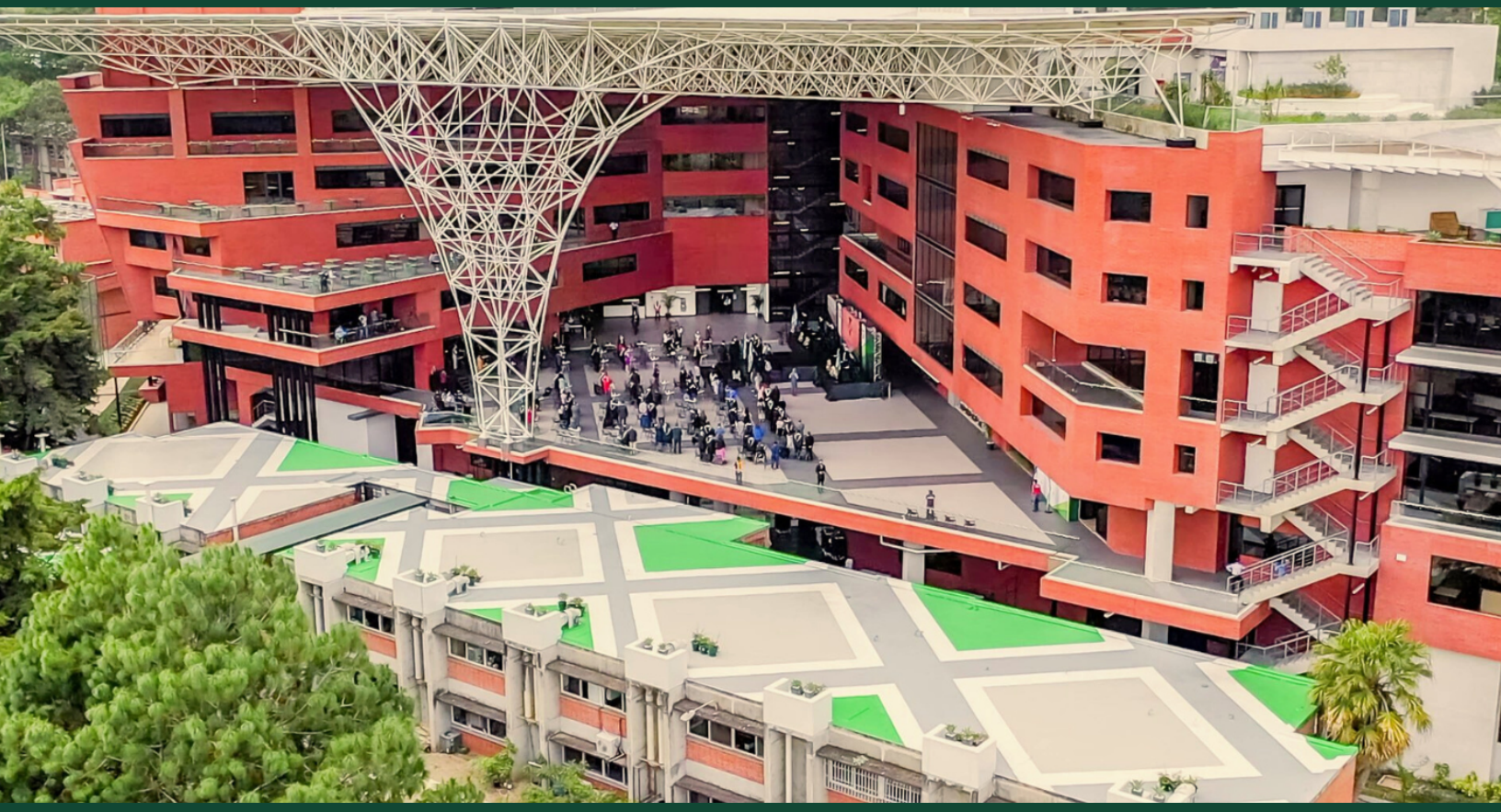
Mejora en la implementación de la virtualización multisistema operativo, gestión de imágenes e implementación de características de escalado automático de recursos en OpenStack

Gabriel Andrade Castejón



UVG | UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Mejora en la implementación de la virtualización multisistema operativo, gestión de imágenes e implementación de características de escalado automático de recursos en OpenStack

Trabajo de graduación presentado por Gabriel Andrade Castejón para optar al grado académico de Licenciado en Ingeniería Electrónica

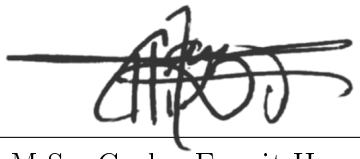
Guatemala,

2025

Vo.Bo.:

(f) 

M.Sc. Jonathan de los Santos

(f) 

M.Sc. Carlos Esquit Hernández

Fecha de aprobación: Guatemala, 20 de noviembre de 2025.

Prefacio

El presente trabajo de graduación es el resultado de un proceso de aprendizaje, perseverancia y crecimiento tanto personal como profesional, sin embargo, no es un logro individual, muchas personas han contribuido a que este proyecto se haga realidad. Deseo expresar, en primer lugar, mi más profundo agradecimiento a mi madre, cuyo apoyo incondicional, ejemplo y amor constante han sido un pilar fundamental en cada etapa de este camino. Su confianza en mí ha sido una fuente permanente de motivación.

Agradezco a mi asesor, Jonathan de los Santos, cuya guía rigurosa, criterio académico y disposición para orientar cada detalle enriquecieron la calidad de este trabajo. Su acompañamiento fue decisivo para llevar esta investigación a buen término. Extiendo también mi gratitud al Departamento de Ingeniería Electrónica de la Universidad del Valle, cada profesor fue determinante en la formación de mis conocimientos y habilidades; su dedicación a la enseñanza ha dejado una huella imborrable en mi desarrollo profesional.

Finalmente, agradezco a todas las personas que, de manera directa o indirecta, aportaron a la realización de este trabajo.

Índice general

Índice de figuras

Índice de cuadros

Resumen

El presente trabajo fue realizado con el objetivo de ampliar y mejorar las capacidades de la nube privada implementada en el Departamento de Electrónica de la Universidad del Valle de Guatemala, con el fin de superar las limitaciones encontradas en la virtualización y la falta de escalabilidad dinámica de recursos. Este problema fue abordado mediante una metodología de despliegue modular y flexible.

Mediante Glance y el uso de la herramienta QEMU-img, se estandarizó la conversión de imágenes a formato QCOW2, lo que optimizó el almacenamiento y la gestión de imágenes para diversos sistemas. Se implementó un catálogo de imágenes y *flavors* predefinidos, lo que facilitó la creación de instancias adaptadas a diferentes necesidades y aseguró compatibilidad con KVM mediante propiedades específicas del *firmware* y bus del disco.

La innovación principal del proyecto radica en la incorporación de capacidades de escalado automático de recursos mediante la orquestación de Heat, telemetría de Ceilometer y gestión de alarmas Aodh para configurar un circuito de retroalimentación autónomo (*closed-loop elasticity*). Se diseñó una política de escalado horizontal basada en el uso de CPU, validada mediante pruebas de carga artificial que demostraron la capacidad del sistema para adaptarse dinámicamente a variaciones en la demanda.

Como resultado, se estableció una plataforma de nube privada robusta y escalable que refleja las tendencias actuales en computación en la nube y proporciona una base sólida para futuras mejoras e innovaciones en el ámbito de la virtualización y gestión de recursos. Se concluye que los objetivos del proyecto fueron alcanzados con recomendaciones como un dashboard de monitoreo, reportes automatizados, escalabilidad vertical, integración de Ceph, balanceo de carga avanzado y pruebas automatizadas para validar las políticas de escalado.

Palabras clave: OpenStack, elasticidad horizontal, virtualización multisistema, *autoscaling group*, Heat.

Abstract

The present graduation work was carried out with the objective of expanding the capabilities of the OpenStack cloud computing platform at the Electronic Engineering Department of the Universidad del Valle de Guatemala, overcoming the initial limitations of virtualization and the lack of dynamic scalability. The problem was addressed using a modular deployment methodology.

Multi-operating system virtualization was established through the advanced configuration of the Glance service and the use of the QEMU-img tool for standardizing images to the QCOW2 format. A catalog of pre-defined images and flavors was implemented to facilitate the rapid creation of instances tailored to different needs, ensuring compatibility with KVM through specific properties of firmware and disk bus.

The main innovation of the project lies in the incorporation of automatic resource scaling capabilities using the Heat orchestration service, Ceilometer telemetry, and Aodh alarm management, configuring an autonomous feedback loop (closed-loop elasticity). A horizontal scaling policy based on CPU usage was designed and validated through artificial load testing, demonstrating the system's ability to dynamically adapt to demand variations.

As a result, a robust and scalable private cloud platform was established, reflecting current trends in cloud computing and providing a solid foundation for future improvements and innovations in virtualization and resource management. It is concluded that the project objectives were fully achieved, with future recommendations including a centralized monitoring dashboard, automated reports, vertical scalability, Ceph integration, advanced load balancing, and automated tests to validate scaling policies.

Keywords: OpenStack, horizontal elasticity, multi-operating system virtualization, autoscaling group, Heat.

CAPÍTULO 1

Introducción

La evolución constante en tecnologías informáticas exige que las plataformas académicas dispongan de infraestructuras flexibles y escalables. En este contexto, este trabajo se inscribe dentro del campo de la computación en la nube y orquestación de infraestructuras, centrándose en el uso de OpenStack, una plataforma de nube *open source* que opera bajo un modelo de infraestructura como servicio, consolidada como un pilar fundamental para el despliegue de nubes privadas y escalables.

La finalidad de este proyecto fue expandir y optimizar las capacidades de la nube privada OpenStack del Departamento de Electrónica de la Universidad del Valle de Guatemala, con el fin de superar las limitaciones existentes en términos de virtualización y escalabilidad dinámica de recursos. Para lograrlo, se implementaron diversas estrategias y herramientas que permitieron mejorar la gestión y el despliegue de recursos computacionales, adaptándose a las demandas de los usuarios.

Este proyecto se delimitó a dos ejes principales. El primero consistió en la estandarización y optimización del manejo de imágenes de máquinas virtuales mediante el servicio Glance de OpenStack, utilizando el formato QCOW2 para mejorar la eficiencia en el almacenamiento y la gestión de imágenes. Se desarrolló un catálogo de imágenes y *flavors* predefinidos, lo que facilitó la creación rápida de instancias adaptadas a diversas necesidades. El segundo eje se centró en la incorporación de capacidades de escalado automático de recursos mediante los servicios de orquestación Heat, telemetría Ceilometer y gestión de alarmas Aodh. Se diseñó una política de escalado horizontal basada en el uso de CPU, validada mediante pruebas de carga artificial que demostraron la capacidad del sistema para adaptarse de forma dinámica a variaciones en la demanda. Este enfoque permitió establecer un circuito de retroalimentación autónomo (*closed-loop elasticity*), que optimizó el uso de recursos y mejoró la eficiencia operativa.

Así se concluye con una virtualización multisistema operativo completa y eficiente, que habilita sistemas cruciales como Windows en su versión de servidor y Professional, así como diversas distribuciones de Linux tanto en sus versiones de servidor como de escritorio, un sistema de escalado automático robusto, personalizable y flexible validada mediante pruebas de carga. La plataforma resultante no solo refleja las tendencias actuales en computación en la nube, sino que también proporciona una base sólida para futuras mejoras e innovaciones en el ámbito de la virtualización y gestión de recursos para el Departamento de Electrónica, contribuyendo significativamente a la formación académica y el desarrollo tecnológico en la institución.

CAPÍTULO 2

Antecedentes

El desarrollo de soluciones basadas en tecnologías de virtualización y computación en la nube ha permitido una transformación sustancial en la forma en que las instituciones académicas, organizaciones gubernamentales y empresas privadas gestionan sus recursos informáticos. En este contexto, OpenStack ha emergido como una plataforma robusta, escalable y de código abierto que permite el despliegue y la gestión de infraestructuras de nube privadas y públicas.

El trabajo [1] representa un esfuerzo significativo por introducir OpenStack en el entorno académico del Departamento de Electrónica de la Universidad del Valle de Guatemala, logrando el despliegue modular de sus principales servicios sobre una infraestructura de cómputo de alto rendimiento (HPC). No obstante, su implementación se limitó a sistemas Linux, dejando fuera de la primera iteración la virtualización de sistemas operativos como Windows, y sin incluir estrategias de elasticidad computacional en la nube.

Casos internacionales refuerzan el valor de OpenStack en contextos académicos. Un ejemplo prominente es el CERN, la Organización Europea para la Investigación Nuclear, que ha utilizado OpenStack desde 2013 para gestionar miles de nodos de cómputo. Su implementación ha evolucionado desde la provisión de máquinas virtuales simples hasta soluciones complejas que incluyen contenedores, redes definidas por *software* (SDN) y automatización de flujos de trabajo [2].

Complementando lo anterior, la tesis desarrollada en la Universidad de las Ciencias Informáticas de Cuba [3] sobre la gestión de imágenes de sistema operativo en Nova-LTSP destaca la necesidad de interfaces amigables y módulos funcionales para facilitar la virtualización multiplataforma. El trabajo plantea una solución para reducir la complejidad del manejo de imágenes en entornos educativos, aportando con ello una perspectiva concreta sobre los retos técnicos y operativos que también son relevantes para OpenStack.

Asimismo, el estudio presentado por [4] demuestra el uso efectivo de OpenStack en una arquitectura basada en contenedores y servicios distribuidos para instituciones académicas. El enfoque permite gestionar recursos de cómputo de manera eficiente, haciendo uso de Ceilometer y Heat para escalar servicios bajo demanda, lo cual se alinea con el objetivo de implementar *elastic computing* en la propuesta actual.

Finalmente, el trabajo de tesis presentado en [5] aborda una problemática similar al presente proyecto. Su propuesta incluye la implementación de herramientas de monitoreo, autenticación segura, y la integración de Ceph para almacenamiento, lo que aporta una visión holística de una nube académica avanzada.

Estos antecedentes demuestran que el proyecto en desarrollo no solo es pertinente, sino también alineado con tendencias internacionales de innovación tecnológica en educación. Además, ofrecen una base sólida para mejorar y añadir características nuevas respecto el trabajo previo, especialmente en cuanto a la virtualización de sistemas operativos diversos y la incorporación de mecanismos de escalabilidad dinámica en la infraestructura de nube.

CAPÍTULO 3

Justificación

La plataforma OpenStack desplegada en el Departamento de Electrónica de la Universidad del Valle de Guatemala representa un recurso estratégico para la enseñanza, investigación y optimización del uso de la infraestructura de cómputo de alto rendimiento (HPC) disponible en el campus. No obstante, el entorno actual permite la virtualización de sistemas basados en Linux y no toma en consideración capacidades dinámicas de escalamiento automático de recursos.

Esta restricción reduce el alcance de uso de la plataforma, ya que excluye sistemas operativos utilizados en entornos académicos e industriales, como Windows y anula la posibilidad de virtualización de dispositivos independientes. Además, la falta de capacidades de asignación de recursos de forma dinámica impide que el sistema ajuste su capacidad de procesamiento de manera automática en función de la demanda, generando riesgos de sobrecarga o subutilización de los recursos.

El presente proyecto busca resolver estos problemas mediante la habilitación de la virtualización de sistemas operativos fuera del ambiente Linux y la implementación de mecanismos de escalamiento automático utilizando Heat y Ceilometer. De esta manera, se ampliará la versatilidad de la plataforma, se optimizará el uso de la infraestructura existente y se ofrecerá a los estudiantes y docentes un entorno de nube más realista, completo y alineado con las necesidades actuales de la industria.

La ejecución de este proyecto es viable, ya que se aprovechan las bases sólidas del despliegue modular previo, utilizando herramientas y servicios nativos de OpenStack, de código abierto y ampliamente documentados. Asimismo, el impacto esperado es significativo, dado que permitirá extender el uso académico y de investigación de la nube, incrementar el rendimiento del *hardware* disponible, y fortalecer la formación práctica de los futuros ingenieros en tecnologías de virtualización y *cloud computing*.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Ampliar las capacidades de la plataforma de OpenStack desplegada en el Departamento de Electrónica de la UVG mediante la mejora de la virtualización de todo tipo de sistemas operativos y la implementación de *elastic computing*.

4.2. Objetivos específicos

- Configurar y adaptar el servicio de imágenes Glance para soportar y gestionar adecuadamente nuevos formatos compatibles con KVM.
- Implementar el servicio Heat de OpenStack para gestionar la creación automática de recursos en función de plantillas *(autoscaling)*.
- Instalar y configurar el servicio Ceilometer para la recopilación de métricas y el monitoreo de instancias.
- Diseñar y validar una política de escalamiento automático basada en el consumo de recursos como CPU o RAM.
- Documentar de manera detallada el proceso de instalación, configuración y pruebas realizadas, proponiendo lineamientos para futuras expansiones del sistema.

CAPÍTULO 5

Alcance

El proyecto está previsto dentro de la infraestructura existente del Departamento de Electrónica de la Universidad del Valle de Guatemala, una plataforma OpenStack capaz de ofrecer virtualización multisistema y computación elástica plenamente operativa. Para lograrlo, se considera únicamente el siguiente hardware: un Dell Precision 7920 (88 vCPU, 376 GB RAM, 12 TB de bloques) y un Dell PowerEdge T560 (112 vCPU, 256 GB RAM, 2 TB RAID-1 para SO), ambos integrados como nodos de cómputo gobernados por el nodo controlador actual.

A nivel funcional, el proyecto avanza en cuatro fases concatenadas. Primero se ejecuta un diagnóstico detallado de servicios (Keystone, Nova, Neutron, Glance y Horizon) para identificar limitaciones y puntos de mejora. Luego se aborda la preparación de imágenes multiplataforma: conversión a QCOW2, optimización con VirtIO, incorporación de Cloud-Init/Cloudbase-Init y carga en Glance mediante CLI y Horizon, definiendo metadatos y flavors específicos en Nova que garanticen rendimiento consistente.

La tercera fase introduce elastic computing, instalación y configuración de Heat como motor de orquestación, Ceilometer como colector de métricas y Aodh para alarmas, de modo que las plantillas HOT gobiernen grupos de autoescalado basados en consumo de CPU o RAM. Finalmente se efectúan pruebas de validación y ajuste, sometiendo la nube a cargas artificiales con herramientas de comandos mediante CLI para afinar umbrales y tiempos de enfriamiento.

El cronograma previsto cubre desde enero hasta noviembre del 2025, con hitos en cada trimestre que parten de análisis inicial y concluyen con monitoreo continuo. Como entregable se generará, repositorio de imágenes QCOW2 optimizadas y documentadas, catálogo de flavors personalizados, conjunto de plantillas HOT y scripts de configuración para Heat, Ceilometer y Aodh informe técnico con resultados de pruebas de carga, métricas de escalado y recomendaciones operativas y guía de buenas prácticas para futuras expansiones.

Queda expresamente fuera del alcance la compra de hardware adicional, la integración de contenedores (Kubernetes) o almacenamiento distribuido Ceph, la facturación multitenant y la migración masiva de todas las cargas existentes, el proyecto demostrará únicamente instancias de referencia necesarias para validar la solución. Así, se focaliza en dotar a la comunidad académica de un entorno de nube más versátil y resiliente sin desbordar los recursos disponibles.

CAPÍTULO 6

Marco teórico

6.1. Computación en la nube

La computación en la nube, también conocida como *cloud computing*, se refiere a un modelo de prestación de servicios computacionales bajo demanda a través de redes tanto públicas como privadas, donde permiten acceder a distintos tipos de recursos tales como almacenamiento, procesamiento, aplicaciones y redes de una manera altamente flexible y escalable. Los recursos mencionados pueden ser provisionados con un esfuerzo mínimo de gestión, creando una infraestructura donde los usuarios pueden consumir servicios informáticos sin necesidad de poseer ni gestionar la infraestructura física. Este tipo de modelo de computación provee una gran eficiencia en costos, disponibilidad cercana al 100 % y una optimización y flexibilidad en el uso de recursos, permitiendo a las organizaciones adaptarse rápidamente a las demandas cambiantes del mercado y a las necesidades tecnológicas

Este modelo de computación se ha convertido en una herramienta y pilar fundamental para la modernización de tecnologías informáticas, gracias a sus capacidades de:

- **Escalabilidad**: permite ajustar la capacidad de recursos a proveer de una forma dinámica, adaptándose a las necesidades cambiantes de los usuarios y cargas de trabajo.
- **Flexibilidad**: facilita la integración de tecnologías y servicios diversos en un entorno unificado gracias a las máquinas virtuales y contenedores.
- **Eficiencia en costos**: los usuarios pagan únicamente por los recursos consumidos, eliminando la necesidad de inversiones significativas en infraestructura física ni mantenimiento de la misma.

- **Accesibilidad**: los recursos y servicios tienen disponibilidad desde cualquier ubicación con una conexión a la red donde se haya establecido el servicio.
- **Mantenimiento y actualización simplificados**: el proveedor de servicios en la nube es el encargado de proveer el mantenimiento y actualizaciones de toda la infraestructura, dejando a los usuarios enfocados en el apartado funcional y aplicativo de sus requerimientos [6].

6.1.1. Modelos de servicio en la nube

La computación en la nube contiene tres modelos de servicio principales, estos definen el nivel de control y responsabilidad que tienen los usuarios sobre los servicios e infraestructura que se está proveyendo. Los modelos principales son los siguientes:

- **Infraestructura como servicio (IaaS)**: este servicio provee recursos de la infraestructura computacional, como lo son servidores, almacenamiento y redes, dando permisos a los usuarios para gestionar y controlar el sistema operativo, aplicaciones y configuraciones de red. Los usuarios tienen control total sobre la infraestructura virtualizada, permitiendo una gran flexibilidad y personalización. Ejemplos de IaaS incluyen Amazon Web Services (AWS) EC2, Microsoft Azure Virtual Machines y Google Compute Engine.
- **Plataforma como servicio (PaaS)**: este servicio proporciona plataformas o entornos de desarrollo completos, permitiendo que los usuarios desarrollen, implementen y gestionen las aplicaciones sin tener que ocuparse de la infraestructura, esto logra que los desarrolladores logren enfocarse en el desarrollo de software. Ejemplos de PaaS incluyen Google App Engine, Microsoft Azure App Service y Heroku.
- ***Software* como servicio (SaaS)**: este servicio entrega aplicaciones y *software* completos a través de la nube, permitiendo a los usuarios acceder y utilizar aplicaciones sin necesidad de instalarlas o gestionarlas localmente. Los usuarios solo interactúan con la aplicación a través de una interfaz web o API, mientras que el proveedor se encarga de la infraestructura subyacente. Ejemplos de SaaS incluyen Google Workspace, Microsoft 365 y Salesforce [7].

