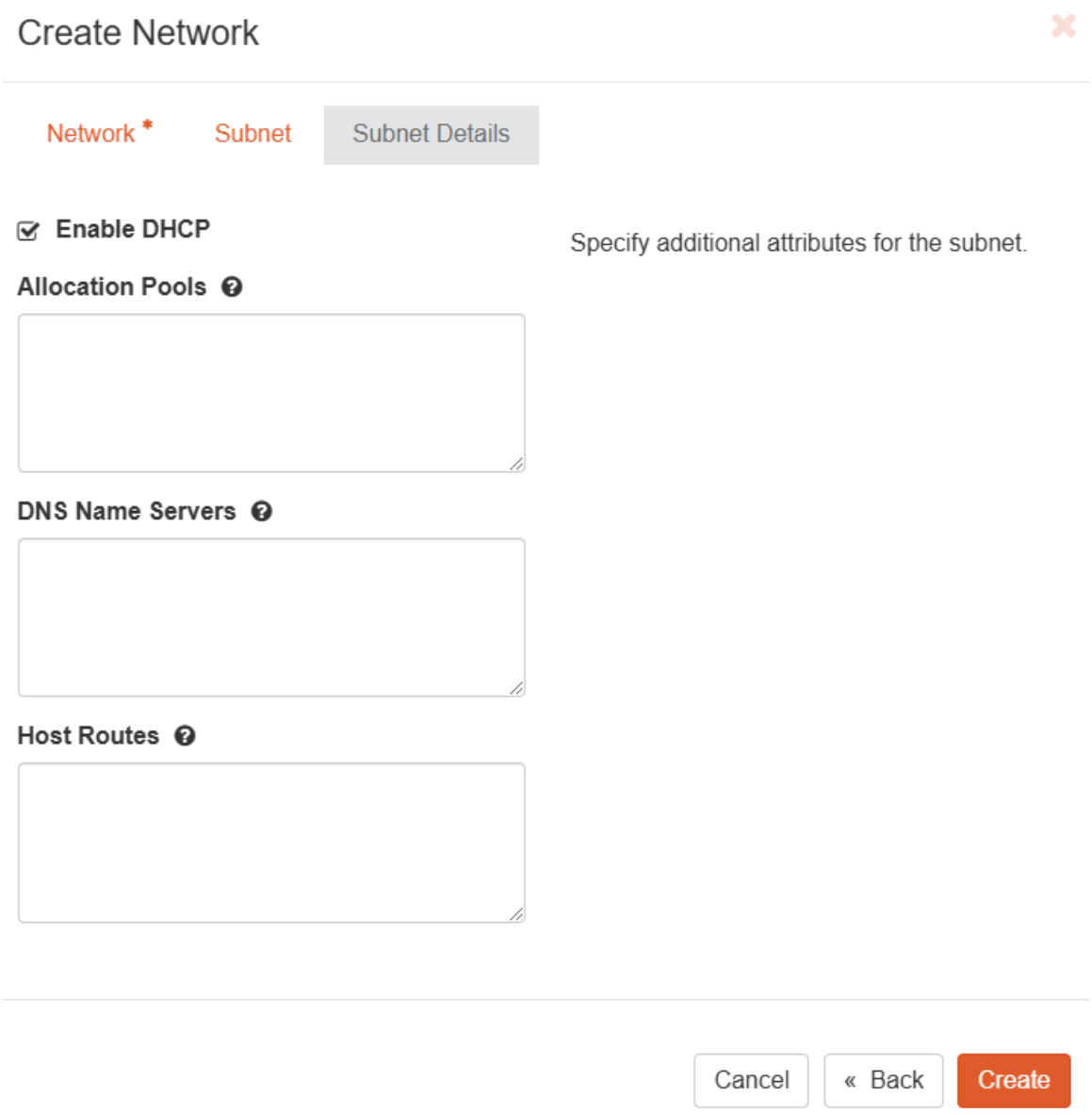

Figura 15: Menú de creación de redes, apartado de opciones avanzadas

En el menú de navegación de Horizon, seleccionar la opción **Project** > **Network** > **Routers**. Esta sección permite gestionar los *routers* disponibles en el entorno.

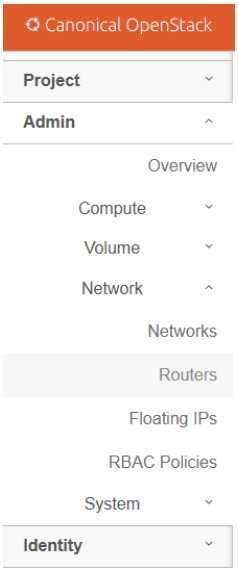

Figura 16: Ubicación del apartado de *routers* en el panel lateral de Horizon

La pantalla muestra todos los *routers* disponibles en el entorno. Para crear un nuevo *router*, seleccionar el botón **Create Router**.

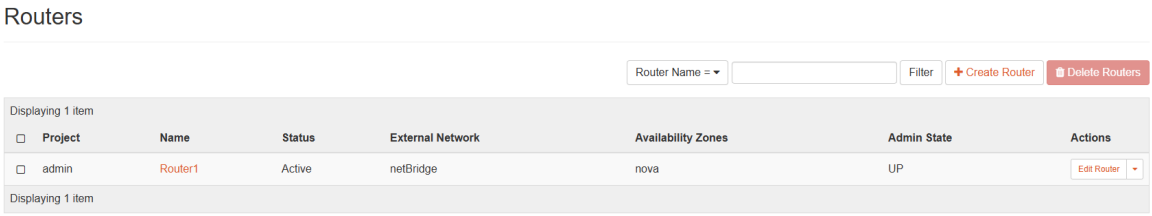

Figura 17: Listado de *routers* y opciones adicionales

En el formulario para crear un *router*, se deben configurar los siguientes campos clave:

- **Router Name**: Definir un nombre para identificar el *router*.
- **External Network**: Seleccionar la red externa asociada.
- **Enable Admin State**: Habilitar el estado administrativo.
- **Enable SNAT**: Activar la traducción de direcciones de red (SNAT).

Al completar todos los campos, hacer clic en **Create Router** para finalizar la creación.