6.1.2. Modelos de implementación de la nube

Los modelos de implementación indican la forma y dónde se gestiona la infraestructura de nube, cada uno adaptándose a distintos contextos organizacionales para suplir necesidades específicas.

- **Nube pública**: en este modelo, los recursos son totalmente propiedad por un proveedor externo, por lo mismo, gestionado totalmente por dicho proveedor. Este proveedor los ofrece al público general. Es el modelo más rentable y fácil de escalar e implementar, sin embargo, al tener menos control sobre la infraestructura y la seguridad, puede no ser adecuado para todas las organizaciones.
- **Nube privada**: en este modelo, la infraestructura es exclusiva para la misma organización que está utilizado los servicios. Puede ser ubicada y gestionada en instalaciones propias o

con proveedores externos en centros de datos. Este modelo ofrece más control, acceso a la seguridad y personalización, sin embargo, es más costoso y complejo de gestionar.

- **Nube híbrida**: este modelo combina características de nubes públicas y privadas, permitiendo que las organizaciones utilicen ambos tipos de nubes para diferentes cargas de trabajo. Esto permite a las organizaciones aprovechar la escalabilidad y eficiencia de costos de la nube pública, mientras mantienen el control y seguridad de la nube privada para datos sensibles o aplicaciones críticas.
- **Nube comunitaria**: este modelo es compartido por organizaciones con intereses o requisitos alineados, como regulaciones en la seguridad o cumplimiento de servicio. La infraestructura puede ser gestionada por estas mismas organizaciones o un proveedor externo. Este modelo permite compartir costos y recursos, su desafío principal radica en la gestión y coordinación de las políticas y necesidades de cada entidad participante [7].

6.2. Virtualización

La virtualización se refiere a una tecnología que permite ejecutar múltiples sistemas operativos o aplicaciones simultáneamente en un mismo *hardware* físico, mediante uso de capas de abstracción denominadas hipervisores. Esta tecnología permite la creación de entornos virtuales independientes, denominados como máquinas virtuales (VM), que optimizan el uso de los recursos computacionales, separan los entornos del *hardware* subyacente y facilitan la gestión y despliegue de aplicaciones y servicios.

Dentro del contexto de la computación en la nube, la virtualización es una pieza clave para la creación de instancias de servidores, donde cada instancia puede ser configurada y personalizada para cada usuario o aplicación, de forma independiente [8].

6.2.1. Hipervisores y formatos de imágenes

Un componente clave para la virtualización es el hipervisor, responsable de asignar los recursos computacionales hacia las máquinas virtuales. En OpenStack, el hipervisor utilizado ampliamente es KVM (Kernel-based Virtual Machine), siendo una solución de código abierto también, que convierte el núcleo de Linux en un hipervisor de tipo 1, permitiendo la creación y gestión de máquinas virtuales con un alto rendimiento y eficiencia. KVM es compatible con una amplia variedad de sistemas operativos invitados, incluyendo Linux, Windows y otros sistemas basados en Unix.

Con formato de imagen nos referimos a los archivos que contienen el sistema operativo y aplicaciones necesarias para ejecutar la máquina virtual. Los hipervisores utilizan estos archivos para crear instancias de máquinas virtuales. En OpenStack y KVM, los formatos de imagen más comunes son los siguientes:

- **QCOW2 (QEMU Copy-On-Write)**: es el formato de disco más avanzado, soportado por QEMU y KVM, se diseñó con el fin de funcionalidades avanzadas para entornos virtualizados complejos. Este formato permite crear *snapshots* o clones diferenciados que solo almacenan

cambios respecto a la imagen original, ahorrando espacio en disco y facilitando administración. Integra compresión que permite reducir espacio físico ocupado por las imágenes. Ofrece protección del contenido del disco mediante encriptación a nivel del bloque, mejorando la seguridad de los datos almacenados. Además, permite la creación de imágenes dinámicas que crecen según se necesite, optimizando el uso del espacio en disco.

- **RAW**: es el formato más básico y simple, es una representación bit por bit del disco duro de la máquina virtual. No contiene capas adicionales de metadatos o funcionalidades extras. Dentro de sus ventajas, tiene acceso directo a los bloques de almacenamiento, haciendo que las operaciones de lectura y escritura sean más rápidas y eficientes Aunque tiene un mejor rendimiento, se considera un formato menos funcional por su carencia de mecanismos adicionales de gestión en entornos de desarrollo, el formato RAW se reserva para entornos de nube donde se prioriza absolutamente el rendimiento y la simplicidad.
- **VMDK (Virtual Machine Disk)**: es un formato de disco virtual utilizado por VMware, es compatible con KVM y OpenStack. Permite la creación de discos virtuales que pueden ser utilizados por máquinas virtuales. Este formato es ampliamente utilizado en entornos de virtualización y es compatible con una variedad de herramientas y plataformas. Aunque no ofrece las mismas funcionalidades avanzadas que QCOW2, es una opción popular debido a su compatibilidad y facilidad de uso.
- **VDI (Virtual Disk Image)**: es un formato de disco virtual utilizado por VirtualBox, es compatible con KVM y OpenStack. Permite la creación de discos virtuales que pueden ser utilizados por máquinas virtuales. Aunque no ofrece las mismas funcionalidades avanzadas que QCOW2, es una opción popular debido a su compatibilidad y facilidad de uso [9].

Los formatos VMDK y VDI son útiles para una creación rápida de máquinas virtuales, pruebas de software y desarrollo de aplicaciones en entornos virtualizados, sin embargo, no son compatibles de forma directa con KVM, pero pueden ser convertidos a formatos compatibles como QCOW2 o RAW utilizando herramientas como QEMU-img, que es una utilidad de línea de comandos que permite convertir entre diferentes formatos de imagen de disco.

6.2.2. Conversión de formatos de imagen

El proceso de conversión de imágenes es esencial para la virtualización. Cada formato cuenta con sus propias características y ventajas; en este caso, el formato es QCOW2 por su gran compatibilidad con KVM y OpenStack, además de sus funcionalidades avanzadas como compresión, encriptación y creación de *snapshots*. Este formato de imagen no es tan común en las distribuciones verificadas de los sistemas operativos, por lo que es necesario convertir las imágenes de otros formatos a QCOW2 para su uso en OpenStack.

QEMU-img

QEMU-img es una herramienta de línea de comandos donde es posible convertir imágenes de disco entre diferentes formatos, tales como RAW, QCOW2, VMDK y VDI. Esta herramienta es parte del paquete QEMU y es ampliamente utilizada en entornos de virtualización para gestionar imágenes de disco virtuales.

El comando principal para convertir imágenes de disco es el siguiente:

Cuadro 1. Conversión de formatos de imagen con QEMU-img

```
qemu-img convert -c -p -f <formato_origen> -O <formato_destino> <archivo_origen>
    ↪ <archivo_destino>
```

Nota. Elaboración propia.

- `-c`: activa la compresión de la imagen de destino.
- `-p`: muestra el progreso de la conversión.
- `-f <formato_origen>`: especifica el formato de la imagen de origen.
- `-O <formato_destino>`: especifica el formato de la imagen de destino.
- `<archivo_origen>`: especifica la ruta del archivo de imagen de origen.
- `<archivo_destino>`: especifica la ruta del archivo de imagen de destino.

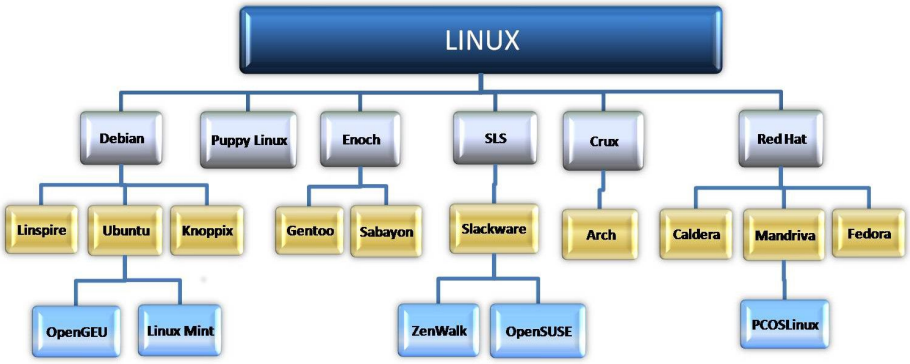
6.2.3. Distribuciones Linux

Linux es un sistema operativo de código abierto basado en el núcleo Linux. Esta característica ha hecho que sea un sistema operativo ampliamente personalizado en distintas distribuciones, donde cada distribución cuenta con enfoques y características específicas, con el fin de adaptarse a diversas necesidades. Estas distribuciones se pueden desglosar en familias principales donde existen subfamilias que comparten características comunes. Algunas de las más destacadas son las siguientes:

- **Debian**: es una de las distribuciones más antiguas y estables; es conocida por su enfoque en la estabilidad y seguridad. Utiliza el sistema de gestión de paquetes APT (Advanced Package Tool) para la instalación y actualización de software. Debian es la base de muchas otras distribuciones populares, como Ubuntu y Linux Mint.
- **Red Hat**: es una distribución comercial que se centra en la estabilidad y el soporte empresarial. Utiliza el sistema de gestión de paquetes RPM (Red Hat Package Manager) y es conocida por su enfoque en la seguridad y la escalabilidad. Red Hat es la base de otras distribuciones populares como CentOS, Fedora y Rocky Linux.
- **Arch Linux**: es una distribución *rolling release* que se centra en la simplicidad y la personalización. Arch Linux utiliza un enfoque autodidacta, lo que permite a los usuarios construir su sistema desde cero y personalizarlo según sus necesidades. Utiliza el sistema de gestión de paquetes Pacman y tiene una amplia documentación disponible. Arch Linux es la base de otras distribuciones populares como Manjaro y EndeavourOS.

- **SUSE**: es una distribución comercial que se centra en la estabilidad y el soporte empresarial. Utiliza el sistema de gestión de paquetes Zypper y es conocida por su enfoque en la seguridad y la escalabilidad. SUSE es la base de otras distribuciones populares como openSUSE y SLE (SUSE Linux Enterprise).
- **Gentoo**: es una distribución que se centra en la personalización y la optimización del rendimiento. Utiliza un enfoque basado en la compilación desde el código fuente, lo que permite a los usuarios personalizar su sistema según sus necesidades específicas. Gentoo utiliza el sistema de gestión de paquetes Portage y es conocida por su enfoque en la flexibilidad y el control total sobre el sistema.
- **Slackware**: es una de las distribuciones más antiguas y estables, es conocida por su enfoque en la simplicidad y la estabilidad. Utiliza un sistema de gestión de paquetes propio y es conocida por su enfoque en la seguridad y la escalabilidad. Slackware es una distribución que se centra en la experiencia del usuario y la personalización del sistema [10].

Figura 1. Familias de distribuciones Linux.



Nota. Adaptado de [10].

6.2.4. Sistemas Windows

Desarrollado por Microsoft y uno de los sistemas operativos más utilizados globalmente, lo caracteriza su interfaz gráfica intuitiva y gran compatibilidad con aplicaciones y *hardware*. Windows es utilizado tanto en entornos domésticos, como en entornos empresariales. Windows cuenta con una gran gama de versiones, cada una diseñada para diferentes propósitos y necesidades. Algunas de las versiones más destacadas son las siguientes:

- **Windows 10**: es una de las versiones más populares y ampliamente utilizadas de Windows. Ofrece una interfaz moderna y una amplia gama de características, incluyendo Cortana, el asistente virtual de Microsoft, y Microsoft Edge, el navegador web. Windows 10 es compatible con una amplia gama de aplicaciones y hardware, lo que lo convierte en una opción popular para usuarios domésticos y empresariales.

- **Windows Server**: es una versión de Windows diseñada específicamente para entornos empresariales y servidores. Ofrece características avanzadas de seguridad, escalabilidad y gestión de redes. Windows Server es utilizado para ejecutar aplicaciones empresariales, gestionar redes y proporcionar servicios de infraestructura. Algunas de las versiones más destacadas de Windows Server son Windows Server 2016, Windows Server 2019 y Windows Server 2022.
- **Windows 11**: es la última versión de Windows, lanzada en octubre de 2021. Ofrece una interfaz moderna y una amplia gama de características, incluyendo mejoras en la productividad, seguridad y rendimiento. Windows 11 es compatible con una amplia gama de aplicaciones y hardware, lo que lo convierte en una opción popular para usuarios domésticos y empresariales.

6.2.5. Ventajas académicas de la virtualización multiplataforma

Contar con la capacidad de virtualizar sistemas operativos y aplicaciones en un entorno de nube, permite crear un ambiente de aprendizaje flexible y accesible tanto para estudiantes como docentes. Tener una gama amplia de sistemas operativos a virtualizar, hace posible experimentar con diferentes tecnologías para adaptarse a las necesidades de cada proyecto o aplicación. Las principales ventajas académicas de la virtualización multiplataforma son las siguientes:

- **Versatilidad de sistema operativo**: esta tecnología permite el uso de *software* propietario común en entornos industriales y académicos como MATLAB, AutoCAD, LabView, entre otros, sin necesidad de adquirir licencias costosas para cada estudiante. Esto permite a los estudiantes aprender y practicar con herramientas que encontrarán en el mundo laboral.
- **Experimentación segura**: la virtualización permite a los estudiantes experimentar con diferentes sistemas operativos y configuraciones sin riesgo de dañar el sistema principal. Esto es especialmente útil para aprender sobre seguridad informática, redes y administración de sistemas. Especialmente útil en una infraestructura donde se cuenta con una alta gama de sistemas operativos y aplicaciones.
- **Facilidad de acceso**: los estudiantes pueden acceder a sus entornos virtuales desde cualquier dispositivo con conexión a Internet, lo que facilita el aprendizaje y la colaboración. Esto es especialmente útil en entornos educativos donde los estudiantes pueden no tener acceso a *hardware* específico o *software* instalado localmente para proyectos académicos que requieran una carga computacional alta.
- **Preparación para el futuro**: la virtualización es una tecnología clave en la industria de la computación en la nube y la infraestructura de TI. Al aprender sobre virtualización y su aplicación en entornos académicos, los estudiantes adquieren habilidades valiosas que son altamente demandadas en el mercado laboral. Esto les prepara para carreras en administración de sistemas, desarrollo de sistemas operativos y tecnologías de la información [11] [12].

6.3. OpenStack como infraestructura modular de nube

OpenStack es una plataforma de servicios de nube de código abierto, orientada principalmente a implementación de nubes privadas, una solución ideal para el despliegue nube en la red de laboratorio de la Universidad del Valle de Guatemala, ofreciendo un despliegue totalmente privado y controlado por la institución, así como grandes posibilidades de expansiones y personalizaciones en el despliegue para un espectro amplio de aplicaciones y servicios. Funciona bajo el modelo de infraestructura como servicio (IaaS), su desarrollo fue iniciado por Rackspace y la NASA en el 2010, desde entonces ha evolucionado como un ecosistema robusto y modular con un respaldo amplio por la comunidad de desarrolladores y empresas tecnológicas.

Su arquitectura modular y distribuida permite la administración de recursos de cómputo, almacenamiento y distribución de redes por medio de servicios independientes y a la vez interconectados entre sí por medio de interfaces API REST y unificado mediante su interfaz gráfica Horizon [13].

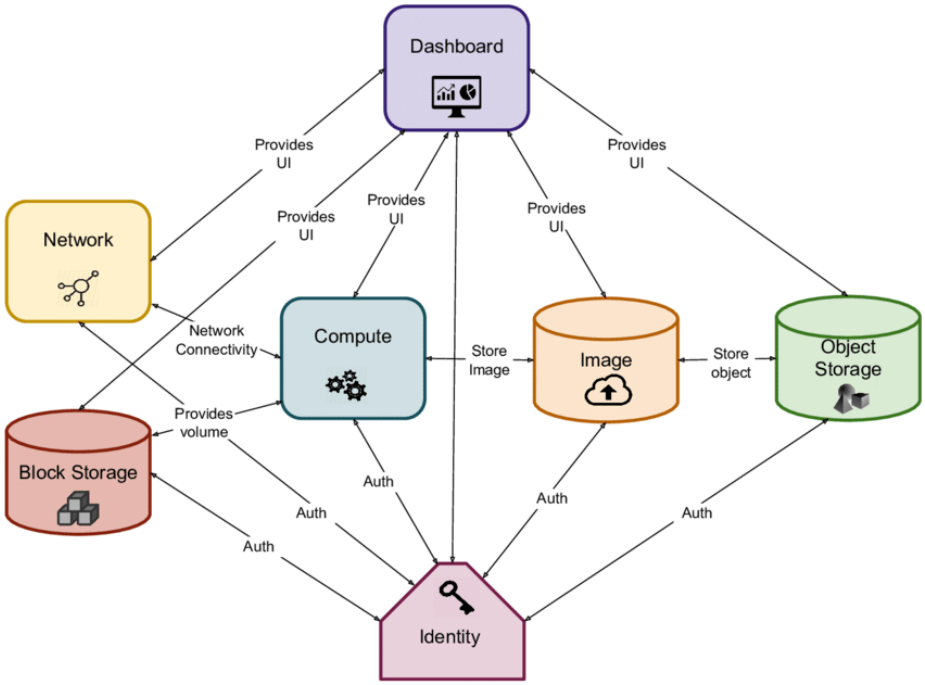
6.3.1. Componentes principales de OpenStack

La fortaleza principal de OpenStack radica en su diseño modular, donde cada función del sistema es gestionada por un componente específico que se comunica entre sí como un modelo de entradas y salidas conectadas entre servicios, lo que permite escalabilidad, mantenimiento y flexibilidad en la implementación de una forma sencilla. Entre los módulos esenciales se encuentran:

- **Nova (*Compute*):** Es el componente encargado de la provisión, gestión y programación de todo el ciclo de vida de las instancias de máquinas virtuales, usando hipervisores como KVM, QEMU o Xen. Nova permite la creación, eliminación y escalado de instancias, así como la asignación de recursos de cómputo a estas instancias.
- **Neutron (*Networking*):** Es el componente qué proporciona conectividad entre las instancias de máquinas virtuales con redes externas u otros recursos de redes externas. Permite creación de redes virtuales privadas, conjunto a subredes con sus correspondientes componentes como *routers* virtuales, IP flotantes, y todos sus conjuntos de reglas y seguridad.
- **Glance (*Image*):** Es el administrador de todo el catálogo de imágenes de tanto sistemas operativos, aplicaciones y discos, con principal soporte en formatos QCOW2, RAW y VHD. Glance también es el encargado de los *snapshots* de las instancias, permitiendo copias de seguridad.
- **Keystone (*Identity*):** Es el componente de autenticación y autorización, dando un servicio centralizado para la gestión de usuarios, sus roles y permisos, así como la integración entre servicios de OpenStack y otros sistemas de autenticación externos. Keystone permite la creación de proyectos y asignación de usuarios a estos, facilitando la gestión de acceso a los recursos de OpenStack.
- **Cinder (*Block Storage*):** Proporciona la infraestructura de almacenamiento en bloque persistente para las instancias de máquinas virtuales conectándose como si se tratara de discos duros virtuales. Permitiendo la creación, eliminación y gestión de estos volúmenes de almacenamiento.

- **Placement (*Resource Management*):** Es el encargado de la gestión de recursos y su asignación a las instancias de máquinas virtuales. Se centra en la planificación y programación eficiente de todos los recursos disponibles, optimizando el uso de los mismos y mejorando el rendimiento del sistema. Placement permite a OpenStack tomar decisiones informadas sobre dónde ubicar las instancias de máquinas virtuales basándose en la disponibilidad y capacidad de los recursos como CPU, RAM y almacenamiento.
- **Horizon (*Dashboard*):** Es la interfaz gráfica basada en web para interacción de la infraestructura sin necesidad de acceder mediante CLI, conectándose desde la IP asignada a este servicio. Permite a los usuarios gestionar y visualizar los recursos de la nube privada, dando un panel de control unificado para acceder a todos los servicios de OpenStack. Horizon facilita la administración de instancias, redes, almacenamiento y otros recursos, proporcionando una experiencia de usuario intuitiva y accesible [13].

Figura 2. Componentes principales de OpenStack y su interacción.



Nota. Adaptado de [14].

6.3.2. Servicios avanzados de OpenStack

Para la implementación de una infraestructura de nube con autoescalado de recursos será necesario instalar y configurar una serie de servicios adicionales a los componentes principales de OpenStack, estos servicios permiten la creación de una infraestructura robusta y escalable, permitiendo la gestión eficiente de recursos y la automatización de tareas, cada uno de los servicios cumple un rol específico para esta infraestructura de autoescalado.

- **Heat (*Orchestration*):** Heat permite definir infraestructuras completas mediante código de plantillas HOT (*Heat Orchestration Templates*), facilitando la creación de *stacks* que agrupan recursos interconectados, facilitando la implementación de arquitecturas complejas y automatización de tareas. Heat permite a los usuarios definir y gestionar sus infraestructuras como código, lo que facilita la replicación y escalabilidad de entornos [15].
- **Ceilometer (*Telemetry*):** Ceilometer es un recopilador de información sobre el uso de recursos tales como CPU, RAM, disco y redes, permitiendo la monitorización y medición del rendimiento de los recursos en la nube. Este servicio es fundamental para la implementación de políticas de autoescalado, ya que proporciona métricas y estadísticas sobre el consumo de recursos [16].
- **Gnocchi (*Telemetry Storage*):** Gnocchi es un sistema de almacenamiento y recuperación de series temporales diseñado para manejar grandes volúmenes de datos de telemetría. Funciona en conjunto con Ceilometer para almacenar métricas de uso de recursos, permitiendo consultas eficientes y análisis histórico. Gnocchi está optimizado para el rendimiento y la escalabilidad, lo que lo hace ideal para entornos de nube donde se generan grandes cantidades de datos [17].
- **Aodh (*Alarming*):** Aodh genera alarmas en base a las métricas recolectadas, activando banderas para acciones automáticas en base a esta información, necesario para escalado de instancias, notificaciones de uso a los administradores o acciones correctivas. Aodh permite definir umbrales y condiciones para activar alarmas, facilitando la gestión proactiva de la infraestructura y la respuesta a eventos críticos [18].

6.4. Herramientas de diagnóstico y monitoreo

La calidad del servicio en una infraestructura de nube privada es fundamental y depende de la observabilidad continua y a mayor detalle posible, tanto en los componentes físicos como los lógicos. Para lograr esto, se pueden utilizar diversas herramientas de diagnóstico y monitoreo que permiten visualizar y analizar el rendimiento de la infraestructura en tiempo real.