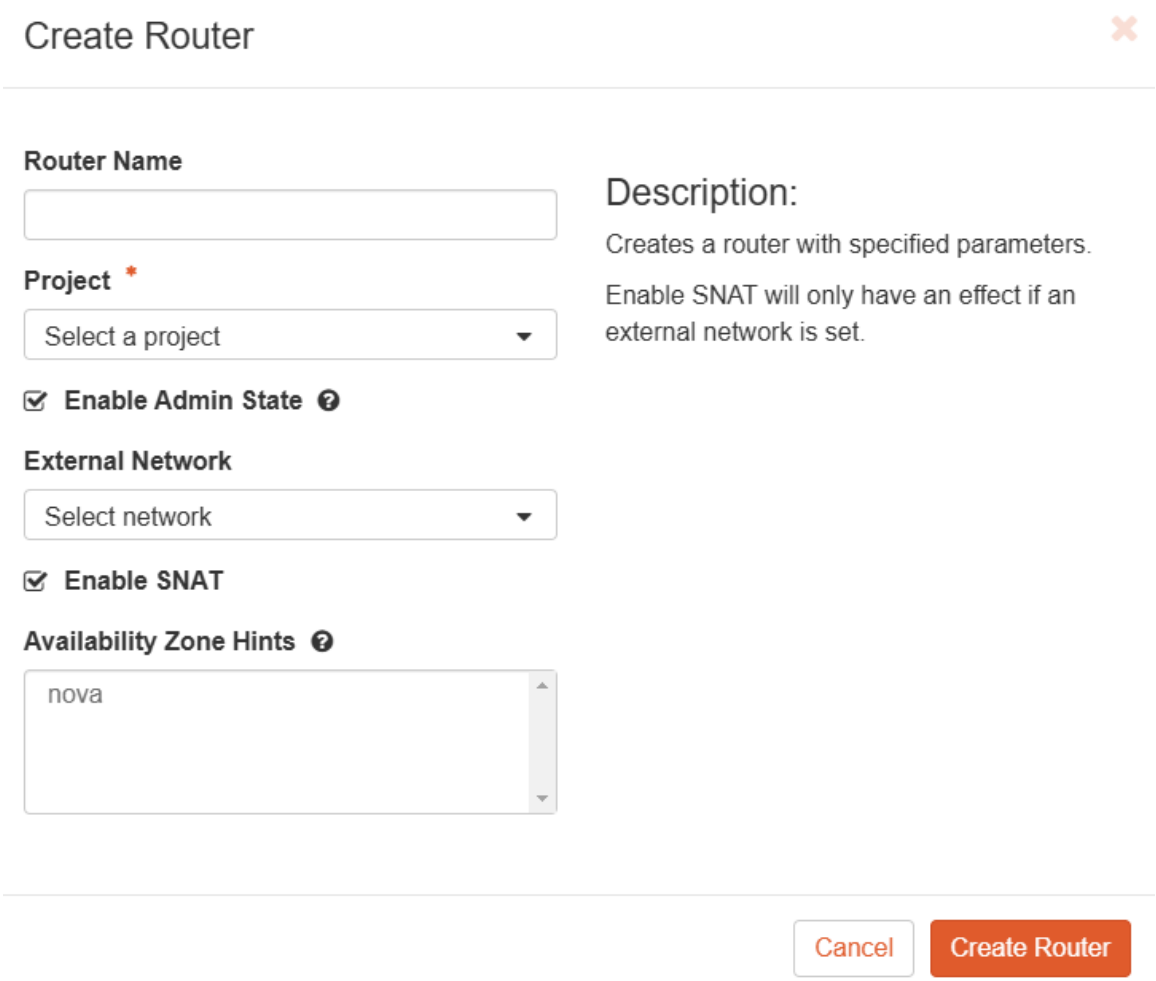

Figura 18: Menú de creación de *routers*

Para permitir la comunicación entre redes y enrutarlas hacia redes externas, es necesario conectar las redes y los *routers*. Este proceso se realiza añadiendo interfaces a los *routers*. En el menú de navegación de Horizon, seleccionar la opción **Project** > **Network** > **Network Topology**. Esta sección proporciona una vista gráfica de las redes, *routers* e instancias en el entorno.

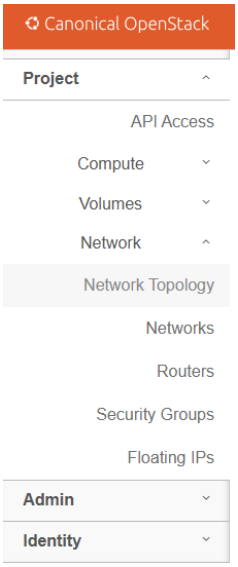

Figura 19: Ubicación del apartado de topología de red en el panel lateral de Horizon

Primero hay que identificar el *router* deseado en la topología, seleccionarlo y presionar la opción **Add Interface**.

- **Subnet**: Seleccionar la subred que se conectará al *router*.
- **IP Address** (opcional): Especificar una dirección IP para la interfaz del *router*. Si no se especifica, se utilizará la dirección IP de la puerta de enlace de la subred.

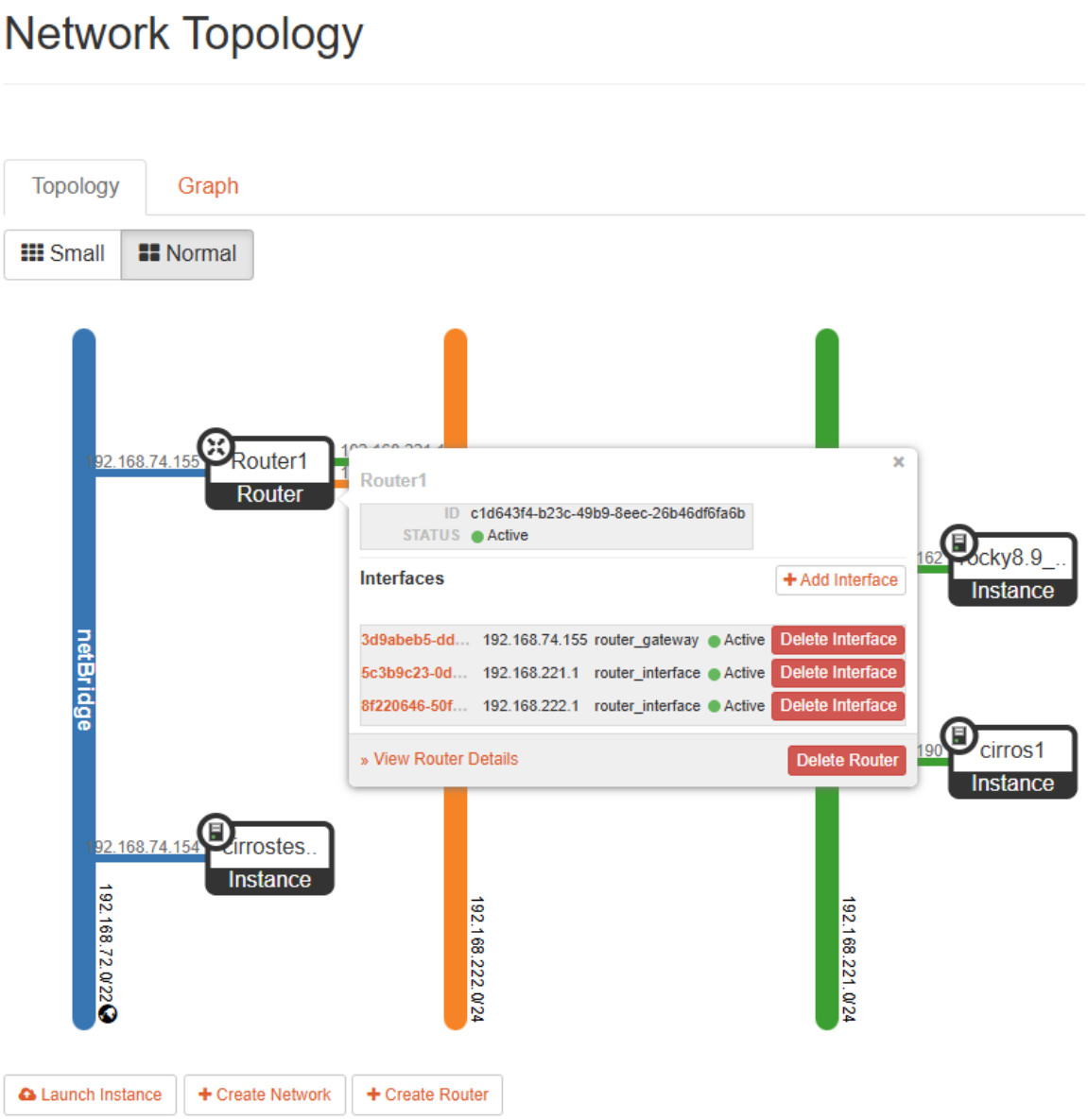

Figura 20: Topología de red activa en Neutron

Para conectar una red a un *router*, es necesario añadir una interfaz al *router*. En el formulario de **Add Interface**, se deben configurar los siguientes campos:

- **Subnet**: Seleccionar la subred que se conectará al *router*.
- **IP Address** (opcional): Especificar una dirección IP para la interfaz del router. Si no se especifica, se utilizará la dirección IP de la puerta de enlace de la subred.