El primer método para lograr un monitoreo de la infraestructura de nube es mediante herramientas de línea de comandos que permiten interrogar al plano de control tanto de OpenStack, como el de servicios adicionales e incluso el del sistema operativo. Estas herramientas proporcionan métricas en tiempo real sobre el estado de los recursos, permitiendo a los administradores identificar cuellos de botella y problemas de rendimiento de manera proactiva.

6.4.1. CLI de OpenStack

La CLI de OpenStack es una herramienta poderosa que permite a los administradores interactuar con los servicios de OpenStack a través de comandos en la terminal. Proporciona una forma rápida y eficiente de obtener información sobre el estado de la infraestructura, así como de realizar cambios en la configuración y gestión de recursos. Los comandos más relevantes para el monitoreo son los siguientes:

- **openstack server list**: muestra una lista de todas las instancias de servidor en ejecución.
- **openstack volume list**: muestra una lista de todos los volúmenes de almacenamiento disponibles.
- **openstack network list**: muestra una lista de todas las redes configuradas en la infraestructura.
- **openstack-status**: resume la salud global de los servicios (Nova, Neutron, Glance, etc.), validando procesos, *sockets* y dependencias.
- **openstack service list**: interroga el catálogo de servicios registrados en Keystone y verifica que sus *endpoints* estén accesibles.
- **openstack compute service list**: consulta la tabla de Nova Scheduler para corroborar que los *compute nodes* reportan correctamente su latencia y estado.
- **openstack image list**: revisa la base de datos de Glance, asegurando que los metadatos de las imágenes sean coherentes y que existan réplicas consistentes.
- **openstack network agent list**: muestra los agentes Neutron (L3, DHCP, OVS) y sus *heartbeats*, lo que permite detectar fallos de redificación antes de que impacten a las instancias virtuales.
- **openstack hypervisor list**: proporciona información sobre los hipervisores en uso, incluyendo su estado y estadísticas de rendimiento.
- **openstack baremetal node list**: muestra una lista de todos los nodos *baremetal* disponibles en la infraestructura.
- **openstack volume show**: muestra información detallada sobre un volumen de almacenamiento específico, incluyendo su estado y rendimiento [19].

Para el diagnóstico del plano de control de OpenStack los subcomandos de openstack actúan como clientes REST que consultan el catálogo de Keystone y se autentican mediante *tokens*, exponiendo el estado interno de cada servicio. Estos comandos materializan el principio de introspección de la infraestructura como servicio, donde cada módulo expone sus métricas a través de API estándar, lo que facilita el diagnóstico distribuido [18].

6.4.2. CLI de Linux

Mediante la interfaz de CLI de Linux se puede tener un control granular sobre el estado del sistema operativo, lo que permite monitorear procesos, recursos y configuraciones en tiempo real. Los comandos más relevantes son los siguientes:

- **top**: muestra una lista de los procesos en ejecución y su uso de recursos.
- **df -h**: muestra el uso del espacio en disco de todos los sistemas de archivos montados.
- **free -m**: muestra información sobre el uso de la memoria RAM y *swap*.

- **netstat -tuln**: muestra las conexiones de red activas y los puertos en escucha.
- **vmstat**: proporciona información sobre procesos, memoria, paginación, bloqueos y actividad de CPU.
- **iostat**: muestra estadísticas de entrada/salida para dispositivos de bloque.
- **mpstat**: muestra estadísticas de uso de CPU por procesador.
- **pidstat**: muestra estadísticas de uso de recursos por proceso.
- **journalctl -f**: muestra los registros del sistema en tiempo real.
- **ip addr**: consulta las tablas RTNL para exponer interfaces, direcciones y rutas, permitiendo correlacionar configuraciones de Neutron con el sistema base.
- **iptables**: opera como cortafuegos en espacio de usuario, implementando filtrado y NAT (*Network Address Translation*) mediante ganchos del *netfilter*.
- **tcpdump**: captura tráfico en crudo, indispensable para localizar pérdidas de paquetes o colisiones de VLAN.
- **stress-ng**: genera cargas sintéticas controladas, permitiendo validar umbrales de saturación y la efectividad de políticas de *autoscaling*.
- **nmap**: ejecuta escaneos de puertos y detección de servicios, lo que ayuda a revelar configuraciones erróneas o servicios expuestos inadvertidamente [20].

Estas herramientas se fundamentan en los *counters* del kernel expuestos en las primitivas *cgroups*, pilares de la contabilidad de recursos en sistemas virtualizados. En análisis de la configuración y seguridad de red para garantizar la integridad de la capa de datos. El conjunto de estas utilidades articula una estrategia de monitorización de la superficie de ataque, fundamental para auditorías de seguridad y cumplimiento normativo.

6.4.3. Grafana

Grafana es una plataforma de análisis y monitoreo que permite visualizar métricas en tiempo real a través de paneles interactivos. Se integra con diversas fuentes de datos, incluyendo bases de datos de series temporales como Prometheus, InfluxDB y Graphite. Las características clave de Grafana son las siguientes:

- **Visualización de datos**: permite crear gráficos, tablas y otros elementos visuales para representar datos de manera efectiva.
- **Alertas**: soporta la configuración de alertas basadas en umbrales, enviando notificaciones a través de múltiples canales (correo electrónico, Slack, etc.).
- ***Dashboards* compartidos**: facilita la creación de dashboards que pueden ser compartidos entre equipos, promoviendo la colaboración.
- ***Plugins* y extensibilidad**: ofrece una amplia gama de plugins para integrar nuevas fuentes de datos y visualizaciones personalizadas.

Grafana se ha convertido en una herramienta esencial para la observabilidad en entornos de microservicios y arquitecturas distribuidas, permitiendo a los equipos de operaciones y desarrollo obtener información valiosa sobre el rendimiento y la salud de sus aplicaciones.

6.4.4. Prometheus

Prometheus es un sistema de monitoreo y alerta diseñado para recopilar y almacenar métricas en tiempo real. Utiliza un modelo de datos basado en series temporales y ofrece un lenguaje de consulta potente para analizar los datos recopilados. Las características clave de Prometheus son las siguientes:

- **Recopilación de métricas**: Prometheus puede recopilar métricas de aplicaciones y servicios a través de *endpoints* HTTP, utilizando un formato de exposición estándar.
- **Almacenamiento de series temporales**: las métricas se almacenan como series temporales, lo que permite realizar consultas eficientes y análisis a lo largo del tiempo.
- **Alertas basadas en consultas**: Prometheus permite definir reglas de alerta basadas en consultas, enviando notificaciones cuando se cumplen ciertas condiciones.
- **Integración con Grafana**: Prometheus se integra fácilmente con Grafana, lo que permite visualizar las métricas recopiladas en paneles interactivos.

6.5. Computación elástica

La computación elástica es un paradigma fundamental en el desarrollo de cómputo para nube, se refiere a la capacidad de una infraestructura de ajustarse dinámicamente ampliando o disminuyendo recursos computacionales en base a la demanda. Esta tecnología fue popularizada por proveedores de nube masivos tales como Amazon Web Services (AWS), Microsoft Azure y Google Cloud Platform (GCP), que ofrecen servicios que permiten a las organizaciones escalar sus recursos de manera eficiente y rentable. Formalizado por el National Institute of Standards and Technology (NIST), clasificándolo como una de las cinco características esenciales del modelo de computación en la nube.

Desde una perspectiva teórica, la elasticidad es vista como una manifestación del principio de adaptabilidad de los sistemas, permitiendo una respuesta autónoma, directa y en tiempo real a la demanda y fluctuaciones de la carga. Se diferencia de la escalabilidad, que es la capacidad potencial de crecimiento, ya que la elasticidad implica acciones programadas y automatizadas [21], [22].

6.5.1. Dimensiones de la elasticidad

La elasticidad en la computación en la nube se puede analizar desde varias dimensiones, cada una de las cuales aborda diferentes aspectos de la capacidad de adaptación de los recursos. Estas dimensiones son las siguientes:

- **Elasticidad vertical**: se refiere a la capacidad de aumentar o disminuir los recursos (CPU, memoria) de una instancia de máquina virtual en función de la demanda. Esto permite a las aplicaciones manejar picos de carga sin necesidad de aprovisionar nuevas instancias [6].
- **Elasticidad horizontal**: implica la adición o eliminación de instancias de máquinas virtuales en un clúster para distribuir la carga de trabajo. Esta dimensión es crucial para aplicaciones distribuidas y microservicios, donde la carga puede variar significativamente entre diferentes componentes [6].
- **Elasticidad temporal**: se refiere a la capacidad de ajustar los recursos a lo largo del tiempo, basándose en patrones de uso predecibles. Esto permite a las organizaciones optimizar costos al reducir recursos durante períodos de baja demanda y aumentarlos durante picos [23].
- **Elasticidad geográfica**: implica la capacidad de desplegar y escalar recursos en múltiples regiones geográficas. Esto es esencial para aplicaciones globales que requieren baja latencia y alta disponibilidad en diferentes ubicaciones [23].
- **Elasticidad en capas:** aplica el concepto de ajuste dinámico no solo al cómputo, sino también a almacenamiento, redes, bases de datos y otros, de forma integral [24].

6.5.2. Arquitectura de computación elástica en OpenStack

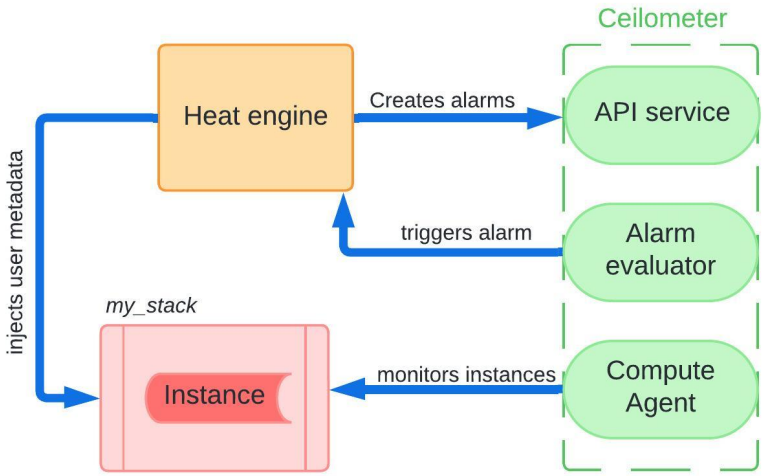
OpenStack no proporciona elasticidad de forma nativa o directamente a través de un solo servicio, sin embargo, ofrece un conjunto de componentes y herramientas necesarias para implementarla de forma modular.

- **Heat (orquestación):** permite definir y desplegar infraestructura como código mediante plantillas HOT. Puede describir la relación entre recursos, condiciones de escalamiento, dependencias, etc [15].
- **Ceilometer (telemetría):** recolecta y almacena métricas de uso de recursos como CPU, RAM, disco y tráfico de red. Estas métricas son usadas como disparadores para el escalamiento [16].
- **Aodh (alarmas):** componente especializado en generar alarmas basadas en las métricas recolectadas por Ceilometer o Gnocchi. Al activarse una alarma, se ejecutan políticas de escalamiento definidas en Heat.

Tomando como referencia el diagrama, se puede observar cómo los diferentes componentes interactúan para lograr una elasticidad efectiva en la infraestructura de OpenStack. Heat Engine (Motor de Heat) es el componente de orquestación que maneja la infraestructura definida en plantillas (como *my stack*). Aquí se gestionan las instancias y recursos desplegados. También inyecta los metadatos del usuario a las instancias, es decir, configura y personaliza las máquinas virtuales según sea necesario. Esta instancia es monitoreada para métricas de rendimiento y estado.

Ceilometer cuenta con tres componentes. Primero, API Service es el componente que recibe y procesa las peticiones para recopilar datos sobre métricas y eventos. Posteriormente, Alarm Evaluator evalúa las métricas recopiladas para detectar patrones en condiciones predefinidas como pueden ser

Figura 3. Diagrama *closed loop elasticity*.



Nota. Adaptado de [25].

umbrales disparadores de alarmas. Por último, el Compute Agent se encarga de monitorear instancias en ejecución, recogiendo métricas sobre el estado y rendimiento actual.

Heat Engine crea alarmas en el servicio API de Ceilometer para monitoreo constante. Ceilometer, mediante Compute Agent, monitorea las instancias activas; cuando se detecta una condición que cumple un criterio, Alarm Evaluator dispara una alarma. Las alarmas disparadas se envían de vuelta al Heat engine que la recibe como un disparador y, a partir de dicho disparador, Heat puede ejecutar acciones automáticas como escalar recursos [26].

Una arquitectura típica de elastic computing en OpenStack incluye los siguientes elementos:

- Un grupo de instancias *(auto scaling group)* definido en Heat.
- Una plantilla HOT que establece mínimos, máximos y políticas de crecimiento.
- Alarmas que monitorean, por ejemplo, si el promedio de CPU de todas las instancias supera el 70 % durante 5 minutos.
- Una acción de Heat que lanza una nueva instancia si la alarma se activa.

Este mecanismo, aunque simple, reproduce comportamientos que normalmente se asocian a soluciones empresariales avanzadas como AWS Auto Scaling o Google Cloud Instance Groups [26].

6.5.3. Retos y consideraciones técnicas

Implementar elastic computing implica también enfrentar ciertos retos:

- **Latencia en reacción:** las métricas se recolectan en intervalos (por ejemplo, cada 60 segundos), lo cual puede generar retraso entre la detección y la acción.
- **Carga de monitoreo:** Ceilometer puede generar carga significativa en el nodo controlador si no se limita el número de recursos monitoreados.
- **Persistencia de datos:** escalar horizontalmente una aplicación que requiere almacenamiento persistente (como una base de datos) requiere arquitecturas más complejas (e.g., uso de volúmenes compartidos, replicación) [27].

El desarrollo de un correcto sistema de elasticidad no solo es valorado por la industria, sino que representan una aproximación moderna e integral al diseño de infraestructura digital, alineada con tendencias como la ingeniería de plataformas, la observabilidad y la computación nativa en la nube [11].

CAPÍTULO 7

Virtualización multisistema

Para el desarrollo se asume una instalación básica del entorno OpenStack, descrita en el siguiente trabajo [28] , que muestra una instalación mediante kolla-ansible y docker para una red privada de OpenStack para dos computadoras de alto rendimiento, generando dos nodos computacionales y un nodo controlador. Es importante asegurar este despliegue para contar con todos los servicios necesarios para la virtualización de sistemas operativos.

También es necesario el usuario *root* con privilegios administrativos para la instalación de paquetes y configuración del entorno así como el usuario administrativo de OpenStack con permisos para la creación y gestión de instancias, imágenes y otros recursos dentro del entorno de nube. Para el caso de este despliegue mediante contenedores también es necesario activar el ambiente de python virtualenv para la ejecución de comandos de OpenStack y el archivo de credenciales para autenticar el usuario y permitir el acceso de modificar los servicios de OpenStack.

Cuadro 2. Activación de usuario root, entorno virtual y archivo de credenciales de OpenStack para virtualización multisistema

```
# activar usuario root
sudo su
password: openstack
# activar entorno virtual
source ~/openstack/os-venv/bin/activate
# cargar archivo de credenciales
source etc/kolla/admin-openrc.sh
```

Nota. Elaboración propia.

7.1. Virtualización de sistemas Linux Server

La virtualización de sistemas Linux Server consiste en desplegar máquinas virtuales que ejecutan distribuciones de Linux optimizadas para tareas de servidor. Estas distribuciones, como Ubuntu Server, Rocky Linux o Fedora, están diseñadas para ofrecer estabilidad, seguridad y eficiencia en el uso de recursos. Mediante la virtualización, es posible ejecutar múltiples servidores independientes sobre un mismo *hardware* físico, lo que permite aprovechar mejor los recursos disponibles y simplificar la administración.

Entre las herramientas más utilizadas para la virtualización de servidores Linux se encuentran KVM *(Kernel-based Virtual Machine)*, QEMU *(Quick Emulator)* y plataformas de gestión como OpenStack. Estas soluciones permiten crear, administrar y escalar instancias de servidores de manera flexible y eficiente, facilitando la automatización y el aprovisionamiento rápido de nuevos servicios.

7.1.1. Cirros

Cirros es una distribución extremadamente ligera y minimalista de Linux, está diseñada específicamente para pruebas y desarrollo en entornos de virtualización y nube. Su tamaño de imagen es reducido, creando un tiempo de arranque muy rápido, siendo ideal en situaciones donde se requieren instancias temporales para validar configuraciones, probar funcionalidades o pruebas piloto. Cirros incluye un conjunto básico de herramientas y utilidades, lo que permite a los desarrolladores y administradores de sistemas realizar pruebas rápidas sin la sobrecarga de una distribución completa. Su simplicidad y eficiencia la convierten en una opción popular para entornos de desarrollo y pruebas en la nube.

Obtención de imagen .img

Para obtener la imagen de Cirros, se puede descargar desde el sitio oficial del proyecto o utilizar herramientas como **wget** para obtenerla directamente desde la línea de comandos. A continuación, se muestra un ejemplo de cómo descargar la imagen de Cirros y prepararla para su uso en OpenStack.

Cuadro 3. Descarga de imagen Cirros con Wget

```
mkdir -p ~/images
wget https://cloud-images.ubuntu.com/releases/cirros/current/\
  cirros-0.5.2-x86_64-disk.img -O ~/images/cirros.img
```

Nota. Elaboración propia.

Registro de imagen en Glance

Una vez descargada la imagen, se debe registrar en OpenStack utilizando el comando **openstack image create**. Este comando permite crear una nueva imagen en el servicio de imágenes de OpenStack (Glance) y especificar sus propiedades, como el formato del disco y la visibilidad pública. A continuación, se muestra un ejemplo de cómo registrar la imagen de Cirros:

Cuadro 4. Registro de imagen Cirros en Glance

```
# registrar imagen en glance
openstack image create \
  --public \
  --container-format bare \
  --disk-format qcow2 \
  --file cirros.img \
  cirros
```

Nota. Elaboración propia.

Una vez registrada, deberá aparecer un cuadro con la información de la imagen creada. Para verificar que la imagen se ha registrado, se puede utilizar el comando **openstack image list**, que mostrará todas las imágenes disponibles en el proyecto.

Cuadro 5. Cuadro informativo de Cirros en Glance

```
| Field            | Value                                                      |
+------------------+------------------------------------------------------------+
| checksum         | c8fc807773e5354afe61636071771906                           |
| container_format | bare                                                       |
| created_at       | 2025-09-10T16:32:22Z                                       |
| disk_format      | qcow2                                                      |
| file             | /v2/images/f40da6c9-36f6-4fea-80e1-0a8dc8865bfc/file       |
| id               | f40da6c9-36f6-4fea-80e1-0a8dc8865bfc                       |
| min_disk         | 0                                                          |
| min_ram          | 0                                                          |
| name             | cirros                                                     |
| owner            | 5260c7767b584c58a1da728373b2d8f3                           |
| properties       | os_hash_algo='sha512', os_hash_value='1103b92ce8ad96       |
|                  | 6e41235a4de260deb791ff571670c0342666c8582fbb9caefe6a       |
|                  | f07ebb11d34f44f8414b609b29c1bdf1d72ffa6faa39c88e8721       |
|                  | d09847952b', os_hidden='False',                            |
|                  | owner_specified.openstack.md5='',                          |
|                  | owner_specified.openstack.object='images/cirros',          |
|                  | owner_specified.openstack.sha256='', stores='file'         |
| protected        | False                                                      |
| schema           | /v2/schemas/image                                          |
| size             | 21430272                                                   |
```

```
| status           | active                                                   |
| tags             |                                                          |
| updated_at       | 2025-09-10T16:32:22Z                                     |
| virtual_size     | 117440512                                                |
| visibility       | public                                                   |
```

Nota. Elaboración propia.

Creación de flavor

Un *flavor* en OpenStack define los recursos asignados a una instancia, como la cantidad de RAM, CPU y espacio en disco. Tomando en cuenta que cirros es una imagen que requiere muy pocos recursos, se creará un *flavor* con las especificaciones mínimas posibles. Para crear un *flavor* adecuado para Cirros, se puede utilizar el siguiente comando:

Cuadro 6. Creación de flavor para Cirros en OpenStack

```
# crear flavor
openstack flavor create cirros.small \
  --ram 256 --disk 1 --vcpus 1 \
  --swap 0 --ephemeral 0 \
  --property os_distro=cirros \
  --property os_version=0.5.2
```

Nota. Elaboración propia.

Posterior a la creación del *flavor*, debe de aparecer un cuadro de texto de confirmación de creación como el siguiente, de no aparecer puede consultarse mediante **openstack flavor show**.

Cuadro 7. Cuadro informativo de flavor cirros.small

```
| Field                       | Value                                      |
+-----------------------------+--------------------------------------------+
| OS-FLV-DISABLED:disabled    | False                                      |
| OS-FLV-EXT-DATA:ephemeral   | 0                                          |
| description                 | None                                       |
| disk                        | 1                                          |
| id                          | f805f373-39e8-45c4-b9a0-582425f0bbfe       |
| name                        | cirros.small                               |
| os-flavor-access:is_public  | True                                       |
| properties                  | os_distro='cirros', os_version='0.5.2'     |
| ram                         | 256                                        |
| rxtx_factor                 | 1.0                                        |
| swap                        | 0                                          |
| vcpus                       | 1                                          |
```

Nota. Elaboración propia.

Lanzamiento de instancia

Una vez que la imagen y el *flavor* están configurados, se puede lanzar una instancia de Cirros utilizando el comando **openstack server create**. Este comando permite especificar la imagen, el *flavor* y otros parámetros necesarios para crear la instancia.

Cuadro 8. Lanzamiento de instancia Cirros en OpenStack

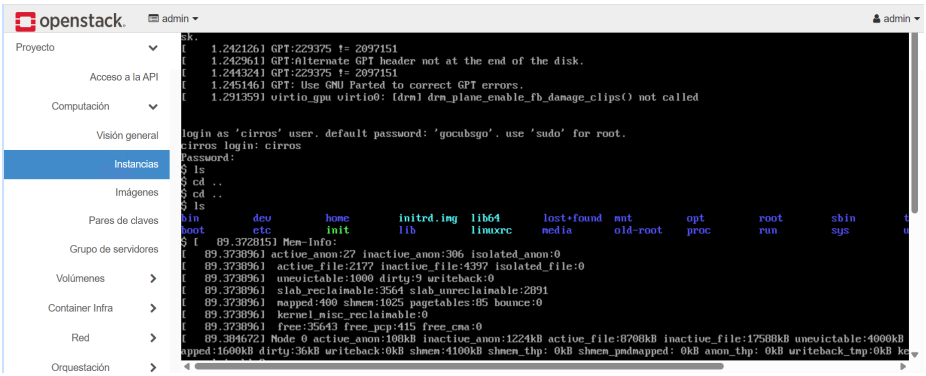
```
openstack server create --image cirros --flavor cirros.small --key-name mykey
    ↪ --security-group default mycirros
```

Nota. Elaboración propia.

Para verificar que la instancia se ha creado correctamente, se puede utilizar el comando **openstack server list**, que mostrará todas las instancias activas en el proyecto. El acceso a la máquina virtual se encuentra en el servicio de Horizon, en la pestaña de instancias, abriendo la consola de la instancia correspondiente. Las credenciales por defecto para iniciar sesión en la instancia son las siguientes:

- Usuario: cirros
- Contraseña: gocubsgo

Figura 4. Máquina virtual Cirros.



Nota. Elaboración propia.

Para eliminar la instancia, se puede utilizar el comando **openstack server delete**, seguido del nombre o ID de la instancia.

Cuadro 9. Eliminación de instancia Cirros en OpenStack

```
openstack server delete mycirros
```

Nota. Elaboración propia.

7.1.2. Ubuntu Server 22.04.03

Ubuntu Server 22.04 es una distribución de Linux optimizada para servidores, basada en la versión LTS *(Long Term Support)* de Ubuntu. Esta versión ofrece un entorno estable y seguro, ideal para implementar aplicaciones y servicios en la nube o en entornos locales. Ubuntu Server 22.04 incluye mejoras en la gestión de contenedores, soporte para *hardware* moderno y actualizaciones de seguridad regulares, lo que la convierte en una opción popular para empresas y desarrolladores que buscan una plataforma confiable para sus cargas de trabajo.

Se optó por Ubuntu Server 22.04 debido a su amplia adopción en la industria, su facilidad de uso y su compatibilidad con una gran cantidad de herramientas y servicios. Además, al ser una versión LTS, garantiza soporte a largo plazo, lo que es crucial para entornos de producción donde la estabilidad y la seguridad son prioritarias.

Obtención de imagen .img

Las imagenes oficiales de Ubuntu server pueden ser recuperadas desde su sitio Web oficial [29]. En este caso, se utilizará la versión 22.04.3 LTS, haciendo referencia a la versión de servidor. El sitio Web cuenta con múltiples formatos, sin embargo, la imagen en formato *.iso* es la más adecuada para su uso en entornos de virtualización. A continuación. Una vez descargada la imagen, es necesario bootearla en una máquina virtual para realizar la configuración inicial del sistema operativo, como la creación de usuarios y la instalación de actualizaciones. Posterior a la configuración inicial, se debe exportar la imagen en formato *.vmdk* para su posterior carga en OpenStack.

Conversión de imagen

Para convertir la imagen *.vmdk* a *.qcow2*, se puede utilizar la herramienta QEMU-img. Esta herramienta permite convertir imágenes de disco entre diferentes formatos, optimizando el tamaño y el rendimiento de las imágenes en OpenStack. A continuación, se muestra cómo instalar QEMU-utils.

Cuadro 10. Instalación de herramientas QEMU en Linux

```
# descargar herramientas de QEMU
apt-get update && apt-get install -y qemu-utils
```

Nota. Elaboración propia.

Una vez instaladas las herramientas de QEMU, se puede proceder a la conversión de la imagen utilizando el siguiente comando:

Cuadro 11. Conversión de imagen .vmdk a .qcow2 para Ubuntu Server 22.04

```
qemu-img convert -p -f vmdk -O qcow2 \
  "Ubuntu 64-serv.vmdk" ubuntu_serv.qcow2
```

Nota. Elaboración propia.

Registro de imagen en Glance

Una vez descargada la imagen, se debe registrar en OpenStack utilizando el comando **openstack image create**. Este comando permite crear una nueva imagen en el servicio de imágenes Glance y especificar sus propiedades. A continuación, se muestra un ejemplo de cómo registrar la imagen de Ubuntu Server 22.04:

Cuadro 12. Registro de imagen Ubuntu Server 22.04 en Glance

```
# registrar imagen en glance
openstack image create \
  --public \
  --container-format bare \
  --disk-format qcow2 \
  --file ubuntu_serv.qcow2 \
  ubuntu-22.04-server
```

Nota. Elaboración propia.

Una vez registrada la imagen, deberá aparecer un cuadro con la información de la imagen creada, incluyendo su ID, nombre y estado. Para verificar que la imagen se ha registrado correctamente, se puede utilizar el comando **openstack image list**, que mostrará todas las imágenes disponibles en el proyecto.

Cuadro 13. Cuadro informativo de Ubuntu Server 22.04 en Glance

```
+------------------+------------------------------------------------------------+
| Field            | Value                                                      |
+------------------+------------------------------------------------------------+
| checksum         | 778bcef0e8248b737ca4d63c62bf9f5a                           |
| container_format | bare                                                       |
| created_at       | 2025-08-11T04:53:59Z                                       |
| disk_format      | qcow2                                                      |
| file             | /v2/images/6b167977-ce95-4dfd-9052-a54c8a41f8aa/file       |
| id               | 6b167977-ce95-4dfd-9052-a54c8a41f8aa                       |
| min_disk         | 0                                                          |
| min_ram          | 0                                                          |
| name             | ubuntu-22.04-server                                        |
| owner            | 431b4b0e9d594207a4ef1df4bbe8463a                           |
| properties       | os_hash_algo='sha512', os_hash_value='9d87a577d16cf52e15f  |
|                  | 3d2887a23b51f64146a51c36bff957968a5118134c4c9f5a6f9079b87  |
|                  | 14a9f1ef816e3d837e22105bd23e92cf9c37ec9cffd9e9acf584',     |
|                  | os_hidden='False', owner_specified.openstack.md5='', owne  |
|                  | r_specified.openstack.object='images/ubuntu-22.04-         |
|                  | server', owner_specified.openstack.sha256='',              |
|                  | stores='file'                                              |
| protected        | False                                                      |
```

```
| schema           | /v2/schemas/image                                                |
| size             | 677302272                                                        |
| status           | active                                                           |
| tags             |                                                                  |
| updated_at       | 2025-08-11T04:54:03Z                                             |
| virtual_size     | 2361393152                                                       |
| visibility       | public                                                           |
+------------------+------------------------------------------------------------------+
```

Nota. Elaboración propia.

Creación de flavor

Un *flavor* en OpenStack define los recursos asignados a una instancia, como la cantidad de RAM, CPU y espacio en disco. Para crear un *flavor* adecuado para Ubuntu Server 22.04, se puede utilizar el siguiente comando:

Cuadro 14. Creación de flavor para Ubuntu Server 22.04 en OpenStack

```
# crear flavor
openstack flavor create ubuntu-server.small \
  --ram 2048 --disk 20 --vcpus 1 \
  --swap 0 --ephemeral 0 \
  --property os_distro=ubuntu \
  --property os_version=22.04
```

Nota. Elaboración propia.

Posterior a la creación del *flavor*, debe de aparecer un cuadro de texto de confirmación de creación como el siguiente, de no aparecer puede consultarse mediante el comando **openstack flavor show ubuntu-server.small**.

Cuadro 15. Cuadro informativo de flavor ubuntu-server.small

```
+----------------------------+-----------------------------------------+
| Field                      | Value                                   |
+----------------------------+-----------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                   |
| OS-FLV-EXT-DATA:ephemeral  | 0                                       |
| description                | None                                    |
| disk                       | 20                                      |
| id                         | auto                                    |
| name                       | ubuntu-server.small                     |
| os-flavor-access:is_public | True                                    |
| properties                 | os_distro='ubuntu', os_version='22.04'  |
| ram                        | 2048                                    |
| rxtx_factor                | 1.0                                     |
```

```
| swap                         | 0                                          |
| vcpus                        | 1                                          |
+------------------------------+--------------------------------------------+
```

Nota. Elaboración propia.

Lanzamiento de instancia

Una vez que la imagen y el *flavor* están configurados, se puede lanzar una instancia de Ubuntu Server 22.04 utilizando el comando **openstack server create**. Este comando permite especificar la imagen, el *flavor* y otros parámetros necesarios para crear la instancia.

Cuadro 16. Lanzamiento de instancia Ubuntu Server 22.04 en OpenStack

```
# lanzar instancia
openstack server create \
  --flavor ubuntu-server.small \
  --image ubuntu-22.04-server \
  --network public \
  --key-name mykey \
  --security-group default \
    ubuntu-server-22.04
```

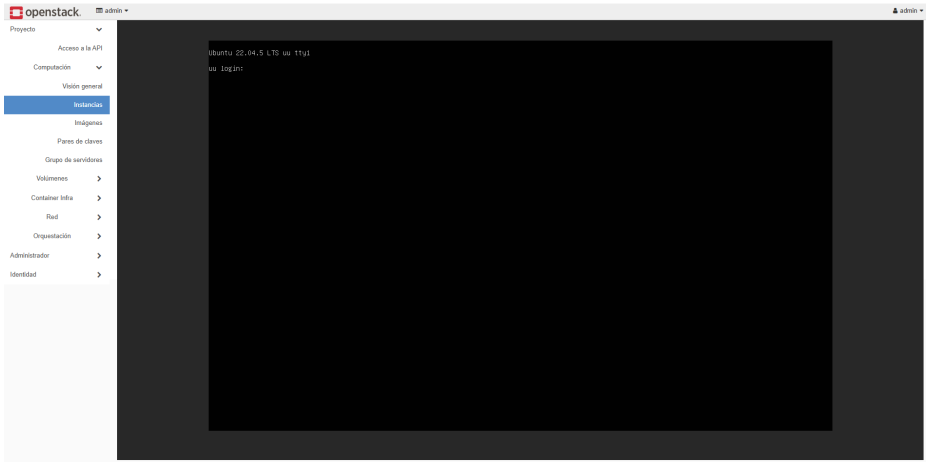
Nota. Elaboración propia.

Para verificar que la instancia se ha creado correctamente, se puede utilizar el comando **openstack server list**, que mostrará todas las instancias activas en el proyecto. El acceso a la máquina virtual se encuentra en el servicio de Horizon, en la pestaña de instancias, abriendo la consola de la instancia correspondiente.

Las credenciales para iniciar sesión en la instancia son las que se configuraron durante la instalación inicial del sistema operativo en la máquina virtual antes de exportarla. En este caso, se utilizó el siguiente usuario y contraseña:

- Usuario: ubuntu
- Contraseña: openstack

Figura 5. Máquina virtual Ubuntu Server 22.04.



Nota. Elaboración propia.

Para eliminar la instancia, se puede utilizar el comando **openstack server delete**, seguido del nombre o ID de la instancia.

7.1.3. Fedora server

Fedora Server es una distribución de Linux desarrollada por la comunidad Fedora, que se enfoca en ofrecer un entorno robusto y flexible para servidores. Esta distribución es conocida por su innovación y adopción temprana de nuevas tecnologías, lo que la convierte en una opción atractiva para administradores de sistemas y desarrolladores que buscan un sistema operativo moderno y eficiente. Fedora Server incluye herramientas avanzadas de administración, soporte para contenedores y virtualización, así como una amplia gama de paquetes y aplicaciones optimizadas para entornos de servidor.

La elección de Fedora Server para la virtualización de servidores se debe a su enfoque en la seguridad, la estabilidad y la facilidad de uso. Fedora Server ofrece actualizaciones regulares y soporte para las últimas tecnologías, lo que permite a los usuarios mantenerse al día con las tendencias del mercado y aprovechar las mejoras en el rendimiento y la seguridad. Además, Fedora Server cuenta con una comunidad activa que proporciona soporte y recursos, facilitando la resolución de problemas y la implementación de soluciones personalizadas.

Obtención de imagen .vdi

Las imagenes oficiales de Fedora server pueden ser recuperadas desde su sitio Web oficial [30]. En este caso, se utilizará la versión *Server*, haciendo referencia a la versión de servidor y más ligera, sin interfaz gráfica. El sitio Web cuenta con múltiples formatos, sin embargo, la imagen en formato *.vdi* es la más adecuada para su uso en entornos de virtualización. Posterior a la descarga de la imagen es necesario virtualizarla en VirtualBox o Vmware para realizar su *booting* y configuración inicial para

mayor comodidad al virtualizar en OpenStack. Una vez configurada la imagen, se debe exportar en formato *.vdi* en el caso de VirtualBox o *.vmdk* para Vmware, para su posterior conversión y carga en OpenStack.

Conversión de imagen

Para convertir la imagen *.vmdk* a *.qcow2*, se utiliza la herramienta QEMU-img para asegurar su formato ideal en Openstack y optimización del tamaño de imágen, así como rendimiento de la imagen. A continuación, se muestra cómo realizar la conversión de imagen.

Cuadro 17. Conversión de .vmdk a .qcow2 para Fedora Server

```
# Convertir imagen .vmdk a .qcow2
qemu-img convert -f vmdk -O qcow2 "Fedora server 64-bit.vmdk" fedora-server.qcow2
```

Nota. Elaboración propia.

Registro de imagen en Glance

La imagen puede ser registrada una vez se tenga el archivo en formato qcow2, este formato es el recomendado para su uso en entornos de virtualización.

Cuadro 18. Registro de imagen Fedora Server en Glance

```
# registrar imagen en glance
openstack image create \
  --public \
  --container-format bare \
  --disk-format qcow2 \
  --file fedora-server.qcow2 \
  fedora-server
```

Nota. Elaboración propia.

Cuadro 19. Cuadro informativo de Fedora Server en Glance

```
+------------------+------------------------------------------------------------------+
| Field            | Value                                                            |
+------------------+------------------------------------------------------------------+
| checksum         | 67ed3cfc69eb1561de3ce9e6b7943821                                 |
| container_format | bare                                                             |
| created_at       | 2025-10-17T23:27:26Z                                             |
| disk_format      | qcow2                                                            |
| file             | /v2/images/2c3a3344-4ad0-48ea-8a71-c44fc522919a/file             |
| id               | 2c3a3344-4ad0-48ea-8a71-c44fc522919a                             |
| min_disk         | 0                                                                |
| min_ram          | 0                                                                |
```

```
| name              | fedora-server                                                    |
| owner             | 5260c7767b584c58a1da728373b2d8f3                                 |
| properties        | os_hash_algo='sha512', os_hash_value='dc36e4a3c7792ee37e089      |
|                   | f0109064450f38a2e3d1ab07a081cb7d7f787653f1b63153fa0b9b47c54      |
|                   | 9b6aa011dd7e3921542f2f6526a89c98fa8358712d61a6d3',               |
|                   | os_hidden='False', owner_specified.openstack.md5='',             |
|                   | owner_specified.openstack.object='images/fedora-server',         |
|                   | owner_specified.openstack.sha256='', stores='file'               |
| protected         | False                                                            |
| schema            | /v2/schemas/image                                                |
| size              | 2116026368                                                       |
| status            | active                                                           |
| tags              |                                                                  |
| updated_at        | 2025-10-17T23:27:39Z                                             |
| virtual_size      | 32212254720                                                      |
| visibility        | public                                                           |
+-------------------+------------------------------------------------------------------+
```

Nota. Elaboración propia.

Creación de flavor

Una vez que la imagen ha sido registrada en Glance, se puede proceder a la creación de un *flavor* que se ajuste a las necesidades de la instancia que se va a desplegar. Las especificaciones recomendadas para un Fedora server mínimo son 4 GB de RAM, 20 GB de disco y 1 vCPU.

Cuadro 20. Creación de flavor para Fedora Server en OpenStack

```
# crear flavor
openstack flavor create --ram 4096 --disk 30 --vcpus 1 fedora-server.small
```

Nota. Elaboración propia.

Cuadro 21. Cuadro informativo de flavor fedora-server.small

```
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| description                | None                                 |
| disk                       | 30                                   |
| id                         | af1d44f1-fc42-4484-b3a0-b676947644bd |
| name                       | fedora-server.small                  |
| os-flavor-access:is_public | True                                 |
| properties                 |                                      |
```

```
| ram                         | 4096                                    |
| rxtx_factor                 | 1.0                                     |
| swap                        | 0                                       |
| vcpus                       | 1                                       |
+-----------------------------+-----------------------------------------+
```

Nota. Elaboración propia.

Lanzamiento de instancia

Para lanzar una instancia de Fedora Server con el *flavor* creado, se puede utilizar el siguiente comando, especificando la imagen, el *flavor*, la clave SSH y el grupo de seguridad. Esta instancia se nombrará como *fedora-server-instance* y se puede acceder a ella a través de la consola de Horizon o mediante SSH utilizando la clave especificada.

Cuadro 22. Lanzamiento de instancia de Fedora Server

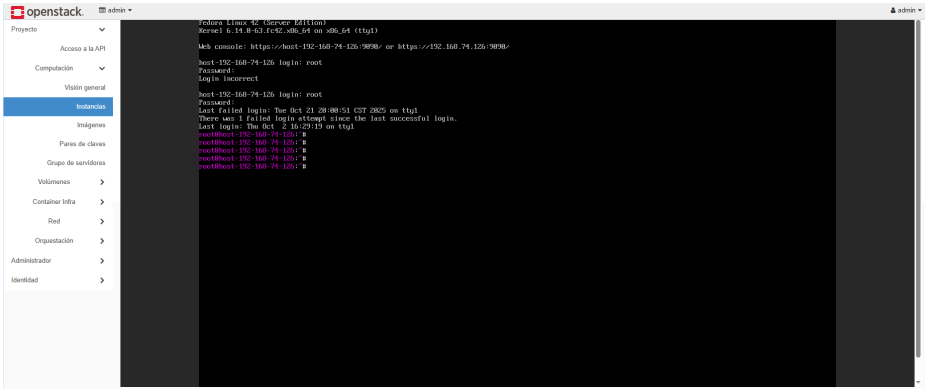
```
openstack server create --flavor fedora-server.small --image fedora-server --key-name
    ↪ mykey --security-group default fedora-server-instance
```

Nota. Elaboración propia.

Una vez que la instancia se ha creado, se puede verificar su estado utilizando el comando **openstack server list**. La instancia debería aparecer en estado *Active* si se ha creado correctamente. Luego aparecerá la terminal de comandos en la sección de instancias y son las siguientes en la sección de consola. Las credenciales por defecto para iniciar sesión en la instancia son las siguientes:

- Usuario: root
- Contraseña: openstack

Figura 6. Máquina virtual Fedora Server.



Nota. Elaboración propia.

7.2. Virtualización de sistemas Linux Desktop

Para la virtualización de sistemas Linux Desktop será necesario contar con imágenes de sistemas operativos en formatos compatibles con OpenStack, como *qcow2* o *vdi*. Estas imágenes se pueden obtener de diversas fuentes, como repositorios oficiales de distribución o mediante herramientas de conversión. A continuación, se describen los pasos para virtualizar diferentes distribuciones de Linux Desktop, incluyendo Ubuntu Desktop 22.04, Ubuntu Desktop 24.04.2, Rocky Linux Desktop 10 y Fedora Desktop.

Para obtener una imagen en formato .vdi se debe crear una máquina virtual en VirtualBox y luego exportar el archivo al nodo de Openstack donde se subirá la imagen a Glance. En la sección de anexos se provee una guía detallada de cómo crear una máquina virtual en VirtualBox y Vmware Workstation para posteriormente exportar la imagen en formato *.vdi* o *.vmdk* dependiendo de la herramienta utilizada.

7.2.1. Ubuntu desktop 22.04

Ubuntu Desktop 22.04 es una distribución altamente demandada y popular gracias a su facilidad de uso, amplia comunidad de soporte y características de Linux avanzadas. Contiene una interfaz de usuario muy amigable y cuenta con herramientas perfectas para desarrolladores y usuarios finales. Esta versión incluye mejoras en el rendimiento, actualizaciones de seguridad y nuevas características que la hacen ideal para entornos de escritorio.

La imagen de Ubuntu Desktop 22.04 en formato .iso se puede descargar desde el sitio oficial de Ubuntu [29]. Esta imagen debe de pasar por un proceso de conversión de varios pasos para ser utilizada y optimizada para Openstack.

Primero, se debe crear una máquina virtual en VirtualBox utilizando la imagen *.iso* de Ubuntu Desktop 22.04. Durante la creación de la máquina virtual, se recomienda asignar al menos 4 GB de RAM y 40 GB de disco duro para garantizar un rendimiento adecuado del sistema operativo.

Una vez que se tiene la imagen en formato *.vdi*, se debe convertir a un formato compatible con OpenStack, como *.qcow2*. Para esto, se puede utilizar la herramienta *qemu-img*, que permite convertir imágenes de disco entre diferentes formatos. Esta herramienta nos permitirá realizar la conversión de imagen en formato *.vdi* a *.qcow2*, que es el formato recomendado para OpenStack.

Conversión de imagen

Cuadro 23. Conversión de imagen .vdi a .qcow2 para Ubuntu Desktop 22.04

```
# Convertir imagen .vdi a .qcow2
qemu-img convert -c -p -f vdi -O qcow2 Ubuntu_desktop22.04.vdi
    ↪ Ubuntu_desktop22.04.qcow2
```

Nota. Elaboración propia.

Registro de imagen en Glance

Posterior a realizar la conversión, es necesario registrar la imagen para reconocimiento de los servicios de Openstack. Para esto, se utiliza el comando **openstack image create**, especificando el formato de disco y la visibilidad de la imagen.

Cuadro 24. Registro de imagen Ubuntu Desktop 22.04 en Glance

```
# Registrar imagen en OpenStack
openstack image create \
  --public \
  --container-format bare \
  --disk-format qcow2 \
  --file Ubuntu_desktop_22.04.qcow2 \
  ubuntu-desktop-22.04
```

Nota. Elaboración propia.

A continuación, se muestra un cuadro con la información de la imagen creada, incluyendo su ID, nombre y estado.