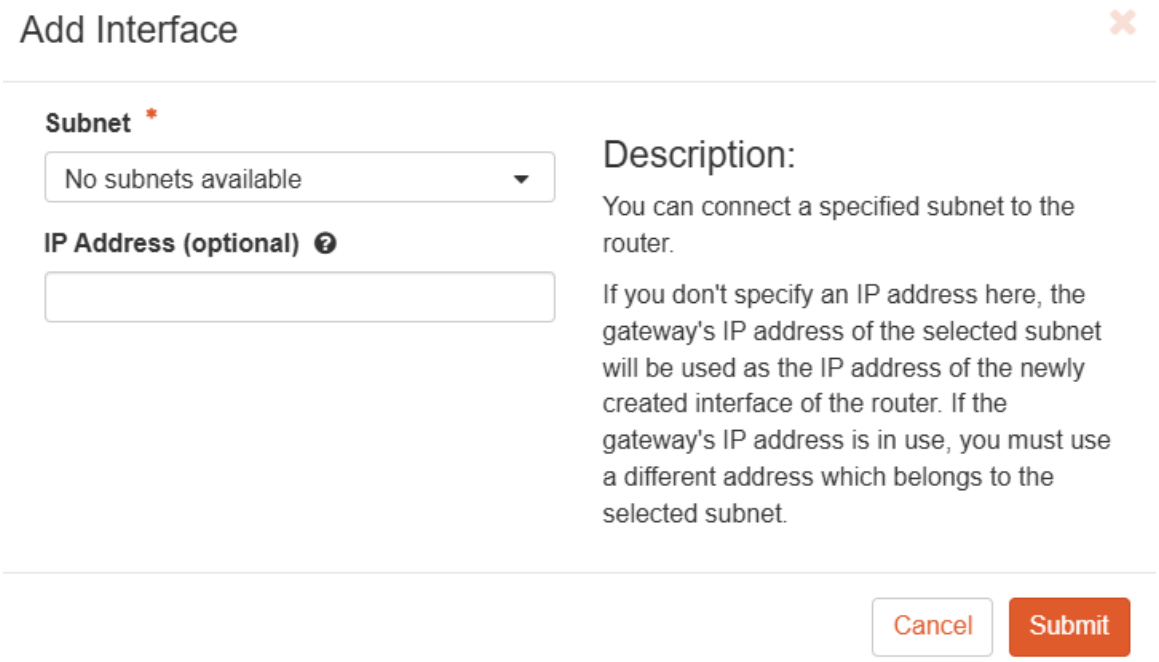

Figura 21: Menú de creación de interfaces en *routers*

Definir sabores para instancias con Nova

Los sabores (*Flavors*) definen las configuraciones de hardware virtualizado, como la cantidad de memoria RAM, núcleos de CPU y tamaño de disco, que se pueden asignar a una instancia. En el menú de navegación de Horizon, seleccionar la opción **Admin** > **Compute** > **Flavors**. Esta sección permite gestionar los sabores disponibles en el entorno.

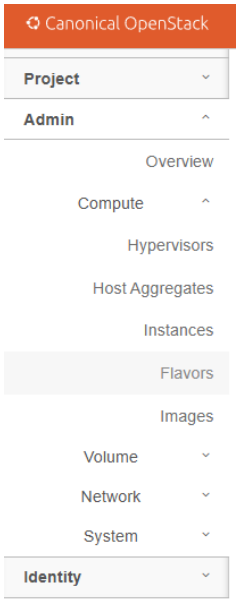

Figura 22: Ubicación del apartado de sabores en el panel lateral de Horizon

La pantalla muestra todos los sabores disponibles en el entorno, incluyendo detalles como el número de núcleos de CPU (*VCPUs*), la memoria RAM, el disco raíz y más. Para crear un nuevo sabor, seleccionar el botón **Create Flavor**.

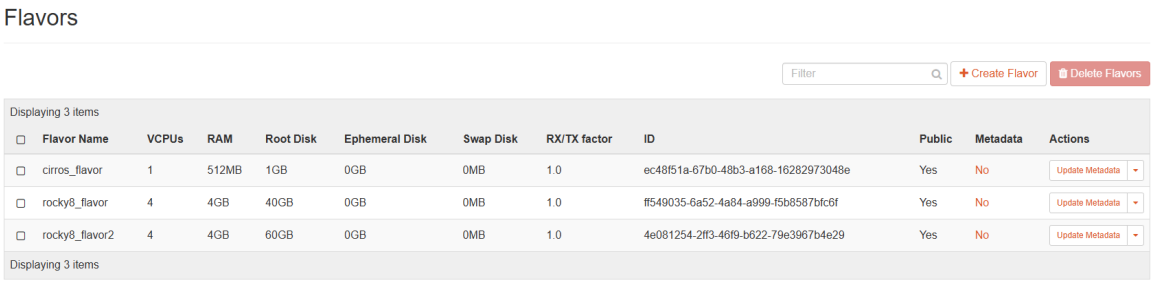

Figura 23: Listado de sabores y opciones adicionales

En el formulario para crear un sabor, se deben configurar los siguientes campos clave:

- **Name**: Proporcionar un nombre descriptivo para identificar el sabor.
- **VCPUs**: Especificar el número de núcleos de CPU que se asignarán a las instancias que utilicen este sabor.
- **RAM (MB)**: Definir la cantidad de memoria RAM (en megabytes) que se asignará.
- **Root Disk (GB)**: Configurar el tamaño del disco raíz (en gigabytes) para las instancias que utilicen este sabor.

Hacer clic en **Create Flavor** para finalizar la creación del sabor.