Cuadro 25. Cuadro informativo de Ubuntu Desktop 22.04 en Glance

```
+------------------+------------------------------------------------+
| Field            | Value                                          |
+------------------+------------------------------------------------+
| checksum         | 216b8172e2297e86fd9b8d0d7ce736c7               |
| container_format | bare                                           |
| created_at       | 2025-08-20T17:07:24Z                           |
| disk_format      | qcow2                                          |
| file             | /v2/images/0443dd05-85b1-42da-8d83-            |
|                  | f48710969e1b/file                              |
| id               | 0443dd05-85b1-42da-8d83-f48710969e1b           |
| min_disk         | 0                                              |
| min_ram          | 0                                              |
| name             | ubuntu-desktop-22.04                           |
| owner            | 431b4b0e9d594207a4ef1df4bbe8463a               |
| properties       | os_hash_algo='sha512', os_hash_value='ff5ed4d  |
|                  | 4e4d716c6ca5f1ba3480fc206a511e6f9c5b70271a95c  |
|                  | c451828fc38d4c6d2e261ea236e936995c102f4a5dd22  |
|                  | a31f9aed7001f73405651812fedbb46',              |
|                  | os_hidden='False',                             |
|                  | owner_specified.openstack.md5='', owner_speci  |
|                  | fied.openstack.object='images/ubuntu-          |
|                  | desktop-22.04',                                |
|                  | owner_specified.openstack.sha256='',           |
|                  | stores='file'                                  |
```

```
| protected        | False                                          |
| schema           | /v2/schemas/image                              |
| size             | 5280833024                                     |
| status           | active                                         |
| tags             |                                                |
| updated_at       | 2025-08-20T17:07:58Z                           |
| virtual_size     | 21474836480                                    |
| visibility       | public                                         |
+------------------+------------------------------------------------+
```

Nota. Elaboración propia.

Creación de flavor

Por último, antes de la creación de la instancia, es necesario configurar un *flavor* que sea adecuado para el sistema operativo que se está virtualizando. Para Ubuntu Desktop 22.04, creamos un *flavor* con recursos mínimos que aún permitan una experiencia fluida, para ello se utiliza el comando **openstack flavor create**.

Cuadro 26. Creación de flavor para Ubuntu Desktop 22.04 en OpenStack

```
# Crear flavor para Ubuntu Desktop 22.04
openstack flavor create ubuntu-desktop.small --id auto --ram 4096 --disk 40 --vcpus 2
```

Nota. Elaboración propia.

Cuadro 27. Cuadro informativo de flavor ubuntu-desktop.small

```
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| description                | None                                 |
| disk                       | 40                                   |
| id                         | 22fc28da-42d8-45a3-b83a-636ebd7b0329 |
| name                       | ubuntu-desktop.small                 |
| os-flavor-access:is_public | True                                 |
| properties                 |                                      |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       | 0                                    |
| vcpus                      | 2                                    |
```

Nota. Elaboración propia.

Este *flavor* está configurado para proporcionar a la instancia 4 GB de RAM, 40 GB de disco y 2 vCPU, para una experiencia correcta con el sistema operativo.

Lanzamiento de instancia

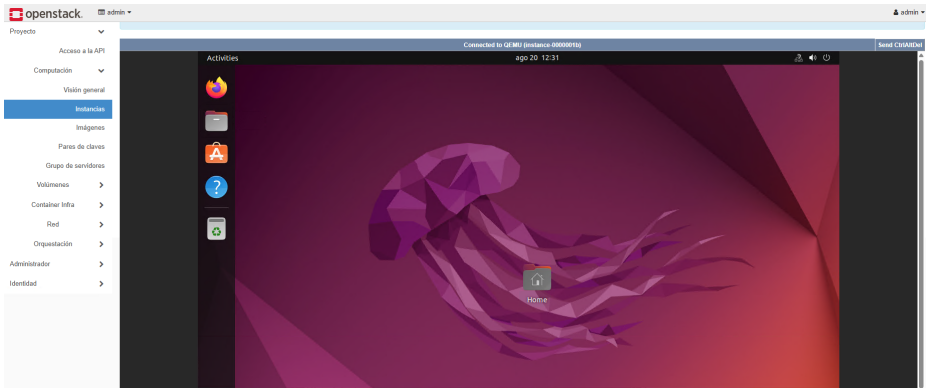
Para crear la instancia se puede realizar con dos métodos, mediante la interfáz gráfica de Horizon en la sección de instancias, o como en este caso, mediante la línea de comandos con el comando **openstack server create**.

Cuadro 28. Lanzamiento de instancia Ubuntu Desktop 22.04 en OpenStack

```
# Lanzar instancia de Ubuntu Desktop 22.04
openstack server create \
  --flavor ubuntu-desktop.small \
  --image ubuntu-desktop-22.04 \
  --network demo-net \
  --key-name mykey \
  --security-group default \
  ubuntu-desktop-22.04-instance
```

Nota. Elaboración propia.

Figura 7. Máquina virtual Ubuntu Desktop 22.04.



Nota. Elaboración propia.

7.2.2. Ubuntu desktop 24.04.2

Ubuntu 24.04.2 es la última versión de la serie de lanzamientos de Ubuntu Desktop. Esta versión incluye mejoras en la interfaz de usuario, actualizaciones de seguridad y soporte para *hardware* más reciente.

Al ser la distribución más popular de Linux, se optó por virtualizar esta versión también para aprovechar sus características avanzadas y su amplia compatibilidad con aplicaciones y herramientas de desarrollo. La imagen de Ubuntu Desktop 24.04.2 en formato .iso se puede descargar desde el sitio oficial de Ubuntu [29].

Conversión de imagen

Nuevamente se necesita convertir la imagen de Ubuntu Desktop 24.04.2 de formato *.iso* a *.qcow2* para su posterior carga a OpenStack.

Cuadro 29. Conversión de imagen .iso a .qcow2 para Ubuntu Desktop 24.04.2

```
# Convertir imagen .iso a .qcow2
qemu-img convert -f raw -O qcow2 ubuntu-24.04.2-desktop.iso
    ↪ ubuntu-24.04.2-desktop.qcow2
```

Nota. Elaboración propia.

Registro de imagen en Glance

Para registrar la imagen en Glance, se utiliza el siguiente comando:

Cuadro 30. Registro de imagen Ubuntu Desktop 24.04.2 en Glance

```
# Registrar imagen en OpenStack
openstack image create \
  --public \
  --container-format bare \
  --disk-format qcow2 \
  --file Ubuntu_desktop_24.04.2.qcow2 \
  ubuntu-desktop-24.04.2
```

Nota. Elaboración propia.

Cuadro 31. Cuadro informativo de Ubuntu Desktop 24.04.2 en Glance

```
+------------------+------------------------------------------------+
| Field            | Value                                          |
+------------------+------------------------------------------------+
| checksum         | 13d38f203086f18b162c1be436bbfe14               |
| container_format | bare                                           |
| created_at       | 2025-08-20T18:38:38Z                           |
| disk_format      | qcow2                                          |
| file             | /v2/images/93cb66ef-4693-48dc-                 |
|                  | bc76-2ca23307dd9a/file                         |
| id               | 93cb66ef-4693-48dc-bc76-2ca23307dd9a           |
| min_disk         | 0                                              |
| min_ram          | 0                                              |
| name             | ubuntu-desktop-24.04.2                         |
| owner            | 431b4b0e9d594207a4ef1df4bbe8463a               |
| properties       | os_hash_algo='sha512', os_hash_value='8a75c09  |
|                  | 5a1009f92fa6cab3685456e8aa5e45c62b4d517751fd4  |
```

```
|                  | 8892de7ef73c449f79915d670fee46b8921ef2624eb02 |
|                  | 1603b71994eb84e9250030fe78ee67b',             |
|                  | os_hidden='False',                            |
|                  | owner_specified.openstack.md5='', owner_speci |
|                  | fied.openstack.object='images/ubuntu-         |
|                  | desktop-24.04.2',                             |
|                  | owner_specified.openstack.sha256='',          |
|                  | stores='file'                                 |
| protected        | False                                         |
| schema           | /v2/schemas/image                             |
| size             | 6998523904                                    |
| status           | active                                        |
| tags             |                                               |
| updated_at       | 2025-08-20T18:39:22Z                          |
| virtual_size     | 10737418240                                   |
| visibility       | public                                        |
+------------------+-----------------------------------------------+
```

Nota. Elaboración propia.

Para el flavor, se puede utilizar el mismo que se creó para Ubuntu Desktop 22.04, ya que las especificaciones son adecuadas para ambas versiones y no hay mucha variabilidad en cuanto a procesamiento entre una versión y otra.

Lanzamiento de instancia

Para lanzar una instancia de Ubuntu Desktop 24.04.2, se puede utilizar el siguiente comando:

Cuadro 32. Lanzamiento de instancia Ubuntu Desktop 24.04.2 en OpenStack

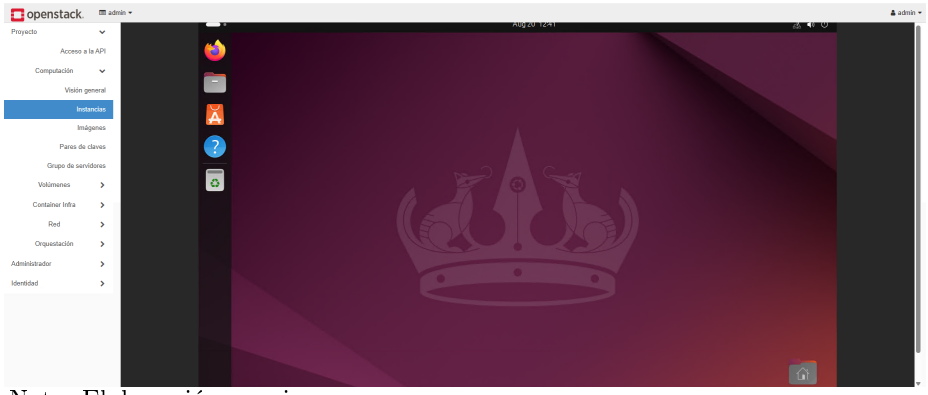
```
# Lanzar instancia en OpenStack
openstack server create \
  --flavor ubuntu-desktop.small \
  --image ubuntu-desktop-24.04.2 \
  --network private \
  ubuntu-desktop-24.04.2-instance
```

Nota. Elaboración propia.

Las credenciales por defecto para iniciar sesión en la instancia son las siguientes:

- Usuario: uvg
- Contraseña: openstack

Figura 8. Máquina virtual Ubuntu Desktop 24.04.2.



Nota. Elaboración propia.

7.2.3. Manjaro Gnome desktop 25.0.5

Manjaro Gnome es una distribución de Linux basada en Arch Linux, que utiliza el entorno de escritorio Gnome. La versión 25.0.5 incluye mejoras en el rendimiento y la estabilidad, así como actualizaciones de software ; es la última versión estable y se centra en proporcionar una experiencia de usuario fluida y moderna, con un enfoque en la personalización y la facilidad de uso.

Es ideal para usuarios que buscan un sistema operativo fácil de usar y altamente personalizable. Con un rendimiento optimizado y una amplia gama de aplicaciones disponibles, Manjaro Gnome se adapta a las necesidades de diferentes tipos de usuarios, desde principiantes hasta expertos.

Obtención de imagen .vdi

La imagen .vdi puede ser descargada desde el sitio oficial de Manjaro [31], conjunto a múltiples interfaces como lo son Plasma KDE y Xfce. En este caso, se optó por la versión Gnome, que es una de las interfaces más populares y modernas.

Conversión de imagen

La descarga de la imagen .vdi en la página oficial de Manjaro es solo el primer paso. A continuación, se debe convertir la imagen al formato .qcow2, que es más eficiente para su uso en entornos de virtualización.

Cuadro 33. Conversión de imagen .vdi a .qcow2 para Manjaro Gnome 25.0.5

```
# Convertir imagen .vdi a .qcow2
qemu-img convert -c -p -f vdi -O qcow2 Manjaro_gnome25.0.5.vdi
    ↪ Manjaro_gnome25.0.5.qcow2
```

Nota. Elaboración propia.

Registrar imagen en Glance

Una vez que la imagen ha sido convertida al formato .qcow2, se puede registrar en Glance para que esté disponible para su uso en OpenStack especificando el formato de disco y la visibilidad de la imagen.

Cuadro 34. Registro de imagen Manjaro Gnome 25.0.5 en Glance

```
# Registrar imagen en OpenStack
openstack image create \
  --public \
  --container-format bare \
  --disk-format qcow2 \
  --file Manjaro_gnome25.0.5.qcow2 \
  manjaro-gnome-25-0-5
```

Nota. Elaboración propia.

Cuadro 35. Cuadro informativo de Manjaro Gnome 25.0.5 en Glance

```
+------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | 7b42aad0892c45eb7d5c6edde7991bd3                     |
| container_format | bare                                                 |
| created_at       | 2025-08-12T00:47:23Z                                 |
| disk_format      | qcow2                                                |
| file             | /v2/images/19940ee9-c65e-4b45-960b-                  |
|                  | dc5b1ed93168/file                                    |
| id               | 19940ee9-c65e-4b45-960b-dc5b1ed93168                 |
| min_disk         | 0                                                    |
| min_ram          | 0                                                    |
| name             | manjaro-gnome-25-05-5                                |
| owner            | 431b4b0e9d594207a4ef1df4bbe8463a                     |
| properties       | os_hash_algo='sha512', os_hash_value='57121af8e092   |
|                  | 7c778007d8e0e8b85a91f071c2a03910931a209b990f7dfd26   |
|                  | 3345efdbe6dac70cb080ef9edf8bf7bc4f8133455d0a8c3577   |
|                  | 82d93db3d566071c', os_hidden='False',                |
|                  | owner_specified.openstack.md5='',                    |
|                  | owner_specified.openstack.object='images/manjaro-    |
|                  | gnome-25-05-5',                                      |
|                  | owner_specified.openstack.sha256='', stores='file'   |
| protected        | False                                                |
| schema           | /v2/schemas/image                                    |
| size             | 4022043136                                           |
| status           | active                                               |
| tags             |                                                      |
```

```
| updated_at       | 2025-08-12T00:47:48Z                                 |
| virtual_size     | 32423477248                                          |
| visibility       | public                                               |
+------------------+------------------------------------------------------+
```

Nota. Elaboración propia.

Creación de flavor

El *flavor* necesario para Manjaro Gnome es el siguiente no requiere muchos recursos gracias a su diseño ligero, sin embargo, se recomienda asignar al menos 4 GB de RAM y 2 vCPU para un rendimiento óptimo.

Cuadro 36. Creación de flavor para Manjaro Gnome

```
openstack flavor create manjaro-gnome.small --ram 4096 --disk 20 --vcpus 2
```

Nota. Elaboración propia.

Cuadro 37. Cuadro informativo de flavor manjaro-gnome.small

```
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| description                | None                                 |
| disk                       | 20                                   |
| id                         | 8ba19809-bb65-4686-bedf-306580b400a0 |
| name                       | manjaro-gnome.small                  |
| os-flavor-access:is_public | True                                 |
| properties                 |                                      |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       | 0                                    |
| vcpus                      | 2                                    |
+----------------------------+--------------------------------------+
```

Nota. Elaboración propia.

Lanzamiento de instancia

Para lanzar una instancia de Manjaro Gnome, se puede utilizar el siguiente comando, especificando la imagen, el *flavor*, la red y la clave SSH. Se recomienda utilizar la red `demo-net` para facilitar la conexión con una red de prueba.

Cuadro 38. Lanzamiento de instancia Manjaro Gnome 25.0.5 en OpenStack

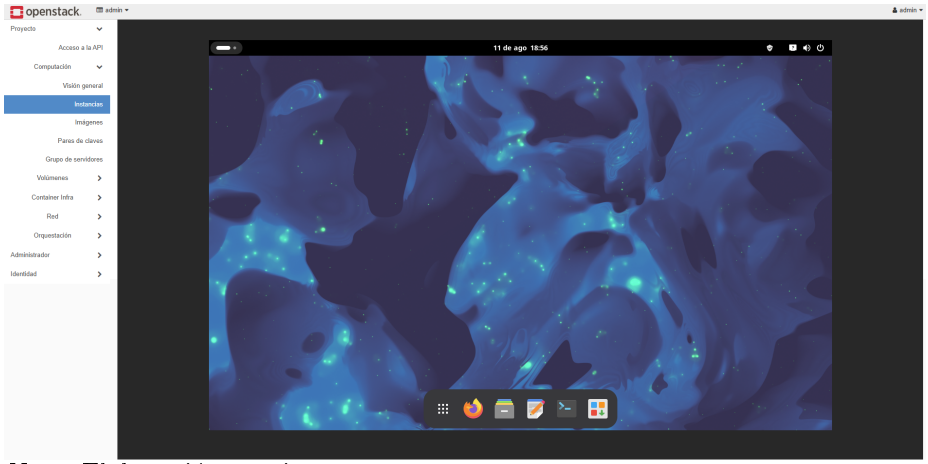
```
# Lanzar instancia de Manjaro Gnome 25.0.5
openstack server create \
  --flavor manjaro-gnome.small \
  --image manjaro-gnome-25.0.5 \
  --network demo-net \
  --key-name mykey \
  --security-group default \
  manjaro-gnome-instance
```

Nota. Elaboración propia.

Las credenciales por defecto para iniciar sesión en la instancia son las siguientes:

- Usuario: Manjaro
- Contraseña: openstack

Figura 9. Máquina virtual Manjaro Gnome 25.0.5.



Nota. Elaboración propia.

7.3. Virtualización de sistemas Windows

7.3.1. Windows Server

Windows Server está diseñado específicamente para entornos de servidor, con una característica especial, ya que incluye interfaz gráfica y herramientas de administración avanzadas para la gestión de redes, seguridad y almacenamiento. Esto lo convierte en una solución versátil y robusta para entornos empresariales o educativos.

En este caso se instalará Windows Server 2022, siendo la última versión disponible de Windows Server y con amplias mejoras en rendimiento, seguridad y administración. Incluyendo el Windows Admin Center para una gesión centralizada.

Seguiremos un proceso similar al de las distribuciones de Linux, pero con algunas diferencias específicas para Windows ya que es un sistema operativo que cuenta con mala compatibilidad con VirtualBox, por lo que su instalación se realizará en Vmware Workstation Pro.

La imagen .iso de Windows Server 2022 puede descargarse desde el sitio oficial de Microsoft [32]. Una vez descargada la imagen, se debe crear una máquina virtual en Vmware Workstation Pro y configurar los recursos necesarios, como la cantidad de RAM, CPU y disco duro. Se recomienda asignar al menos 8 GB de RAM y 60 GB de disco duro para un rendimiento óptimo.

Conversión de imagen

Una vez creada y configurada la máquina virtual y sus credenciales de administrador, se debe de obtener la imagen en formato *.vmdk* desde la sección de *file* y la opción de `Export to OVF`. En la carpeta de la máquina virtual creada en Vmware Workstation Pro se encontrará el archivo *.vmdk*. Al igual que en los casos anteriores, esta imagen debe de ser convertida a un formato compatible con OpenStack, el método de conversión es el mismo, mediante la herramienta *qemu-img*, sin embargo, en este caso se debe de especificar el formato de origen como *.vmdk*.

Cuadro 39. Conversión de imagen .vmdk a .qcow2 para Windows Server 2022

```
# Convertir imagen .vmdk a .qcow2
qemu-img convert -p -f vmdk -O qcow2 \
  "Windows Server 2022.vmdk" windows_server_2022.qcow2
```

Nota. Elaboración propia.

Registro de imagen en Glance

Para registrar la imagen en Glance para su adición a OpenStack, se utilizó el siguiente comando OpenStack image create, especificando el formato del disco y continuando con la nomenclatura establecida.

Cuadro 40. Registro de imagen Windows Server 2022 en Glance

```
# Registrar imagen en OpenStack
openstack image create "Windows Server 2022" \
  --file windows_server_2022.qcow2 \
  --disk-format qcow2 \
  --container-format bare \
  --public
```

Nota. Elaboración propia.

La imagen utilizada no es una *cloud image* genérica, sino una imagen de Windows Server 2022 instalada en VMware bajo condiciones específicas del *hardware* virtual asignado, tales como sus discos, controladores y configuración de red. Al cargar una imagen tan compleja como esta en OpenStack, las diferencias de entorno pueden causar problemas de compatibilidad para el *booting* de la instancia.

Para evitar estas complicaciones, se debe indicar al hipervisor cómo presentar el *hardware* físico de forma que coincida con OpenStack luego de ser migrado, de esta manera se asegura que la instancia de Windows Server 2022 se pueda lanzar correctamente. Para esto, se utiliza el siguiente comando.

Cuadro 41. Configuración de propiedades de imagen Windows Server 2022 en Glance

```
# Modificar la imagen subida en Glance
openstack image set "Windows Server 2022" \
  --property hw_firmware_type=uefi \
  --property os_type=windows \
  --property os_distro=windows \
  --property hw_disk_bus=sata \
  --property hw_machine_type=q35 \
  --property os_secure_boot=disabled
```

Nota. Elaboración propia.

La propiedad `uefi` indica que la instancia debe de arrancar con un *firmware* UEFI, que es el método más común para Windows Server. Las propiedades `os_type` y `os_distro` indican que la imagen es de Windows, mientras que `hw_disk_bus` y `hw_machine_type` especifican el tipo de bus de disco y la máquina virtual utilizada. Por último, `os_secure_boot` se establece en "deshabilitado"para evitar problemas de compatibilidad con el arranque seguro.

Creación de flavor

Para crear un *flavor* adecuado para Windows Server 2022, se utilizará el comando **openstack flavor create**, especificando los recursos necesarios. En este caso, se recomienda un mínimo de 8 GB de RAM, 60 GB de disco y 2 vCPU. Se utiliza el parámetro `-id auto` para que OpenStack genere un ID automáticamente.

Cuadro 42. Creación de flavor para Windows Server 2022 en OpenStack

```
# Crear flavor para Windows Server 2022
openstack flavor create windows-server.small --id auto --ram 8192 --disk 60 --vcpus 2
```

Nota. Elaboración propia.

Para asegurar la correcta compatibilidad, se asignan los mismos parámetros asignados a la imagen de Windows Server 2022. Esto incluye el tipo de *firmware*, el tipo de máquina y el bus de disco.

Estos parámetros son cruciales para que OpenStack pueda manejar correctamente la instancia de Windows Server 2022.

Cuadro 43. Configuración de propiedades de flavor para Windows Server 2022

```
openstack flavor set windows-server.small \
  --property hw_firmware_type=uefi \
  --property hw_machine_type=q35 \
  --property hw_disk_bus=sata
```

Nota. Elaboración propia.

Lanzamiento de la instancia

Por último, se podrá lanzar la instancia de Windows Server 2022 con el comando **openstack server create**.