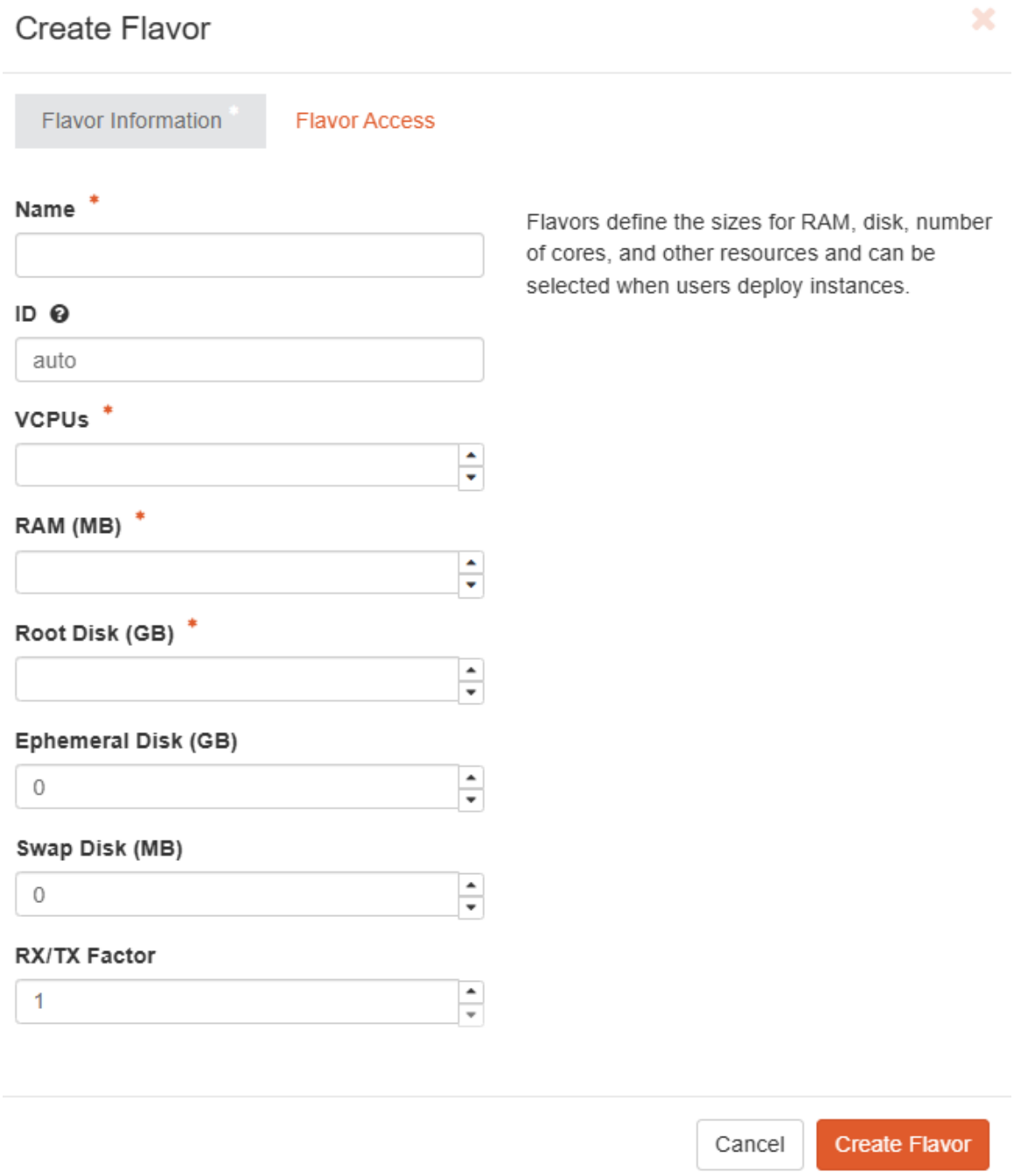

Figura 24: Menú de creación de sabores

Lanzar instancias con Nova

En el menú de navegación de Horizon, seleccionar la opción **Project** > **Compute** > **Instances**. Esta sección permite gestionar las instancias disponibles en el entorno.

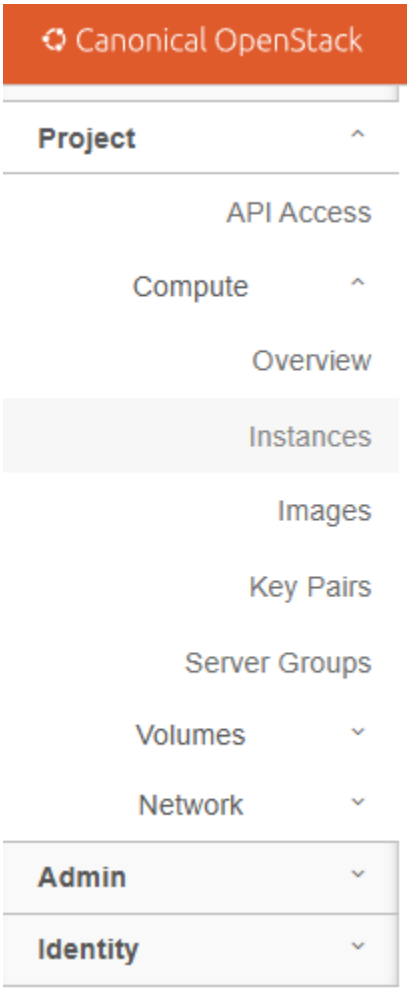

Figura 25: Ubicación del apartado de las instancias en el panel lateral de Horizon

La pantalla muestra todas las instancias disponibles en el entorno. Para crear una nueva instancia, seleccionar el botón **Launch Instance**.

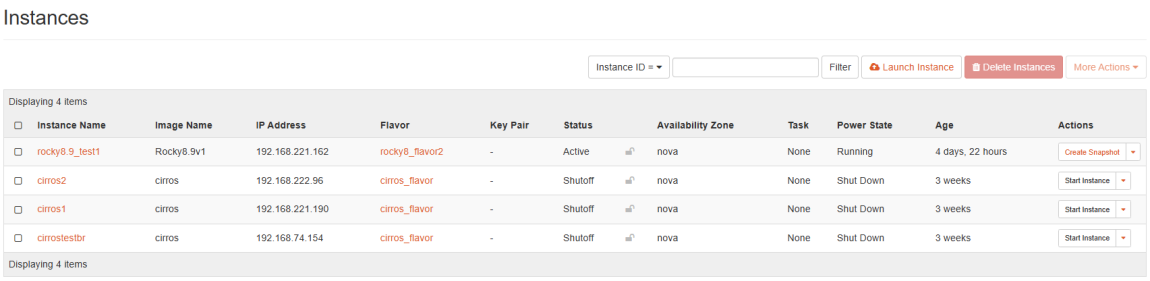

Figura 26: Listado de instancias y opciones adicionales

En la pestaña **Details**, se deben configurar los siguientes campos:

- **Instance Name**: Definir un nombre para identificar la instancia.

- **Description**: (Opcional) Añadir una descripción para la instancia.
- **Count**: Especificar la cantidad de instancias que se desea crear con esta configuración.

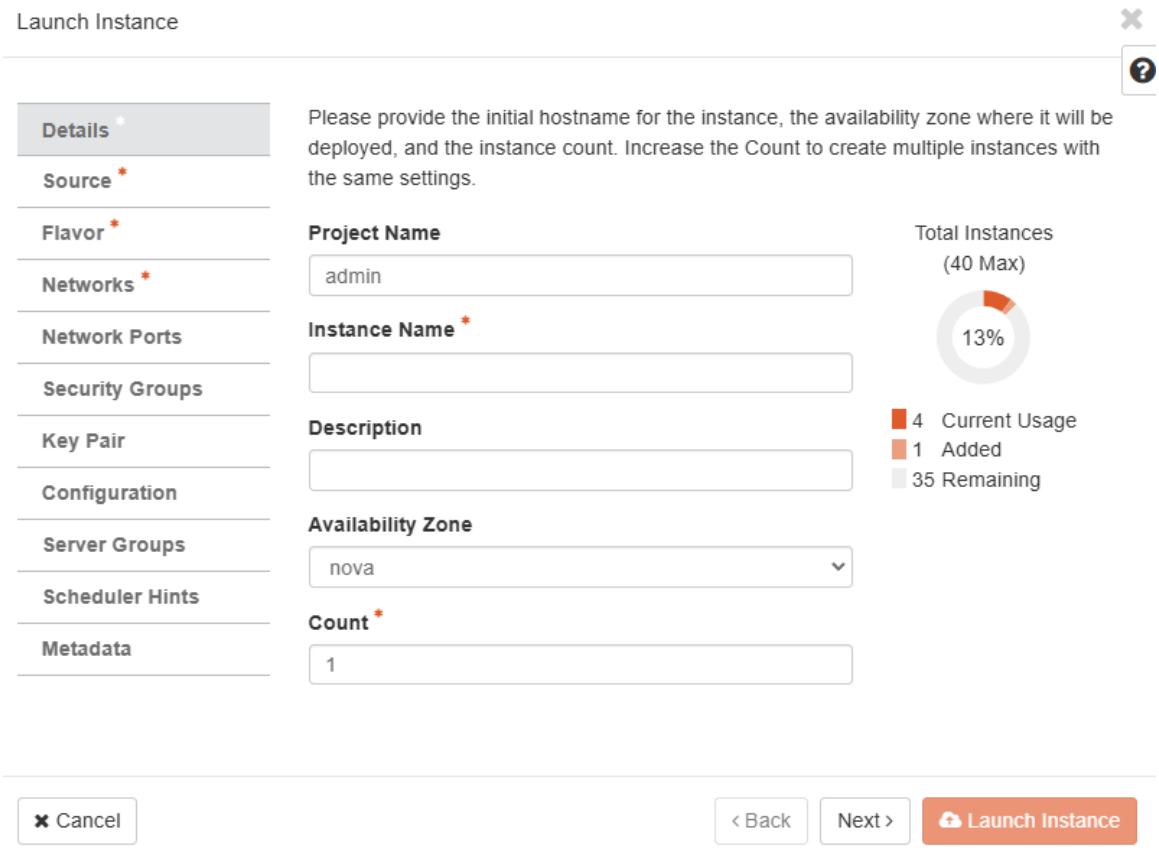

Figura 27: Menú de creación de instancias, detalles generales

En la pestaña **Source**, se deben configurar los siguientes campos:

- **Select Boot Source**: Seleccionar la fuente de arranque de la instancia (por ejemplo, *Image*).
- **Create New Volume**: Especificar si se debe crear un nuevo volumen.
- **Volume Size (GB)**: Definir el tamaño del volumen en gigabytes.
- **Delete Volume on Instance Delete**: Configurar si el volumen debe eliminarse al eliminar la instancia.
- **Image Selection**: Seleccionar la imagen que se usará para la instancia.

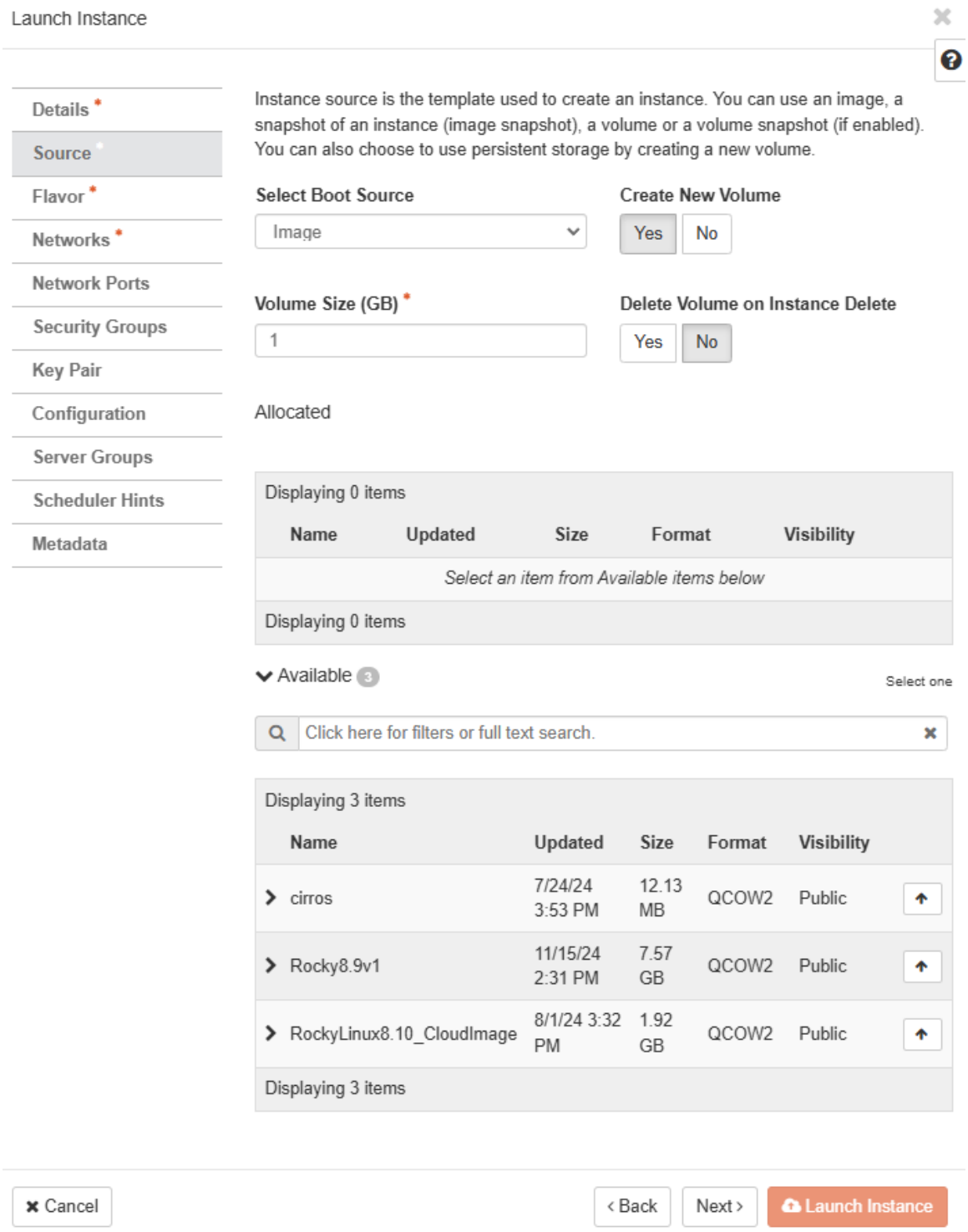

Figura 28: Menú de creación de instancias, fuetnte de imagen y volúmenes

En la pestaña **Flavor**, seleccionar el sabor (*Flavor*) que define la configuración de hardware de la instancia.