Cuadro 44. Lanzamiento de instancia Windows Server 2022 en OpenStack

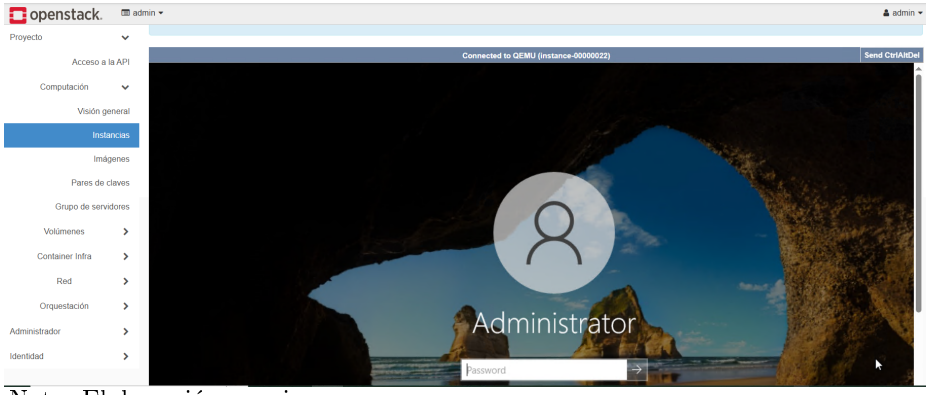
```
# Lanzar instancia de Windows Server 2022
openstack server create \
  --flavor windows-server.small \
  --image "Windows Server 2022" \
  --network demo-net \
  --key-name mykey \
  --security-group default \
  windows-server-2022-instance
```

Nota. Elaboración propia.

y posteriormente visualizarlo en la interfaz gráfica de Horizon. Las credenciales de administrador se habrán configurado previamente en la máquina virtual creada en Vmware Workstation Pro y son las siguientes:

- Usuario: `Administrator`
- Contraseña: `openstack`

Figura 10. Máquina virtual Windows Server 2022.



Nota. Elaboración propia.

7.3.2. Windows 11 pro

Windows 11 Pro es la versión profesional del sistema operativo Windows 11, diseñado para usuarios avanzados y entornos empresariales. Ofrece características adicionales en comparación con la versión estándar de Windows 11, como soporte para dominios, cifrado de datos y herramientas de administración avanzadas. Estas características hacen que Windows 11 Pro sea ideal para profesionales y empresas que requieren un mayor control sobre su entorno informático.

La imagen .iso de Windows 11 Pro puede descargarse desde el sitio oficial de Microsoft [33]. Al igual que con Windows Server 2022, se debe crear una máquina virtual en Vmware Workstation Pro y configurar los recursos necesarios, como la cantidad de RAM, CPU y disco duro. Se recomienda asignar al menos 8 GB de RAM y 60 GB de disco duro para un rendimiento óptimo.

Conversión de imagen

Una vez creada y configurada la máquina virtual y sus credenciales de administrador, se debe de obtener la imagen en formato *.vmdk* desde la sección de *file* y la opción de `Export to OVF`. En la carpeta de la máquina virtual creada en Vmware Workstation Pro se encontrará el archivo *.vmdk*.

Al igual que en los casos anteriores, esta imagen debe de ser convertida a un formato compatible con OpenStack, el método de conversión es el mismo, mediante la herramienta *qemu-img*, sin embargo, en este caso se debe de especificar el formato de origen como *.vmdk*.

Cuadro 45. Conversión de imagen .vmdk a .qcow2 para Windows 11 Pro

```
# Convertir imagen .vmdk a .qcow2
qemu-img convert -p -f vmdk -O qcow2 \
  "Windows 11 x64.vmdk" windows_11_pro.qcow2
```

Nota. Elaboración propia.

Registro de imagen en Glance

Para registrar la imagen en Glance para su adición a OpenStack, se utilizará el siguiente comando OpenStack image create, especificando el formato del disco y continuando con la nomenclatura establecida.

Cuadro 46. Registro de imagen en Glance

```
# Registrar imagen en Glance
openstack image create \
  --file windows_11_pro.qcow2 \
  --disk-format qcow2 \
  --container-format bare \
  --public \
  "Windows 11 Pro"
```

Nota. Elaboración propia.

Cuadro 47. Detalles de imagen Windows 11 Pro en Glance

```
  +------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | db52facd78567a5c31c8f18467d7bcd5                     |
| container_format | bare                                                 |
| created_at       | 2025-10-17T01:40:58Z                                 |
| disk_format      | qcow2                                                |
| file             | /v2/images/56f09c50-9fb5-4b47-b20f-                  |
|                  | 2ace8f3668ea/file                                    |
| id               | 56f09c50-9fb5-4b47-b20f-2ace8f3668ea                 |
| min_disk         | 0                                                    |
| min_ram          | 0                                                    |
| name             | Windows 11 Pro                                       |
| owner            | 5260c7767b584c58a1da728373b2d8f3                     |
| properties       | os_hash_algo='sha512', os_hash_value='7e6c070ba4e    |
|                  | 897d0dd9d98a13d6f17f49b014392ec6159b8c0825bc573c2    |
|                  | 18a2e2a0cac84916786a68771f46029f06f8f93c868c8ffa8    |
|                  | 9c7b38a33e11c78e00c', os_hidden='False',             |
|                  | owner_specified.openstack.md5='',                    |
|                  | owner_specified.openstack.object='images/Windows     |
|                  | 11 Pro', owner_specified.openstack.sha256='',        |
|                  | stores='file'                                        |
| protected        | False                                                |
| schema           | /v2/schemas/image                                    |
| size             | 18812436480                                          |
| status           | active                                               |
| tags             |                                                      |
```

```
| updated_at       | 2025-10-17T01:42:58Z                                 |
| virtual_size     | 68719476736                                          |
| visibility       | public                                               |
+------------------+------------------------------------------------------+
```

Nota. Elaboración propia.

Al igual que con Windows Server 2022, la imagen utilizada no es una *cloud image* genérica, sino que una imagen de Windows 11 Pro previamente instalada en VMware bajo condiciones específicas del *hardware* virtual asignado, tales como sus discos, controladores y configuración de red. Al cargar una imagen tan compleja como esta en OpenStack, las diferencias de entorno pueden causar problemas de compatibilidad para el *booting* de la instancia.

Para evitar estas complicaciones, se debe indicar al hipervisor cómo presentar el *hardware* físico de forma que coincida con OpenStack luego de ser migrado. Para esto se utiliza el siguiente comando.

Cuadro 48. Configuración de propiedades de imagen Windows 11 Pro en Glance

```
# Modificar la imagen subida en Glance
openstack image set "Windows 11 Pro" \
  --property hw_firmware_type=uefi \
  --property os_type=windows \
  --property os_distro=windows \
  --property hw_disk_bus=sata \
  --property hw_machine_type=q35 \
  --property os_secure_boot=disabled
```

Nota. Elaboración propia.

Creación de flavor

Para crear un *flavor* adecuado para Windows 11 Pro, se utilizará el comando **openstack flavor create**, especificando los recursos necesarios. En este caso, se recomienda un mínimo de 8 GB de RAM, 60 GB de disco y 2 vCPU.

Cuadro 49. Creación de flavor para Windows 11 Pro en OpenStack

```
# Crear flavor para Windows 11 Pro
openstack flavor create windows-11.small --ram 8192 --disk 60 --vcpus 2
```

Nota. Elaboración propia.

Cuadro 50. Detalles de flavor Windows 11 Pro en OpenStack

```
  +----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
```

```
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| description                | None                                 |
| disk                       | 60                                   |
| id                         | eb16a73d-de23-4ee7-800b-dee5823808fe |
| name                       | windows-11.small                     |
| os-flavor-access:is_public | True                                 |
| properties                 |                                      |
| ram                        | 8192                                 |
| rxtx_factor                | 1.0                                  |
| swap                       | 0                                    |
| vcpus                      | 2                                    |
+----------------------------+--------------------------------------+
```

Nota. Elaboración propia.

Para asegurar la correcta compatibilidad, se asignan los mismos parámetros asignados a la imagen de Windows 11 Pro. Esto incluye el tipo de *firmware*, el tipo de máquina y el bus de disco. Estos parámetros son cruciales para que OpenStack pueda manejar correctamente la instancia de Windows 11 Pro.

Cuadro 51. Configuración de propiedades de flavor para Windows 11 Pro

```
openstack flavor set windows-11.small \
  --property hw_firmware_type=uefi \
  --property hw_machine_type=q35 \
  --property hw_disk_bus=sata
```

Nota. Elaboración propia.

Lanzamiento de la instancia

Por último, se podrá lanzar la instancia de Windows 11 Pro con el comando **openstack server create** y posteriormente visualizarlo en la interfaz gráfica de Horizon.

Cuadro 52. Lanzamiento de instancia Windows 11 Pro en OpenStack

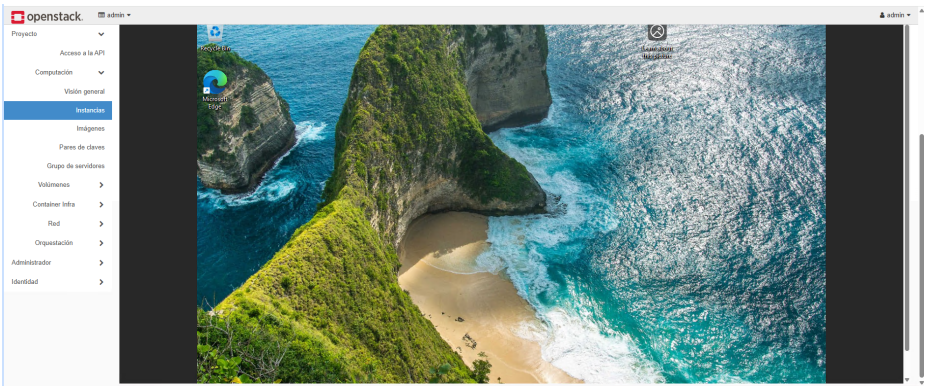
```
# Lanzar instancia de Windows 11 Pro
openstack server create \
  --flavor windows-11.small \
  --image "Windows 11 Pro" \
  --network demo-net \
  --key-name mykey \
  --security-group default \
  windows-11-pro-instance
```

Nota. Elaboración propia.

El usuario y contraseña establecidos durante la creación de la máquina virtual en Vmware Workstation Pro serán necesarios para iniciar sesión en la instancia de Windows 11 Pro una vez que esté en funcionamiento. Las credenciales son las siguientes:

- Usuario: **Windows**
- Contraseña: **openstack**

Figura 11. Máquina virtual Windows 11 Pro.



Nota. Elaboración propia.

CAPÍTULO 8

Implementación de elasticidad horizontal

Para el desarrollo de la elasticidad horizontal en OpenStack, se asume que se cuenta con una instalación funcional de OpenStack utilizando Kolla-Ansible, siguiendo los pasos descritos en el trabajo [28]. Además, se asume que se tiene acceso administrativo al entorno de OpenStack para realizar las configuraciones necesarias así como perfil *root* en el servidor donde se encuentra instalado OpenStack y al entorno virtualizado del despliegue mencionado.

Cuadro 53. Activación de usuario root, entorno virtual y archivo de credenciales de OpenStack para elasticidad horizontal

```
# activar usuario root
sudo su
password: openstack
# activar entorno virtual
source ~/openstack/os-venv/bin/activate
# cargar archivo de credenciales
source etc/kolla/admin-openrc.sh
```

Nota. Elaboración propia.

8.1. Instalación de servicios por medio de contenedores

Para la implementación de elasticidad horizontal es necesario instalar servicios nuevos de monitoreo, orquestación y manejo de alarmas. En este caso se implementó mediante los servicios de Heat (orquestación), Ceilometer (monitoreo), Aodh (alarmas) y Gnocchi (almacenamiento de métricas).

Estos servicios permiten monitorear el estado de las instancias, gestionar alarmas y realizar acciones automáticas en función de las métricas recolectadas.

Antes de iniciar con la instalación de estos servicios, se recomienda hacer un respaldo del archivo de configuración `globals.yml`, ubicado en el directorio de configuración de OpenStack. Esto es importante para poder restaurar la configuración en caso de que algo salga mal durante la instalación.

Cuadro 54. Respaldo de archivo de configuración

```
cp /etc/kolla/globals.yml globals.yml.$( date +%Y%m%d%H%M%S )
```

Nota. Elaboración propia.

8.1.1. Instalación de Aodh

Primero se debe de ingresar al directorio de configuración de OpenStack y editar el archivo `globals.yml` para habilitar el servicio de Aodh. En este archivo, se debe buscar la sección correspondiente a Aodh y asegurarse de que esté configurada correctamente.

Cuadro 55. Configuración de Aodh en globals.yml

```
# Ingresar al archivo globals.yml
nano /etc/kolla/globals.yml

# Habilitar Aodh (modificar y descomentar la siguiente linea)
aodh_enabled: "yes"
```

Nota. Elaboración propia.

Luego de editar el archivo, se debe de guardar y salir del editor. A continuación, se debe de ejecutar el siguiente comando para aplicar los cambios y reiniciar los servicios de OpenStack, esto puede tardar varios minutos dependiendo del *hardware* utilizado y los servicios previamente instalados.

Cuadro 56. Despliegue de servicios en OpenStack con Kolla-Ansible

```
kolla-ansible deploy -i all-in-one
```

Nota. Elaboración propia.

El comando **kolla-ansible deploy** se encarga de aplicar los cambios realizados en el archivo `globals.yml` y reiniciar los servicios de OpenStack para que los cambios surtan efecto. Este comando es especialmente útil cuando se agregan nuevos servicios o se realizan cambios significativos en la configuración de OpenStack. Si se realizan cambios menores, se puede utilizar el comando **kolla-ansible reconfigure** en su lugar, que es más rápido ya que solo reinicia los servicios afectados por los cambios.

8.1.2. Instalación de Ceilometer y Gnocchi

Estos servicios se habilitan en conjunto ya que Ceilometer se encarga de recolectar métricas y Gnocchi almacena estas métricas para su posterior análisis. Para instalar estos servicios, se debe seguir un proceso similar al de Aodh. Primero, se debe editar el archivo `globals.yml` para habilitar Ceilometer y Gnocchi. En este caso, se deben buscar las secciones correspondientes a estos servicios y asegurarse de que estén configuradas correctamente.

Cuadro 57. Configuración de Ceilometer y Gnocchi en globals.yml

```
# Habilitar Ceilometer y Gnocchi
ceilometer_enabled: "yes"
gnocchi_enabled: "yes"
```

Nota. Elaboración propia.

Luego de editar el archivo, se debe de guardar y salir del editor. A continuación, se debe de ejecutar el siguiente comando para aplicar los cambios y reiniciar los servicios de OpenStack:

Cuadro 58. Reconfiguración de servicios en OpenStack con Kolla-Ansible

```
kolla-ansible reconfigure -i all-in-one
```

Nota. Elaboración propia.

8.1.3. Verificación de servicios instalados

Para verificar que los servicios se han instalado correctamente, se pueden utilizar los siguientes comandos:

Primero se puede verificar que los contenedores de docker correspondientes a Aodh, Gnocchi y Ceilometer estén en ejecución, estos contenedores son los que manejan los servicios instalados, es esencial que todos estén en estado `UP` y funcionando correctamente.

Cuadro 59. Verificación de servicios en contenedores

```
docker ps --format '{{.Names}}' | grep -E 'aodh|gnocchi|ceilometer|heat'
```

Nota. Elaboración propia.

Cuadro 60. Respuesta esperada de verificacion de servicios en contenedores

```
gnocchi_metricd
gnocchi_api
ceilometer_compute
ceilometer_central
ceilometer_notification
aodh_notifier
aodh_listener
```

```
aodh_evaluator
aodh_api
heat_engine
heat_api_cfn
heat_api
```

Nota. Elaboración propia.

También se puede verificar que los endpoints de los servicios estén registrados correctamente en OpenStack. Los endpoints son las direcciones URL que los clientes utilizan para acceder a los servicios de OpenStack. Es importante que los endpoints de Aodh, Gnocchi y Heat estén presentes y correctamente configurados, de lo contrario, los servicios no podrán ser utilizados por los clientes.

Cuadro 61. Verificación de *endpoints* de servicios instalados

```
openstack endpoint list | grep -E 'aodh|gnocchi|heat'
```

Nota. Elaboración propia.

Cuadro 62. Respuesta esperada de verificación de *endpoints* de servicios instalados

```
| 42723c5ea7364723bb1c23797221f07f | RegionOne | aodh         | alarming        |
    ↪ True    | internal  | http://192.168.74.69:8042                 |
| 46230178010f467bab310e689104d046 | RegionOne | heat-cfn     | cloudformation  |
    ↪ True    | public    | http://192.168.74.69:8000/v1              |
| b7858faefcba400f9332fc53c3d82382 | RegionOne | aodh         | alarming        |
    ↪ True    | public    | http://192.168.74.69:8042                 |
| bd7e4fda16fa427f8950c1e5036ee693 | RegionOne | heat         | orchestration   |
    ↪ True    | public    | http://192.168.74.69:8004/v1/%(tenant_id)s |
| be359f1e84c548ff84c194725bdb91b7 | RegionOne | gnocchi      | metric          |
    ↪ True    | public    | http://192.168.74.69:8041                 |
| e9ec4379e4a34a86bf7e0ba97a8ee880 | RegionOne | heat         | orchestration   |
    ↪ True    | internal  | http://192.168.74.69:8004/v1/%(tenant_id)s |
| f8306273ae7b42f0a257f05af67ac8e1 | RegionOne | gnocchi      | metric          |
    ↪ True    | internal  | http://192.168.74.69:8041                 |
| fb18e3a806624e30a127f7462e10dd99 | RegionOne | heat-cfn     | cloudformation  |
    ↪ True    | internal  | http://192.168.74.69:8000/v1              |
```

Nota. Elaboración propia.

Los siguientes comandos permiten visualizar la lista de alarmas, métricas y recursos almacenados en Aodh, Ceilometer y Gnocchi respectivamente. Si los servicios están funcionando correctamente, se debería ver una lista de alarmas, métricas y recursos disponibles si han sido creados previamente, de ser una instalación nueva, es posible que estas listas estén vacías.

Cuadro 63. Verificación de servicios

```
# Verificar Heat
openstack stack list

# Verificar estado de Aodh
openstack alarm list

# Verificar estado de Ceilometer
openstack meter list

# Verificar estado de Gnocchi
gnocchi resource list
```

Nota. Elaboración propia.

8.2. Introducción a Heat

Heat es el servicio de orquestación de OpenStack, que permite a los usuarios definir y proporcionar infraestructura en la nube utilizando plantillas. Estas plantillas son archivos en formato YAML que describen los recursos que se desean crear, así como sus relaciones y dependencias. Heat utiliza el concepto de *stacks* para gestionar la creación, actualización y eliminación de recursos de manera coherente y controlada.

Un archivo YAML de Heat generalmente incluye secciones para definir los recursos, parámetros, salidas y condiciones. Los recursos pueden incluir instancias de computación, redes, volúmenes de almacenamiento y otros servicios de OpenStack. Los parámetros permiten a los usuarios personalizar la plantilla al momento de su despliegue, mientras que las salidas proporcionan información útil sobre los recursos creados.

La creación de estas plantillas permite automatizar el despliegue de recursos en OpenStack, facilitando la gestión y escalabilidad de las aplicaciones. Además, Heat permite la integración con otros servicios de OpenStack, como Neutron para redes y Cinder para almacenamiento, lo que proporciona una solución completa para la orquestación de infraestructuras en la nube.

8.2.1. Creación de una plantilla básica

Para crear una plantilla básica en Heat, se debe definir un archivo YAML que describa los recursos que se desean crear. Para esto es necesario tener bien definidos los recursos que se van a utilizar, como imágenes, *flavors* y redes. Los recursos disponibles en Openstack se pueden consultar mediante los siguientes comandos:

Cuadro 64. Listar recursos disponibles

```
# Listar imagenes disponibles
openstack image list

# Listar flavors disponibles
openstack flavor list

# Listar redes disponibles
openstack network list
```

Nota. Elaboración propia.

Luego de conocer los recursos disponibles, se debe crear una carpeta en el directorio personal del usuario para almacenar las plantillas de Heat. Por ejemplo, se puede crear una carpeta llamada `heat-templates` en el directorio personal del usuario. Esto ayudará a mantener organizadas las plantillas y facilitará su gestión.

Cuadro 65. Creación de carpeta y archivo de plantilla de Heat

```
#crear carpeta
mkdir -p ~/openstack/heat_templates

#acceder a carpeta
cd ~/openstack/heat_templates

#crear archivo de plantilla
nano hello_heat.yaml
```

Nota. Elaboración propia.

Una vez creado el archivo, se procede a realizar una plantilla básica mediante un archivo YAML con la siguiente estructura:

Cuadro 66. Plantilla básica de Heat para lanzar una instancia

```
heat_template_version: 2016-10-14

description: Plantilla simple para lanzar una instancia

resources:
  test_instance:
    type: OS::Nova::Server
    properties:
      name: test-vm
      image: cirros
      flavor: m1.micro
      networks:
```

```
        - network: public
```

Nota. Elaboración propia.

Esta plantilla define una instancia llamada `test-vm` que utiliza la imagen `cirros` y el *flavor* `m1.micro`, conectada a la red `public`. Este archivo es un ejemplo básico que puede ser modificado y ampliado según las necesidades del usuario. En este caso, se cuenta con dos secciones principales: `description` y `resources`. La sección `description` proporciona una breve descripción de la plantilla mientras que la sección `resources` define los recursos que se van a crear, en este caso, una instancia de servidor con las propiedades definidas. Este archivo debe guardarse con la extensión `.yaml` para que pueda ser reconocido como una plantilla de Heat, para lanzar y gestionar esta plantilla, se utilizarán los comandos de Heat.

Cuadro 67. Lanzamiento de plantilla básica de Heat

```
# Lanzar plantilla
openstack stack create -t hello_heat.yaml hello_stack
```

Nota. Elaboración propia.

Luego de lanzar la plantilla, aparecerá un cuadro de estado del stack, donde se podrá observar el progreso de la creación de los recursos definidos en la plantilla. Este cuadro mostrará información como el nombre del stack, su estado actual (por ejemplo, `CREATE_IN_PROGRESS`), la fecha y hora de creación, y otros detalles relevantes. Es importante monitorear este estado para asegurarse de que todos los recursos se creen correctamente y para identificar cualquier problema que pueda surgir durante el proceso.

Cuadro 68. Respuesta esperada al lanzar plantilla de Heat

```
+---------------------+----------------------------------------------+
| Field               | Value                                        |
+---------------------+----------------------------------------------+
| id                  | 6602a88e-a2d0-43b8-9d13-eb5ff665ec1a         |
| stack_name          | hello_stack                                  |
| description         | Plantilla simple para lanzar una instancia   |
| creation_time       | 2025-10-25T04:02:46Z                         |
| updated_time        | None                                         |
| stack_status        | CREATE_IN_PROGRESS                           |
| stack_status_reason | Stack CREATE started                         |
+---------------------+----------------------------------------------+
```

Nota. Elaboración propia.