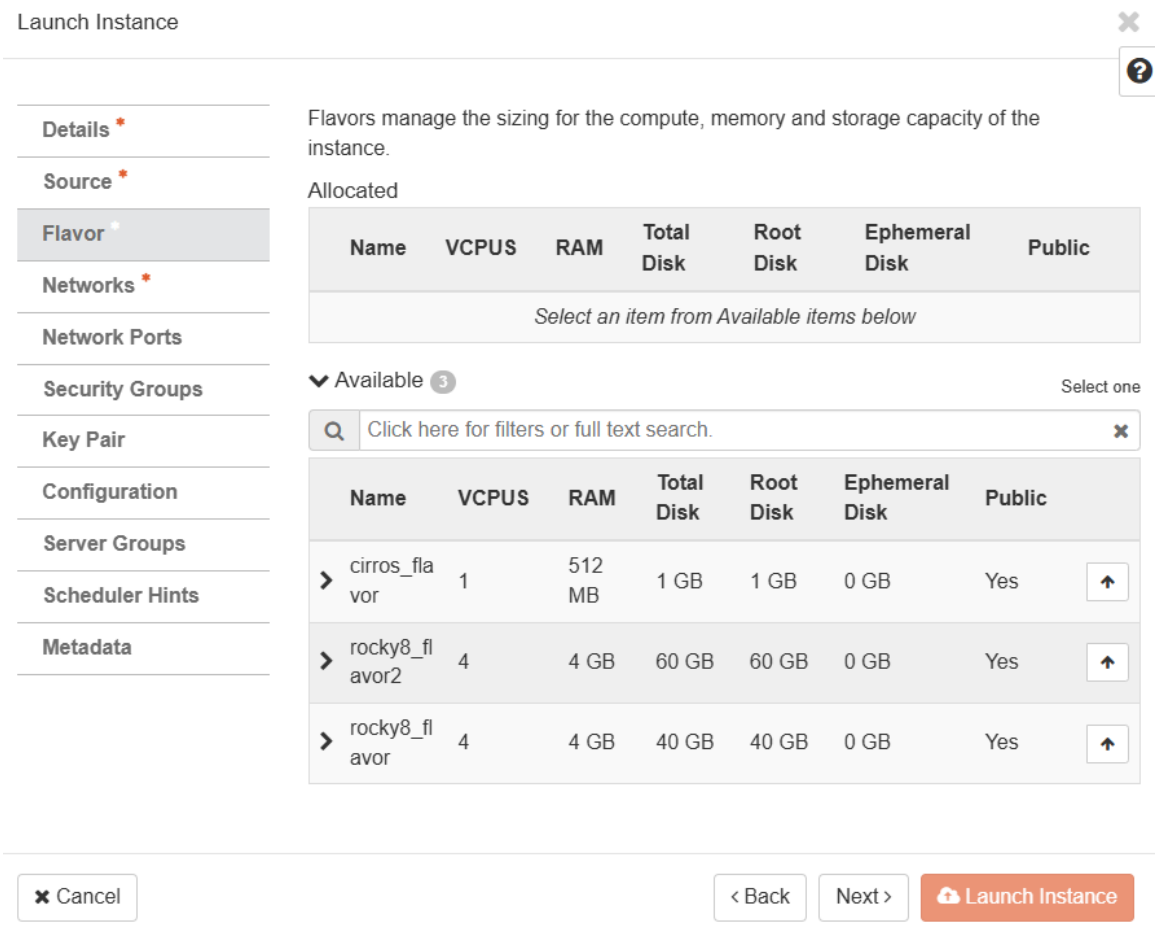

Figura 29: Menú de creación de instancias, sección de sabores

En la pestaña **Networks**, seleccionar las redes disponibles para asignarlas a la instancia. Las redes permiten la comunicación entre la instancia y otros recursos del entorno. Hacer clic en **Launch Instance** para desplegar la instancia con la configuración seleccionada.

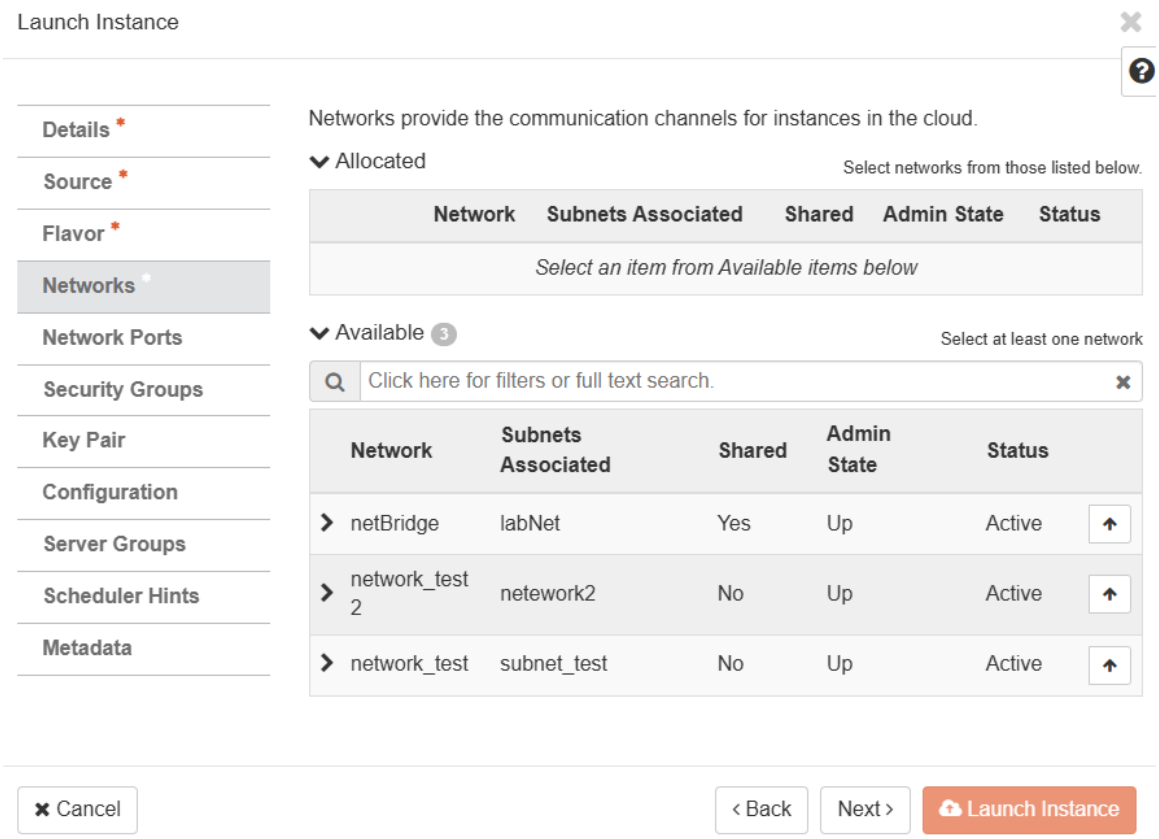

Figura 30: Menú de creación de instancias, configuraciones de red

CAPÍTULO 8

Conclusiones

- Se documentó todo el proceso de instalación, garantizando la replicabilidad y facilitando futuras implementaciones; en la guía se incluye también un apartado de se explica brevemente el uso de OpenStack a través de Horizon.
- Se implementó y configuró el servicio de identidad Keystone, permitiendo la gestión de autenticación y autorización de usuarios.
- Se desplegó el servicio de imágenes Glance para la gestión centralizada de imágenes de máquinas virtuales.
- Se configuró el servicio Placement para optimizar la asignación de recursos físicos.
- Se implementó el servicio de red Neutron, proporcionando conectividad mediante redes virtuales integradas.
- Se configuró el servicio de cómputo Nova, permitiendo la creación y gestión de instancias de máquinas virtuales basadas en imágenes qcow2.
- Se desplegó y configuró el panel de administración web Horizon, facilitando la gestión de los recursos a través de una interfaz intuitiva.
- Se habilitó el servicio de almacenamiento en bloque Cinder, ofreciendo almacenamiento persistente y flexible para las instancias.

CAPÍTULO 9

Recomendaciones

1. Dado que el proyecto se desarrolló como una prueba de concepto, se recomienda implementar mejoras en temas de seguridad y escalabilidad para orientarlo a un ambiente de producción.

2. Integrar más módulos de OpenStack para añadir funcionalidades y capacidades al entorno de computación en la nube implementado en la Universidad del Valle de Guatemala, las cuales podrían ser aprovechadas por otros cursos y proyectos de investigación.

3. Añadir un nodo dedicado para el servicio de almacenamiento en bloque Cinder, liberando esta carga del nodo de computación actual y logrando un despliegue más ordenado de los componentes.

4. Configurar más nodos de cómputo para aumentar los recursos disponibles en OpenStack, permitiendo el despliegue de más instancias de máquinas virtuales y el desarrollo de proyectos más sofisticados y demandantes.

5. Explorar otras opciones de despliegue de OpenStack, como OpenStack-Ansible, Kolla o Helm; para comparar y contrastar las ventajas y desventajas de cada uno.

6. Capacitar al personal técnico de la Universidad del Valle de Guatemala en el uso y administración de OpenStack, asegurando un manejo eficiente de la plataforma y su mantenimiento y actualización en el futuro.

CAPÍTULO 10

Bibliografía

[1] J. León, "Advanced features of the CERN OpenStack Cloud," *EPJ Web of Conferences*, vol. 214, pág. 07026, ene. de 2019. DOI: 10.1051/epjconf/201921407026.

[2] Q. Fu y S.-A. Yu, "China Mobile and Ericsson collaborating in automation and CI/CD for network cloud deployment," sep. de 2020. dirección: https://www.ericsson.com/en/blog/2020/9/china-mobile-and-ericsson-collaborating-in-automation-and-cicd-for-network-cloud-deployment.