Para verificar la instancia que se ha creado, se puede utilizar el comando **openstack server list**, que mostrará una lista de todas las instancias activas en el entorno de OpenStack. En esta lista, se podrá ver la instancia `test-vm` que fue creada por la plantilla de Heat, junto con su ID, estado, red y otra información relevante.

Cuadro 69. Verificación de instancia creada por plantilla de Heat

```
# Verificar instancia creada
openstack server list
```

Nota. Elaboración propia.

Cuadro 70. Respuesta esperada al verificar instancia creada por plantilla de Heat

```
+-----------+-----------+--------+-----------+-----------+---------------+
| ID        | Name      | Status | Networks  | Image     | Flavor        |
+-----------+-----------+--------+-----------+-----------+---------------+
| 3b81794f- | test-vm   | ACTIVE | public=19 | cirros    | m1.micro      |
| 49b9-     |           |        | 2.168.74. |           |               |
| 4698-     |           |        | 126       |           |               |
| 9c53-     |           |        |           |           |               |
| 53ccc7999 |           |        |           |           |               |
| c76       |           |        |           |           |               |
| bc8338a7- | vm_privad | ACTIVE | red_priva | cirros    | m1.micro      |
| 0718-     | a         |        | da=192.16 |           |               |
| 44bc-     |           |        | 8.100.34  |           |               |
| 86ad-fced |           |        |           |           |               |
| 99add3b4  |           |        |           |           |               |
| 7debaccd- | test-si2y | ACTIVE | test=10.0 | magnum-   | flavor-       |
| b978-     | ckdr4zm6- |        | .0.32, 19 | fedora-   | kubernetes    |
| 4855-     | node-0    |        | 2.168.74. | coreos-40 |               |
| 9435-     |           |        | 129       |           |               |
| fc8704977 |           |        |           |           |               |
| 654       |           |        |           |           |               |
| 2ed767d2- | test-si2y | ACTIVE | test=10.0 | magnum-   | flavor-       |
| e578-     | ckdr4zm6- |        | .0.225, 1 | fedora-   | kubernetes    |
| 451b-     | master-0  |        | 92.168.74 | coreos-40 |               |
| b914-     |           |        | .128      |           |               |
| b73398017 |           |        |           |           |               |
| a1a       |           |        |           |           |               |
+-----------+-----------+--------+-----------+-----------+---------------+
```

Nota. Elaboración propia.

Para eliminar la plantilla y todos los recursos creados por ella, se puede utilizar el comando **openstack stack delete**, especificando el nombre del stack que se desea eliminar.

Cuadro 71. Eliminación de stack de Heat

```
# Eliminar stack
openstack stack delete hello_stack
```

Nota. Elaboración propia.

8.2.2. Creación de una plantilla con red privada

Las plantillas de Heat pueden ser más complejas e incluir redes privadas, volúmenes de almacenamiento y otros recursos. A continuación, se muestra un ejemplo de una plantilla que crea una instancia en una red privada:

Cuadro 72. Plantilla de Heat para crear una red privada y lanzar una instancia en ella

```
heat_template_version: 2016-10-14

description: >
  Crea red privada, subred, router, y lanza una instancia en esa red.

parameters:
  public_network:
    type: string
    default: public
    description: Nombre de la red publica a la que se conecta el router

resources:
  red_privada:
    type: OS::Neutron::Net
    properties:
      name: red_privada

  subred_privada:
    type: OS::Neutron::Subnet
    properties:
      name: subred_privada
      network_id: { get_resource: red_privada }
      cidr: 192.168.100.0/24
      gateway_ip: 192.168.100.1
      ip_version: 4
      enable_dhcp: true

  router_privado:
    type: OS::Neutron::Router
    properties:
      name: router_privado
      external_gateway_info:
        network: { get_param: public_network }

  router_interface:
    type: OS::Neutron::RouterInterface
    properties:
      router_id: { get_resource: router_privado }
```

```
      subnet_id: { get_resource: subred_privada }

  vm_privada:
    type: OS::Nova::Server
    properties:
      name: vm_privada
      image: cirros
      flavor: m1.micro
      networks:
        - network: { get_resource: red_privada }

outputs:
  private_network:
    description: ID de la red privada creada
    value: { get_resource: red_privada }

  instance_ip:
    description: IP de la instancia privada
    value: { get_attr: [vm_privada, first_address] }
```

Nota. Elaboración propia.

Esta plantilla permite automatizar la creación de una infraestructura básica en OpenStack, facilitando el despliegue de instancias en redes privadas y su conexión a redes públicas mediante un router. El uso de parámetros en la plantilla hace posible reutilizarla en diferentes entornos, adaptando el nombre de la red pública según sea necesario. Esta plantilla es un caso de uso mucho más avanzado que el ejemplo básico anterior, y puede servir como base para crear infraestructuras más complejas en OpenStack. La plantilla cuenta con varias secciones importantes:

- `parameters`: define los parámetros que pueden ser personalizados al momento de lanzar la plantilla. En este caso, se define el nombre de la red pública a la que se conectará el router. Haciendo un diseño modular y reutilizable.
- `resources`: define los recursos que se van a crear, incluyendo la red privada, subred, router, interfaz de router e instancia de servidor. Cada recurso tiene sus propias propiedades que especifican cómo debe ser configurado.
- `outputs`: proporciona información útil sobre los recursos creados, como el ID de la red privada y la dirección IP de la instancia para su posterior uso y manejo.

Al lanzar esta plantilla, se crearán automáticamente los siguientes recursos:

- Una red privada llamada `red_privada`.
- Una subred privada llamada `subred_privada` con un rango de direcciones IP definido por 192.168.100.0/24.

- Un router llamado `router_privado` que conecta la red privada a la red pública.
- Una interfaz de router que conecta el router a la subred privada.
- Una instancia llamada `vm_privada` que se conecta a la red privada.

Al momento de eliminar la plantilla, todos los recursos creados se eliminarán automáticamente, incluyendo la red privada, la subred, el router, la interfaz de router y la instancia. Haciendo que la gestión de recursos sea más sencilla y eficiente y dejando el entorno limpio al momento de eliminar la plantilla. Para lanzar y gestionar esta plantilla, se utilizarán los mismos comandos de Heat mencionados anteriormente, adaptando el nombre del archivo y del stack según sea necesario.

Cuadro 73. Lanzamiento de plantilla con red privada

```
openstack stack create -t private_network.yaml red_privada_stack
```

Nota. Elaboración propia.

Luego de lanzar la plantilla, aparecerá un cuadro de estado del stack, donde se podrá observar el progreso de la creación de los recursos definidos en la plantilla. Este cuadro mostrará información como el nombre del stack, su estado actual (por ejemplo, `CREATE_IN_PROGRESS`), la fecha y hora de creación, y otros detalles relevantes. Es importante monitorear este estado para asegurarse de que todos los recursos se creen correctamente y para identificar cualquier problema que pueda surgir durante el proceso.

Cuadro 74. Estado del stack tras lanzar plantilla con red privada

```
+---------------------+------------------------------------------------+
| Field               | Value                                          |
+---------------------+------------------------------------------------+
| id                  | ae31f439-e62c-4dab-a285-1b6c002d2b47           |
| stack_name          | red_privada_stack                              |
| description         | Crea red privada, subred, router, y lanza una  |
|                     | instancia en esa red.                          |
|                     |                                                |
| creation_time       | 2025-10-25T03:35:47Z                           |
| updated_time        | None                                           |
| stack_status        | CREATE_IN_PROGRESS                             |
| stack_status_reason | Stack CREATE started                           |
+---------------------+------------------------------------------------+
```

Nota. Elaboración propia.

La instancia creada en dicha red privada puede ser verificada en la interfaz gráfica de Horizon, donde se podrá observar la instancia `vm_privada` en estado activo y conectada a la red privada `red_privada`. Desde Horizon, se pueden gestionar las instancias, redes y otros recursos creados por la plantilla de Heat, proporcionando una interfaz visual para administrar la infraestructura desplegada en OpenStack. También se puede verificar la creación de los recursos utilizando los

comandos de OpenStack para listar redes, subredes, routers e instancias, asegurándose de que todos los componentes definidos en la plantilla se hayan creado correctamente.

Cuadro 75. Verificación de recursos creados por la plantilla con red privada

```
openstack network list
openstack subnet list
openstack router list
openstack server list
```

Nota. Elaboración propia.

Con los comandos anteriores, se podrá verificar que la red privada, subred, router e instancia se hayan creado correctamente y estén en el estado esperado. Esto es crucial para asegurar que la plantilla de Heat haya sido ejecutada con éxito y que los recursos estén disponibles para su uso.

Cuadro 76. Respuesta esperada de verificación de recursos

```
+--------------------------+---------------+--------------------------+
| ID                       | Name          | Subnets                  |
+--------------------------+---------------+--------------------------+
| 0735dc31-6380-4608-bb9e- | public        | d927d5fd-7a90-4d45-90bb- |
| 296f45e19201             |               | 83d11a0dd9ca             |
| 7f97650e-b327-4f7a-84f0- | autoscale-net | 9a028181-7cad-49dd-      |
| 6b32e8b61bb9             |               | bfc4-9d99bf9f20f1        |
| bf751b7d-840e-4d3e-ae1b- | red_privada   | 9467714c-3549-4137-ae1e- |
| c3de35621e48             |               | ad4875744691             |
| dc4174a5-793c-4650-9950- | test          | 1f2ae1ba-ddf3-446c-b043- |
| 3c7d0099d877             |               | 2772d59dee3a             |
+--------------------------+---------------+--------------------------+
+-----------------+-----------------+-----------------+------------------+
| ID              | Name            | Network         | Subnet           |
+-----------------+-----------------+-----------------+------------------+
| 1f2ae1ba-ddf3-  | test-si2yckdr4z | dc4174a5-793c-  | 10.0.0.0/24      |
| 446c-b043-      | m6-network-     | 4650-9950-      |                  |
| 2772d59dee3a    | vwungw47choz-   | 3c7d0099d877    |                  |
|                 | private_subnet- |                 |                  |
|                 | mi7o4s6ikiia    |                 |                  |
| 9467714c-3549-  | subred_privada  | bf751b7d-840e-  | 192.168.100.0/24 |
| 4137-ae1e-      |                 | 4d3e-ae1b-      |                  |
| ad4875744691    |                 | c3de35621e48    |                  |
| 9a028181-7cad-  | autoscale-      | 7f97650e-b327-  | 192.168.100.0/24 |
| 49dd-bfc4-      | subnet          | 4f7a-84f0-      |                  |
| 9d99bf9f20f1    |                 | 6b32e8b61bb9    |                  |
| d927d5fd-7a90-  | public          | 0735dc31-6380-  | 192.168.72.0/22  |
| 4d45-90bb-      |                 | 4608-bb9e-      |                  |
| 83d11a0dd9ca    |                 | 296f45e19201    |                  |
```

```
+-----------------+-----------------+-----------------+------------------+
+-----------+-----------+--------+-------+-----------+-------------+-------+
| ID        | Name      | Status | State | Project   | Distributed | HA    |
+-----------+-----------+--------+-------+-----------+-------------+-------+
| 2235c268- | router_pr | ACTIVE | UP    | 5260c7767 | False       | False |
| fdcd-     | ivado     |        |       | b584c58a1 |             |       |
| 4be5-     |           |        |       | da728373b |             |       |
| 82d4-     |           |        |       | 2d8f3     |             |       |
| a9f86bcb6 |           |        |       |           |             |       |
| 30f       |           |        |       |           |             |       |
| f7ce3abd- | test-si2y | ACTIVE | UP    | 5260c7767 | False       | False |
| 9f28-     | ckdr4zm6- |        |       | b584c58a1 |             |       |
| 46aa-     | network-v |        |       | da728373b |             |       |
| 93d6-     | wungw47ch |        |       | 2d8f3     |             |       |
| 9463bfb3a | oz-extrou |        |       |           |             |       |
| 3dc       | ter-lsg2c |        |       |           |             |       |
|           | ny5d75j   |        |       |           |             |       |
+-----------+-----------+--------+-------+-----------+-------------+-------+
+-----------+-----------+--------+-----------+-----------+---------------+
| ID        | Name      | Status | Networks  | Image     | Flavor        |
+-----------+-----------+--------+-----------+-----------+---------------+
| bc8338a7- | vm_privad | ACTIVE | red_priva | cirros    | m1.micro      |
| 0718-     | a         |        | da=192.16 |           |               |
| 44bc-     |           |        | 8.100.34  |           |               |
| 86ad-fced |           |        |           |           |               |
| 99add3b4  |           |        |           |           |               |
| 7debaccd- | test-si2y | ACTIVE | test=10.0 | magnum-   | flavor-       |
| b978-     | ckdr4zm6- |        | .0.32, 19 | fedora-   | kubernetes    |
| 4855-     | node-0    |        | 2.168.74. | coreos-40 |               |
| 9435-     |           |        | 129       |           |               |
| fc8704977 |           |        |           |           |               |
| 654       |           |        |           |           |               |
| 2ed767d2- | test-si2y | ACTIVE | test=10.0 | magnum-   | flavor-       |
| e578-     | ckdr4zm6- |        | .0.225, 1 | fedora-   | kubernetes    |
| 451b-     | master-0  |        | 92.168.74 | coreos-40 |               |
| b914-     |           |        | .128      |           |               |
| b73398017 |           |        |           |           |               |
| a1a       |           |        |           |           |               |
+-----------+-----------+--------+-----------+-----------+---------------+
```

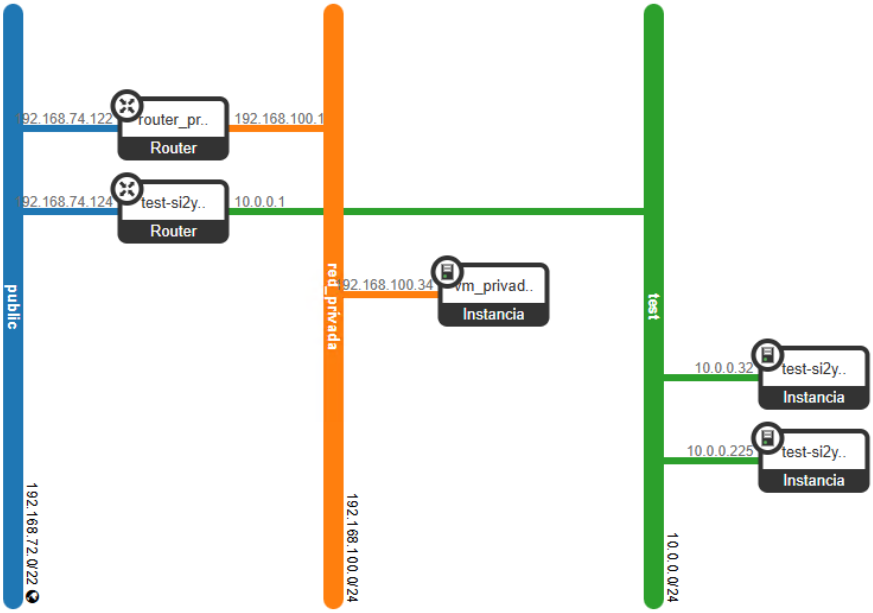
Nota. Elaboración propia.

Los recursos creados se pueden visualizar en los cuadros de detalles de redes, en redes se podrá observar la red privada `red_privada` y en instancias se podrá observar la instancia `vm_privada` en estado activo, conectada a la red privada `red_privada` y utilizando la imagen `cirros` y el *flavor*

`m1.micro`. El router `router_privado` también estará presente, conectando la red privada a la red pública, permitiendo la comunicación entre ambas redes. Estos cuadros proporcionan una vista clara de los recursos creados por la plantilla de Heat y permiten gestionar y monitorear su estado y configuración desde la interfaz gráfica de Horizon.

Dentro de horizon, en la pestaña de redes se podrá observar la red privada creada por la plantilla de Heat, llamada `red_privada`. En topología de red, se podrá ver la estructura de la red, incluyendo la subred `subred_privada` y el router `router_privado` que conecta la red privada a la red pública. Esta visualización facilita la comprensión de cómo están interconectados los diferentes componentes de la red creada por la plantilla de Heat.

Figura 12. Topología de red creada por plantilla de Heat.



Nota. Elaboración propia.

8.3. Elasticidad horizontal mediante autoscaling groups

La elasticidad horizontal es una característica clave en entornos de nube que permite ajustar dinámicamente la cantidad de recursos computacionales en función de la demanda. En OpenStack, esta funcionalidad se puede implementar utilizando grupos de autoescalado *(autoscaling groups)* junto con servicios como Heat para la orquestación y Aodh para la gestión de alarmas. Los grupos de autoescalado permiten definir políticas que especifican cuándo y cómo escalar los recursos, ya sea aumentando o disminuyendo el número de instancias en respuesta a métricas específicas, como el uso de CPU o la carga de red. Al configurar un grupo de autoescalado, se pueden establecer umbrales para las métricas monitoreadas, de modo que cuando se superen ciertos límites, se desencadenen acciones automáticas para agregar o eliminar instancias. Esto no solo mejora la eficiencia operativa al garantizar que los recursos estén disponibles cuando se necesitan, sino que también ayuda a optimizar

los costos al reducir la cantidad de recursos durante períodos de baja demanda. La implementación de la elasticidad horizontal en OpenStack es esencial para mantener un rendimiento óptimo y una alta disponibilidad en aplicaciones desplegadas en la nube.

Frente a la elasticidad vertical, el modelo horizontal presenta ventajas sustanciales, especialmente en entornos de demanda impredecible. La principal fortaleza del escalado horizontal es su escalabilidad casi ilimitada, ya que no está restringido por los límites físicos de un único servidor, a diferencia del escalado vertical. Además, ofrece una inherente alta disponibilidad y tolerancia a fallos ya que si un nodo de cómputo falla, el balanceador de carga simplemente redirige el tráfico a los nodos restantes, eliminando el punto único de fallo (SPOF). La desventaja principal radica en la complejidad arquitectónica que exige la aplicación, esta debe ser capaz de gestionar el estado de manera distribuida y sincronizada entre múltiples nodos.

Los casos de uso de esta característica son variados, desde aplicaciones web que experimentan picos de tráfico impredecibles hasta servicios de procesamiento de datos que requieren recursos adicionales durante períodos específicos. Al aprovechar los grupos de autoescalado, las organizaciones pueden garantizar que sus aplicaciones permanezcan receptivas y eficientes, adaptándose automáticamente a las fluctuaciones en la carga de trabajo sin intervención manual. Un ejemplo de uso concreto es el escalado automático de servidores web en función del tráfico entrante, donde se pueden agregar instancias adicionales durante horas pico y reducirlas durante períodos de menor actividad, asegurando así una experiencia de usuario consistente y un uso eficiente de los recursos.

Un ejemplo claro y enfocado en la Universidad del Valle de Guatemala es el manejo de picos de demanda durante los periodos de inscripción estudiantil. Normalmente, el tráfico del portal web de la universidad es moderado. Sin elasticidad, la UVG se vería obligada a mantener servidores dimensionados para la carga máxima, lo que resultaría en un subaprovechamiento de recursos y costos excesivos la mayor parte del tiempo. Con la implementación de elasticidad horizontal en una infraestructura como OpenStack, la UVG puede configurar su portal para que automáticamente despliegue y use solo la capacidad mínima necesaria. Cuando miles de estudiantes intenten acceder al sistema simultáneamente, el autoescalado se activaría y crearía instantáneamente réplicas de las máquinas virtuales del servidor web y de aplicación. Esto garantiza una experiencia de usuario fluida y sin caídas (cero latencia). Una vez que la presión del tráfico se normaliza, el sistema automáticamente daría de baja las máquinas virtuales adicionales, asegurando que la universidad solo pague por los recursos utilizados durante el pico de demanda.

8.3.1. Creación de ambiente de elasticidad horizontal

Para implementar la elasticidad horizontal en OpenStack, es necesario crear un ambiente que incluya los componentes y servicios requeridos. Esto implica la configuración de un grupo de autoescalado, así como la definición de las políticas de escalado y las métricas a monitorear. A continuación, se describen los pasos para crear este ambiente.

Verificación de servicios de autoscaling

Antes de iniciar la configuración de la elasticidad horizontal, es importante validar que los servicios necesarios estén instalados y en correcto funcionamiento, para esto realizaremos distintas pruebas. El comando **openstack service list** permite visualizar los servicios registrados en el catálogo de Keystone, en este caso se buscarán los servicios relacionados con la elasticidad horizontal: Gnocchi (almacenamiento de métricas), Aodh (alarmas), Heat (orquestación) y Heat-cfn (compatibilidad con CloudFormation). Esta verificación es crucial para asegurar que los servicios necesarios para la elasticidad horizontal estén disponibles y operativos antes de proceder con la configuración y despliegue de recursos.

Cuadro 77. Verificación de servicios de autoscaling

```
openstack service list | egrep 'alarming|metric|orchestration|cloudformation'
```

Nota. Elaboración propia.

Cuadro 78. Respuesta esperada a verificación de servicios de autoscaling

```
| ID                               | Name      | Type            |
|----------------------------------|-----------|-----------------|
| 5b9b82f5d67f4eb9884762f2376505bb | gnocchi   | metric          |
| 5d2efde9f12348e9ba5ae64db912970f | aodh      | alarming        |
| 81ea1d340f564191a3e4be69f1633948 | heat-cfn  | cloudformation  |
| 8d74a11ca6314e89947c09db6f0c730a | heat      | orchestration   |
```

Nota. Elaboración propia.

La respuesta nos indica que dichos servicios clave están disponibles y en funcionamiento. Esto es esencial para la implementación de la elasticidad horizontal, ya que estos servicios trabajan en conjunto para monitorear el estado de las instancias, gestionar alarmas y orquestar la creación y eliminación de recursos según las necesidades del sistema.

A continuación, se verifican los *endpoints* de estos servicios para asegurar que estén correctamente configurados y accesibles. Esto es importante para garantizar que los servicios puedan ser utilizados por los clientes y otros componentes de OpenStack.

Cuadro 79. Verificación de *endpoints* de servicios de *autoscaling*

```
openstack endpoint list | egrep 'aodh|gnocchi|heat|heat-cfn'
```

Nota. Elaboración propia.