[3] E. Pincus, "The rise of cloud computing: Trends and predictions," nov. de 2023. dirección: https://www.infosecinstitute.com/resources/cloud/the-rise-of-cloud-computing-trends-and-predictions/.

[4] I. Ahmad, "Cloud Computing - A Comprehensive Definiton," *Journal of Computing and Management Studies*, vol. 1, ene. de 2017. dirección: https://www.researchgate.net/publication/314072571_Cloud_Computing_-_A_Comprehensive_Definiton.

[5] N. Khan, N. Ahmad, T. Herawan y M. Mat Deris, "Cloud Computing: Analysis of Various Services," sep. de 2012, ISBN: 978-3-642-34061-1. DOI: 10.1007/978-3-642-34062-8_52. dirección: https://www.researchgate.net/publication/228067930_Cloud_Computing_Analysis_of_Various_Services.

[6] A. G. Prajapati, S. J. Sharma y V. S. Badgujar, "All About Cloud: A Systematic Survey," en *2018 International Conference on Smart City and Emerging Technology (ICSCET)*, 2018, págs. 1-6. DOI: 10.1109/ICSCET.2018.8537277.

[7] K. Kaur, "A Review of Cloud Computing Service Models," *International Journal of Computer Applications*, vol. 140, n.º 7, mar. de 2016, ISSN: 0975-8887. dirección: https://typeset.io/pdf/a-review-of-cloud-computing-service-models-4ml94drnyu.pdf.

[8] I. Odun-Ayo, M. Ananya, F. Agono y R. Goddy-Worlu, "Cloud Computing Architecture: A Critical Analysis," en *2018 18th International Conference on Computational Science and Applications (ICCSA)*, 2018, págs. 1-7. DOI: 10.1109/ICCSA.2018.8439638.

[9] M. A. Nadeem y T. Karamat, "A survey of cloud network overlay protocols," en *2016 Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, 2016, págs. 177-182. DOI: 10.1109/DICTAP.2016.7544023.

[10] "IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks," *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*, págs. 1-1993, 2018. DOI: 10.1109/IEEESTD.2018.8403927.

[11] M. Mahalingam, D. Dutt, K. Duda et al., *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*, RFC 7348, ago. de 2014. DOI: 10.17487/RFC7348. dirección: https://www.rfc-editor.org/info/rfc7348.

[12] A. Alnaim, A. Alwakeel y E. Fernández, "A Pattern for an NFV Virtual Machine Environment," abr. de 2019, págs. 1-6. DOI: 10.1109/SYSCON.2019.8836847. dirección: https://www.researchgate.net/publication/335866538_A_Pattern_for_an_NFV_Virtual_Machine_Environment.

[13] J. K. Meena y R. Kumar Banyal, "Efficient Virtualization in Cloud Computing," en *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, 2021, págs. 227-232. DOI: 10.1109/ICCMC51019.2021.9418425.

[14] Y. Li, W. Li y C. Jiang, "A Survey of Virtual Machine System: Current Technology and Future Trends," *Electronic Commerce and Security, International Symposium*, vol. 0, págs. 332-336, jul. de 2010. DOI: 10.1109/ISECS.2010.80. dirección: https://www.researchgate.net/publication/232625032_A_Survey_of_Virtual_Machine_System_Current_Technology_and_Future_Trends.

[15] O. contributors, *OpenStack Install Guide*, OpenInfra Foundation Project, 2024. dirección: https://docs.openstack.org//install-guide/InstallGuide.pdf.

[16] OpenStack, *Keystone Documentation Release 21.0.1.dev13*, OpenInfra Foundation Project, 2023. dirección: https://docs.openstack.org//keystone/yoga/doc-keystone.pdf.

[17] OpenStack, *Glance Documentation Release 24.2.1.dev1*, OpenInfra Foundation Project, 2019. dirección: https://docs.openstack.org/glance/yoga/install/.

[18] O. Fundation, *Placement Documentation Release 7.0.1.dev3*, OpenInfra Foundation Project, 2022. dirección: https://docs.openstack.org//placement/yoga/doc-placement.pdf.

[19] N. development team, *Neutron Documentation Release 20.5.1.dev23*, OpenInfra Foundation Project, 2024. dirección: https://docs.openstack.org//neutron/yoga/doc-neutron.pdf.

[20] O. Fundation, *Nova Documentation Release 25.2.2.dev5*, OpenInfra Foundation Project, 2023. dirección: https://docs.openstack.org//nova/yoga/doc-nova.pdf.

[21] O. Fundation, *Horizon Documentation Release 22.1.1.dev42*, OpenInfra Foundation Project, 2023. dirección: https://docs.openstack.org//horizon/yoga/doc-horizon.pdf.

[22] C. Contributors, *Cinder Documentation Release 20.3.2.dev3*, OpenInfra Foundation Project, 2023. dirección: https://docs.openstack.org//cinder/yoga/doc-cinder.pdf.

CAPÍTULO 11

Glosario

AMQP: Protocolo avanzado de cola de mensajes (*Advanced Message Queuing Protocol*) diseñado para la comunicación entre aplicaciones distribuidas . 88

Apache: Servidor HTTP de código abierto utilizado para alojar aplicaciones web y manejar solicitudes HTTP . 88

API: Interfaz de programación de aplicaciones; conjunto de reglas y especificaciones que permiten la comunicación entre aplicaciones . 88

Backend: Parte de un sistema o aplicación que maneja la lógica, el almacenamiento y las operaciones internas no visibles para el usuario . 88

Bare-metal: Servidor físico; servidor que no tiene una capa de virtualización entre el hardware y el sistema operativo. . 88

Block storage: Tipo de almacenamiento que divide los datos en bloques individuales y los gestiona como unidades separadas en discos virtuales o físicos . 88

Bootstrap: Proceso inicial para configurar y preparar un sistema o servicio antes de que esté completamente operativo . 88

Bridge: Dispositivo o software de red que conecta múltiples redes para que funcionen como una sola . 88

Capa 2: Nivel de enlace de datos en el modelo OSI que gestiona la transferencia de datos entre dispositivos en la misma red . 88

Capa 3: Nivel de red en el modelo OSI que se encarga del enrutamiento de datos entre redes diferentes . 88

Cinder: Nombre del servicio de almacenamiento en bloques de OpenStack . 88

Cirros: Sistema operativo ligero diseñado para pruebas y desarrollo . 88

CLI: Interfaz de Línea de Comandos (*Command Line Interface*) que permite interactuar con un sistema a través de comandos escritos . 88

Cloud computing: Computación en la nube; acceso bajo demanda a recursos de computación alojados en servidores accesibles a través de la red . 88

Daemon: Proceso que se ejecuta en segundo plano para gestionar tareas específicas dentro de un sistema operativo o aplicación . 88

DHCP: Protocolo de configuración dinámica de host (*Dynamic Host Configuration Protocol*) utilizado para asignar automáticamente direcciones IP y otros parámetros de red . 88

Django: *Framework* de desarrollo web de alto nivel basado en Python, diseñado para facilitar la creación de aplicaciones web seguras y escalables . 88

DNS: Sistema de nombres de dominio (*Domain Name System*) que traduce nombres de dominio legibles por humanos en direcciones IP . 88

Driver: Componente de software que permite que un sistema operativo o aplicación interactúe con un hardware o servicio específico . 88

Endpoint: Dirección específica donde un servicio o recurso puede ser accedido por un cliente . 88

Estrato: Nivel jerárquico dentro de la arquitectura de NTP que indica la proximidad de un servidor al reloj de referencia principal . 88

Etcd: Almacén de claves-valor distribuido diseñado para proporcionar configuración compartida y sincronización de servicios . 88

Fernet: Mecanismo de autenticación basado en tokens simétricos, que garantiza la seguridad y la portabilidad mediante el cifrado y la firma de datos . 88

File system: Método de organización y gestión de datos en un medio de almacenamiento, estructurado en archivos y directorios . 88

Firewall: Sistema de seguridad de red que supervisa y controla el tráfico entrante y saliente según políticas de seguridad predefinidas . 88

Framework: Conjunto estructurado de herramientas y bibliotecas que facilita el desarrollo y la implementación de aplicaciones . 88

Glance: Nombre del servicio de imágenes de OpenStack . 88

GUI: Interfaz Gráfica de Usuario (*Graphical User Interface*), diseñada para facilitar la interacción visual con sistemas o aplicaciones . 88

Hipervisor: Software que permite la ejecución de máquinas virtuales sobre una máquina anfitriona . 88

Host: Máquina física o virtual que actúa como anfitrión para sistemas operativos o aplicaciones . 88

HPC: Computadora de alto rendimiento; computadora con altos recursos de computación y almacenamiento que la hacen ideal para tareas demandantes de recursos de cómputo . 88

HTTP: Protocolo de transferencia de hipertexto (*Hypertext Transfer Protocol*) utilizado para la comunicación en la web . 88

IaaS: Infraestructura como servicio; modelo de servicio de la nube que provee control a los usuarios sobre el uso de recursos virtualizados alojados en la nube . 88

Ironic: Servicio de OpenStack que facilita la gestión de hardware *bare-metal*, permitiendo el aprovisionamiento directo de servidores físicos . 88

iSCSI: Protocolo que transporta comandos SCSI a través de redes IP para acceder a dispositivos de almacenamiento remoto . 88

Key-value: Modelo de almacenamiento de datos que organiza la información en pares de clave y valor . 88

Keystone: Nombre del servicio de identidad de OpenStack . 88

KVM: Máquina Virtual basada en Kernel (*Kernel-based Virtual Machine*), un hipervisor de código abierto que permite ejecutar múltiples máquinas virtuales en sistemas Linux . 88

L2 population: Mecanismo en OpenStack que optimiza las redes VXLAN al distribuir información de capa 2 de manera eficiente . 88

LTS: Versión de software con soporte a largo plazo (*Long Term Support*), enfocada en estabilidad y actualizaciones prolongadas . 88

MariaDB: Sistema de gestión de bases de datos de código abierto derivado de MySQL, conocido por su compatibilidad, rendimiento y flexibilidad . 88

Memcached: Sistema de almacenamiento en memoria que se utiliza para acelerar aplicaciones mediante la reducción de accesos a bases de datos . 88

Message queue: Sistema de mensajería que permite la comunicación asincrónica entre servicios a través de colas de mensajes . 88

Middleware: Software que actúa como intermediario para permitir la comunicación y gestión de datos entre aplicaciones o servicios . 88

MySQL: Sistema de gestión de bases de datos relacional ampliamente utilizado para almacenar y gestionar datos estructurados . 88

NAT: Traducción de direcciones de red (*Network Address Translation*) que permite que varios dispositivos compartan una única dirección IP pública . 88

Neutron: Nombre del servicio de manejo de redes de OpenStack . 88

Nodo: Unidad dentro de una infraestructura que puede ejecutar procesos o albergar servicios específicos . 88

Nova: Nombre del servicio de instancias y cómputo de OpenStack . 88

NTP: Protocolo de tiempo de red (*Network Time Protocol*) utilizado para sincronizar los relojes de los sistemas dentro de una red . 88

Object storage: Modelo de almacenamiento de datos que organiza la información en objetos, cada uno con metadatos y una identificación única . 88

OpenStack: Plataforma de código abierto que permite la creación y gestión de ambientes de nube . 88

Oslo: Conjunto de bibliotecas compartidas y herramientas de OpenStack diseñadas para estandarizar y simplificar el desarrollo de sus componentes . 88

PaaS: Plataforma como servicio; modelo de servicio de la nube que provee control a los usuarios sobre una plataforma de desarrollo alojada en la nube . 88

Partición: División lógica de un disco físico que actúa como una unidad de almacenamiento independiente . 88

Placement: Nombre del servicio de rastreo de recursos de OpenStack . 88

Plugin: Componente adicional que amplía las funcionalidades de un sistema o aplicación sin modificar su núcleo . 88

Promiscuous mode: Configuración de red en la que una interfaz de red captura y procesa todo el tráfico que pasa por ella, independientemente del destinatario . 88

Proxy: Servidor intermediario que actúa como puente entre un cliente y un servidor final para procesar solicitudes . 88

QEMU: Emulador y virtualizador de código abierto que permite ejecutar sistemas operativos y programas para arquitecturas diferentes en un único hardware . 88

RabbitMQ: Software de mensajería de código abierto basado en el protocolo AMQP, utilizado para gestionar colas de mensajes entre servicios . 88

Request: Solicitud enviada desde un cliente a un servidor para ejecutar una acción o recuperar información . 88

Router: Dispositivo de red que dirige el tráfico entre diferentes redes, facilitando la comunicación entre ellas . 88

RPC: Llamada a procedimiento remoto (*Remote Procedure Call*) que permite ejecutar funciones en un servidor remoto . 88

SaaS: Software como servicio; modelo de servicio de la nube que provee control a los usuarios sobre software alojado en la nube . 88

Schema: Estructura lógica que define la organización y las relaciones de los datos en una base de datos . 88

Self-service: Modelo de red en OpenStack que permite a los usuarios crear y gestionar redes aisladas de manera autónoma . 88

Shared nothing: Arquitectura en la que cada nodo de un sistema tiene sus propios recursos independientes, sin compartir memoria ni almacenamiento . 88

Snapshot: Copia puntual del estado de una máquina virtual o sistema de almacenamiento que permite restaurarlo posteriormente . 88

SQL: Lenguaje de consulta estructurado (*Structured Query Language*) utilizado para gestionar y manipular bases de datos relacionales . 88

Swift: Servicio de almacenamiento de objetos de OpenStack que permite gestionar grandes cantidades de datos no estructurados de manera distribuida . 88

Tgt: Software de código abierto utilizado para implementar y gestionar objetivos iSCSI en sistemas Linux . 88

Thin-provisioning: Técnica de almacenamiento que asigna espacio de manera dinámica según las necesidades reales, optimizando el uso del almacenamiento físico . 88

Token: Credencial temporal que permite autenticar y autorizar accesos en un sistema . 88

Ubuntu: Distribución de Linux basada en Debian, ampliamente utilizada por su facilidad de uso . 88

Variable de entorno: Parámetro del sistema que almacena información accesible por aplicaciones y *scripts* . 88

VLAN: Virtual Local Area Network; protocolo de red que permite la segmentación de una red física en múltiples redes virtuales . 88

VMware: Plataforma de virtualización propietaria que ofrece soluciones para gestionar y operar máquinas virtuales y entornos de nube . 88

VNC: Computación de red virtual (*Virtual Network Computing*) que permite el acceso remoto a interfaces gráficas . 88

Volumen: Unidad lógica de almacenamiento que puede asignarse a un sistema o máquina virtual para guardar datos . 88

VPN: Red privada virtual (*Virtual Private Network*) que permite crear conexiones seguras y encriptadas entre redes públicas y privadas . 88

VXLAN: Virtual Extensible Local Area Network; protocolo de red basado en VLAN que permite una segmentación de redes adaptada a las necesidades de data centers . 88

WSGI: Interfaz estándar para la comunicación entre servidores web y aplicaciones Python, utilizada ampliamente en aplicaciones web . 88

Yoga: Versión de OpenStack, lanzada en 2022 . 88