Cuadro 80. Respuesta esperada a verificación de *endpoints* de servicios de *autoscaling*

```
| 42723c5ea7364723bb1c23797221f07f | RegionOne | aodh         | alarming        |
    ↪ True    | internal  | http://192.168.74.69:8042                |
| 46230178010f467bab310e689104d046 | RegionOne | heat-cfn     | cloudformation  |
    ↪ True    | public    | http://192.168.74.69:8000/v1             |
```

```
| b7858faefcba400f9332fc53c3d82382 | RegionOne | aodh         | alarming        |
    ↪ True    | public    | http://192.168.74.69:8042                |
| bd7e4fda16fa427f8950c1e5036ee693 | RegionOne | heat         | orchestration   |
    ↪ True    | public    | http://192.168.74.69:8004/v1/%(tenant_id)s |
| be359f1e84c548ff84c194725bdb91b7 | RegionOne | gnocchi      | metric          |
    ↪ True    | public    | http://192.168.74.69:8041                |
| e9ec4379e4a34a86bf7e0ba97a8ee880 | RegionOne | heat         | orchestration   |
    ↪ True    | internal  | http://192.168.74.69:8004/v1/%(tenant_id)s |
| f8306273ae7b42f0a257f05af67ac8e1 | RegionOne | gnocchi      | metric          |
    ↪ True    | internal  | http://192.168.74.69:8041                |
| fb18e3a806624e30a127f7462e10dd99 | RegionOne | heat-cfn     | cloudformation  |
    ↪ True    | internal  | http://192.168.74.69:8000/v1             |
```

Nota. Elaboración propia.

El comando Openstack endpoint list muestra los *endpoints* de los servicios registrados en el catálogo de Keystone. Cada *endpoint* representa una URL a la que se puede acceder para interactuar con un servicio específico. En este caso, se realiza una búsqueda de los *endpoints* relacionados con los servicios de elasticidad horizontal. La respuesta indica que los *endpoints* están correctamente configurados y accesibles.

Cuadro 81. Verificación de contenedores Heat

```
docker ps --format '{{.Names}}' | egrep 'heat_api$|heat_engine$|heat_api_cfn$'
```

Nota. Elaboración propia.

Cuadro 82. Respuesta esperada a verificación de contenedores Heat

```
heat_engine
heat_api_cfn
heat_api
```

Nota. Elaboración propia.

El comando *docker ps* muestra todos los contenedores en ejecución del sistema, en este caso se especifica los nombres de los contenedores específicos que se desean visualizar, en este caso los relacionados con Heat y busca únicamente los nombres de los contenedores en lugar de la tabla completa, esto permite una visualización más clara y concisa de los contenedores relevantes y que están activos.

El contenedor heat_api expone la API de Heat, que permite la gestión de plantillas de infraestructura como código. Heat_engine es el motor que procesa las plantillas y gestiona la creación, actualización y eliminación de recursos. Heat_api_cfn proporciona compatibilidad con la API de AWS CloudFormation, permitiendo a los usuarios utilizar plantillas de CloudFormation en OpenStack.

El comando **openstack metric status** permite verificar el estado del servicio de métricas de Gnocchi. Este comando es esencial para asegurarse de que Gnocchi esté funcionando correctamente y que las métricas se estén procesando adecuadamente.

Cuadro 83. Verificación del estado de métricas de Gnocchi

```
openstack metric status
```

Nota. Elaboración propia.

Cuadro 84. Respuesta esperada a verificación del estado de métricas de Gnocchi

```
+-----------------------------------------------------+-------+
| Field                                               | Value |
+-----------------------------------------------------+-------+
| storage/total number of measures to process         | 0     |
| storage/number of metric having measures to process | 0     |
| metricd/processors                                  | None  |
+-----------------------------------------------------+-------+
```

Nota. Elaboración propia.

La respuesta muestra información sobre cantidad de medidas pendientes, métricas en proceso y si hay *workers* activos procesando medidas. La respuesta indica que no hay medidas pendientes, ni métricas en proceso, lo que sugiere que Gnocchi está funcionando correctamente y no hay acumulación de datos pendientes. También indica que no hay *workers* activos procesando medidas, que en este caso es normal ya que no se han insertado métricas aún.

Cuadro 85. Verificación de contenedores Gnocchi

```
docker ps --format '{{.Names}}' | grep gnocchi
```

Nota. Elaboración propia.

Cuadro 86. Respuesta esperada para verificación de contenedores Gnocchi

```
gnocchi_metricd
gnocchi_api
```

Nota. Elaboración propia.

Al igual que en el caso de los contenedores de Heat, este comando muestra los contenedores en ejecución relacionados con Gnocchi. El contenedor gnocchi_api expone la API de Gnocchi, que permite la gestión y consulta de métricas. El contenedor gnocchi_metricd procesa las métricas, almacenándolas y gestionando su ciclo de vida.

Creación de recurso y métrica de prueba

Para probar el funcionamiento de Gnocchi y Ceilometer, se puede crear un recurso genérico con una métrica asociada. Esto permite verificar que las métricas se crean, almacenanen y consulten de forma correcta.

Para crear un nuevo recurso, se utiliza el comando **openstack metric resource create**, el cual permite definir un recurso que será monitoreado. Se debe de indicar el ID del recurso, el tipo de recurso y el tipo de formato de salida. En este caso, se crea un recurso genérico llamado `test-res` que es guardado en la variable de entorno `RID` para facilidad de uso a futuro.

Cuadro 87. Creación de un recurso genérico

```
RID=$(openstack metric resource create test-res --type generic -f value -c id)
```

Nota. Elaboración propia.

La salida del comando muestra los detalles del recurso creado, incluyendo su ID, tipo y nombre. Este sirve como base para asociar métricas y realizar pruebas de monitoreo.

Cuadro 88. Detalles del recurso creado

```
+-----------------+--------------------------------------------------------------+
| Field           | Value                                                        |
+-----------------+--------------------------------------------------------------+
| id              | e6e6b4e6-3f09-5c73-8c0d-e764d9e600b6                         |
| type            | | generic                                                    |
| project_id      | 92ac5aaf7bdb411b9752b9a32bb46340:431b4b0e9d594207a4ef1df4bbe8|
| user_id         | 92ac5aaf7bdb411b9752b9a32bb46340                             |
| name            | test-res                                                     |
+-----------------+--------------------------------------------------------------+
```

Nota. Elaboración propia.

Ahora es necesario crear una métrica asociada a este recurso. Esto se realiza mediante el comando **openstack metric create**, el cual permite definir una métrica que será monitoreada. Se debe indicar el nombre de la métrica, la política de archivado que se desea utilizar y el ID del recurso al que se asociará la métrica. En este caso, se crea una métrica llamada `test-metric` asociada al recurso `test-res` utilizando la política de archivado `ceilometer-low-rate`. El ID de la métrica creada se almacena en la variable de entorno `MID` para su uso futuro.

Cuadro 89. Creación de una métrica asociada al recurso test-res

```
openstack metric create test-metric \
  --archive-policy-name ceilometer-low-rate \
  --resource-id "$RID"
```

Nota. Elaboración propia.

La salida del comando muestra los detalles de la métrica creada, incluyendo su ID, nombre, unidad, ID del recurso asociado y la política de archivado utilizada.

Cuadro 90. Detalles de la métrica creada test-metric

```
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| id                  | ebc86721-20a0-4d55-9529-         |
|                     | bdad347461d8                     |
| creator             | 92ac5aaf7bdb411b9752b9a32bb46340: |
|                     | 431b4b0e9d594207a4ef1df4bbe8463a  |
| name                | test-metric                      |
| unit                | None                             |
| resource_id         | e6e6b4e6-3f09-5c73-8c0d-         |
|                     | e764d9e600b6                     |
| archive_policy/name | cpu-rate                         |
+---------------------+----------------------------------+
```

Nota. Elaboración propia.

Al igual que con el recurso, es útil extraer el UUID de la métrica a una variable de entorno para su uso futuro, esto mantiene un orden e identificación clara de todos los recursos y métricas creados.

Cuadro 91. Obtención del ID de la métrica test-metric

```
#obtener id de la metrica
MID=$(openstack metric resource show "$RID" -f yaml | awk '/  test-metric:/{print
    ↪ $2}')
```

Nota. Elaboración propia.

Inyección y consulta de medidas en la métrica

El siguiente paso es realizar una llamada curl para insertar una medida en la métrica creada. Para ello, se debe obtener el token de autenticación y el endpoint de la API de métricas.

Primero es necesario obtener el token de autenticación, este se obtiene mediante el comando **openstack token issue**, el cual genera un token válido para autenticarse en los servicios de OpenStack. El token es almacenado en la variable de entorno `TOKEN` para su uso posterior.

Para obtener la URL endpoint interno de Gnocchi se usa el comando **openstack endpoint list**, el cual muestra todos los puntos finales de los servicios registrados en el catálogo de Keystone. Se filtra la salida para obtener únicamente la URL del servicio de métricas (Gnocchi) y se almacena en la variable de entorno `ENDPOINT`.

Por último, antes de realizar el curl, se obtiene la *timestamp* actual en formato UTC, el cual es necesario para registrar la medida con la hora correcta.

Cuadro 92. Obtención del token de autenticación y el endpoint de la API de métricas

```
TOKEN=$(openstack token issue -f value -c id)
ENDPOINT=$(openstack endpoint list --service metric --interface internal -f value -c
    ↪ URL)
TS=$(date -u +"%Y-%m-%dT%H:%M:%S+00:00")
```

Nota. Elaboración propia.

Con todas las variables de entorno extraídas correctamente, se puede realizar el curl. La salida esperada es el código 202, este indica que Gnocchi aceptó la medida y la procesará asíncronamente.

Cuadro 93. Inyección de medida en métrica con curl

```
curl -s -o /dev/null -w "%{http_code}\n" \
  -H "X-Auth-Token: $TOKEN" -H "Content-Type: application/json" \
  -d "[{\"timestamp\":\"$TS\",\"value\":1}]" \
  "$ENDPOINT/v1/metric/$MID/measures"
```

Nota. Elaboración propia.

Para consultar las medidas previamente inyectadas en la métrica test-metric se debe definir una variable de entorno con el timestamp de inicio de la consulta. Esto se puede lograr utilizando el comando date para obtener la hora actual menos una hora en formato UTC.

El comando permite consultar las medidas de una métrica específica en un rango de tiempo determinado. En este caso, se utiliza la variable de entorno `MID` para especificar la métrica que se desea consultar y la variable `START` para definir el inicio del rango de tiempo. La salida mostrará las medidas registradas en la métrica desde el timestamp especificado hasta el momento actual.

Cuadro 94. Consulta de medidas en métrica

```
START=$(date -u -d "1 hour ago" +"%Y-%m-%dT%H:%M:%S+00:00")
openstack metric measures show "$MID" --aggregation rate:mean --start "$START"
    ↪ --granularity 300
```

Nota. Elaboración propia.

Cuadro 95. Respuesta esperada a consulta de medidas en métrica

```
+---------------------------+-------------+-------+
| timestamp                 | granularity | value |
+---------------------------+-------------+-------+
| 2025-08-22T20:30:00-06:00 |       300.0 |   1.0 |
+---------------------------+-------------+-------+
```

Nota. Elaboración propia.

La respuesta nos indica el timestamp, que es la marca de tiempo donde se agrupó la medida, la granularidad indica el intervalo de agregación en segundos, y por último el valor de la medida

registrada. En este caso, se observa que se registró una medida con un valor de 1.0 en el timestamp especificado, con una granularidad de 300 segundos (5 minutos). Esto confirma que la inyección de la medida fue exitosa y que Gnocchi está almacenando y procesando las métricas correctamente.

Creación de plantilla HOT

Antes de crear la plantilla de orquestación es necesario verificar que todos los recursos a utilizar están disponibles, incluyendo la red y la subred.

Para esta implementación de elasticidad horizontal se utilizará una red y subred privada, con el objetivo de tener un entorno aislado y controlado para las instancias que se escalarán. La red y la subred se crearán utilizando los comandos de OpenStack Neutron.

Cuadro 96. Creación de red y subred para autoscaling

```
openstack network create autoscale-net
openstack subnet create autoscale-subnet \
  --network autoscale-net \
  --subnet-range 192.168.100.0/24 \
  --gateway 192.168.100.1
```

Nota. Elaboración propia.

Cuadro 97. Detalles de la red y subred creadas

```
+---------------------------+------------------------------+
| Field                     | Value                        |
+---------------------------+------------------------------+
| admin_state_up            | UP                           |
| availability_zone_hints   |                              |
| availability_zones        |                              |
| created_at                | 2025-08-23T02:31:58Z         |
| description               |                              |
| dns_domain                | None                         |
| id                        | 04eefc7e-35ce-47bb-8947-     |
|                           | 8a72835e6882                 |
| ipv4_address_scope        | None                         |
| ipv6_address_scope        | None                         |
| is_default                | None                         |
| is_vlan_qinq              | None                         |
| is_vlan_transparent       | None                         |
| mtu                       | 1450                         |
| name                      | autoscale-net                |
| port_security_enabled     | True                         |
| project_id                | 431b4b0e9d594207a4ef1df4bbe  |
|                           | 8463a                        |
```

```
| provider:network_type     | vxlan                            |
| provider:physical_network | None                             |
| provider:segmentation_id  | 661                              |
| qos_policy_id             | None                             |
| revision_number           | 1                                |
| router:external           | Internal                         |
| segments                  | None                             |
| shared                    | False                            |
| status                    | ACTIVE                           |
| subnets                   |                                  |
| tags                      |                                  |
| updated_at                | 2025-08-23T02:31:58Z             |
+---------------------------+----------------------------------+
+----------------------+----------------------------------+
| Field                | Value                            |
+----------------------+----------------------------------+
| allocation_pools     | 192.168.100.2-192.168.100.254    |
| cidr                 | 192.168.100.0/24                 |
| created_at           | 2025-08-23T02:32:00Z             |
| description          |                                  |
| dns_nameservers      |                                  |
| dns_publish_fixed_ip | None                             |
| enable_dhcp          | True                             |
| gateway_ip           | 192.168.100.1                    |
| host_routes          |                                  |
| id                   | afa3568b-772c-422b-896f-         |
|                      | 62614ac460bd                     |
| ip_version           | 4                                |
| ipv6_address_mode    | None                             |
| ipv6_ra_mode         | None                             |
| name                 | autoscale-subnet                 |
| network_id           | 04eefc7e-35ce-47bb-8947-         |
|                      | 8a72835e6882                     |
| project_id           | 431b4b0e9d594207a4ef1df4bbe8463a |
| revision_number      | 0                                |
| router:external      | False                            |
| segment_id           | None                             |
| service_types        |                                  |
| subnetpool_id        | None                             |
| tags                 |                                  |
| updated_at           | 2025-08-23T02:32:00Z             |
+----------------------+----------------------------------+
```

Nota. Elaboración propia.

Una vez creadas la red y la subred, se pueden verificar mediante los comandos **openstack network show** y **openstack subnet show**.

Cuadro 98. Verificación de la red para autoscaling

```
# verificacion
openstack network show autoscale-net -f yaml | egrep 'id:|name:|subnets:'
```

Nota. Elaboración propia.

Cuadro 99. Respuesta esperada a verificación de la red para autoscaling

```
id: 04eefc7e-35ce-47bb-8947-8a72835e6882
name: autoscale-net
project_id: 431b4b0e9d594207a4ef1df4bbe8463a
provider:segmentation_id: 661
qos_policy_id: null
subnets:
```

Nota. Elaboración propia.

Cuadro 100. Verificación de la subred para autoscaling

```
# verificacion
openstack subnet show autoscale-subnet -f yaml | egrep
    ↪ 'name:|cidr:|gateway_ip:|enable_dhcp'
```

Nota. Elaboración propia.

Cuadro 101. Respuesta esperada a verificación de la subred para autoscaling

```
cidr: 192.168.100.0/24
enable_dhcp: true
gateway_ip: 192.168.100.1
name: autoscale-subnet
```

Nota. Elaboración propia.

Como instancia a escalar horizontalmente se utilizará una imagen ligera llamada `cirros` y un *flavor* pequeño llamado `m1.micro`. Es importante verificar que estos recursos estén disponibles en el entorno de OpenStack antes de proceder con la creación de la plantilla de orquestación.

Cuadro 102. Verificación del flavor para autoscaling

```
openstack flavor show m1.micro -f yaml | egrep 'name:|ram:|vcpus:|disk:'
```

Nota. Elaboración propia.

Cuadro 103. Respuesta esperada a verificación del flavor para autoscaling

```
disk: 1
name: m1.micro
ram: 128
vcpus: 1
```

Nota. Elaboración propia.

Cuadro 104. Verificación de la imagen para autoscaling

```
openstack image show cirros -f yaml | egrep 'name:|status:|disk_format:'
```

Nota. Elaboración propia.

Cuadro 105. Respuesta esperada a verificación de la imagen para autoscaling

```
disk_format: qcow2
name: cirros
status: active
```

Nota. Elaboración propia.

8.3.2. Plantilla de orquestación con autoscaling

Las plantillas HOT (Heat Orchestration Template) son archivos en formato YAML que describen la infraestructura y los recursos que se desean crear y gestionar en OpenStack utilizando el servicio de orquestación Heat. Estas plantillas permiten definir de manera declarativa los recursos, sus propiedades, relaciones y dependencias, facilitando la automatización del despliegue y la gestión de infraestructuras complejas.

Para este caso los recursos a utilizar son las redes creadas, la imagen y el *flavor*. Además, se definirán políticas de escalado basadas en métricas de CPU, utilizando los servicios de Gnocchi y Aodh para monitorear y reaccionar a los cambios en la carga de trabajo. Se analizará la plantilla de orquestación por segmentos para entender su estructura y funcionalidad.

Parámetros de la plantilla

Cuadro 106. Parámetros de plantilla de orquestación para autoscaling

```
parameters:
  image:
    type: string
    default: cirros
    description: Imagen Nova.
  flavor:
    type: string
```

```
    default: m1.micro
    description: Flavor.
  network:
    type: string
    default: autoscale-net
    description: Red Neutron.

  min_size:
    type: number
    default: 1
  max_size:
    type: number
    default: 5
  desired_capacity:
    type: number
    default: 1

  scale_up_adjustment:
    type: number
    default: 1
  scale_down_adjustment:
    type: number
    default: -1
  cooldown_up:
    type: number
    default: 120
  cooldown_down:
    type: number
    default: 180

  # Gnocchi
  granularity:
    type: number
    default: 300
    description: Debe existir en la archive policy (ceilometer-low-rate soporta 300s).

  evaluation_periods_up:
    type: number
    default: 1
  evaluation_periods_down:
    type: number
    default: 1

  # Umbrales CPU en ns/s.
```

```
  cpu_high_threshold:
    type: number
    default: 800000000   # ~80% de 1 vCPU
  cpu_low_threshold:
    type: number
    default: 200000000   # ~20% de 1 vCPU
```

Nota. Elaboración propia.

La sección de parámetros define los valores que el usuario puede personalizar al momento de lanzar el stack, como valores predeterminados usaremos la imagen `cirros`, el *flavor* `m1.micro` y la red `autoscale-net`. Además, se definen los tamaños frontera deseados del grupo de autoescalado, así como los ajustes de escalado y los períodos de enfriamiento para las políticas de escalado. También se incluyen parámetros relacionados con Gnocchi, como la granularidad de las métricas, los períodos de evaluación y los umbrales de CPU para activar las políticas. Estos parametros pueden ser ajustados según las necesidades específicas del entorno y la carga de trabajo al momento de lanzar la plantilla.

Esta parametrización permite una mayor flexibilidad y reutilización de la plantilla en diferentes escenarios y configuraciones, ya que el resto de la plantilla está definida en función de estos parámetros, creando una infraestructura adaptable y escalable.

Recursos de la plantilla

Cuadro 107. Recursos de la plantilla de orquestación para autoscaling

```
resources:
  asg:
    type: OS::Heat::AutoScalingGroup
    properties:
      min_size: { get_param: min_size }
      max_size: { get_param: max_size }
      desired_capacity: { get_param: desired_capacity }
      resource:
        type: OS::Nova::Server
        properties:
          name:
            list_join: ['-', ['autoscale', { get_param: 'OS::stack_name' }]]
          image: { get_param: image }
          flavor: { get_param: flavor }
          networks:
            - network: { get_param: network }
          config_drive: true
          metadata:
            asg: { get_param: 'OS::stack_name' }
```

Nota. Elaboración propia.

La sección de recursos define un grupo de autoescalado llamado `asg` utilizando el recurso `OS::Heat::AutoScalingGroup`. Este grupo tiene propiedades que definen su tamaño mínimo, máximo y deseado, utilizando los parámetros definidos anteriormente. Dentro del grupo de autoescalado, se especifica el recurso que se va a escalar, que en este caso es una instancia de servidor Nova. Las propiedades de la instancia incluyen el nombre, la imagen, el *flavor*, la red a la que se conectará, la activación del *config_ drive* y los metadatos asociados. El nombre de la instancia se genera dinámicamente utilizando el nombre del stack, lo que facilita la identificación de las instancias dentro del grupo de autoescalado.

Cuadro 108. Políticas de escalado de la plantilla de orquestación para autoscaling

```
scale_up_policy:
    type: OS::Heat::ScalingPolicy
    properties:
      auto_scaling_group_id: { get_resource: asg }
      adjustment_type: change_in_capacity
      scaling_adjustment: { get_param: scale_up_adjustment }
      cooldown: { get_param: cooldown_up }

  scale_down_policy:
    type: OS::Heat::ScalingPolicy
    properties:
      auto_scaling_group_id: { get_resource: asg }
      adjustment_type: change_in_capacity
      scaling_adjustment: { get_param: scale_down_adjustment }
      cooldown: { get_param: cooldown_down }
```

Nota. Elaboración propia.

Las políticas de escalado definen las acciones a realizar cuando se activan. La política de escalado hacia arriba aumenta el número de instancias en el grupo de autoescalado en una cantidad definida por el parámetro `scale_up_adjustment`, con un período de enfriamiento definido por el parámetro `cooldown_up`. La política de escalado hacia abajo disminuye el número de instancias en el grupo de autoescalado en una cantidad definida por el parámetro `scale_down_adjustment`, con un período de enfriamiento definido por el parámetro `cooldown_down`. Estas políticas permiten ajustar dinámicamente la capacidad del grupo de autoescalado en función de la carga de trabajo y las métricas monitoreadas.

Cuadro 109. Alarmas de CPU de la plantilla de orquestación para autoscaling

```
# Alarma de SUBIDA (consulta como STRING JSON; filtrado por display_name LIKE
   ↪ autoscale-<stack>%)
  cpu_high_alarm:
    type: OS::Aodh::GnocchiAggregationByResourcesAlarm
    properties:
      description: Scale up when group CPU is high (display_name like
   ↪ autoscale-<stack>%)
```

```
      metric: cpu
      aggregation_method: rate:mean
      resource_type: instance
      threshold: { get_param: cpu_high_threshold }
      comparison_operator: gt
      granularity: { get_param: granularity }
      evaluation_periods: { get_param: evaluation_periods_up }
      query:
        str_replace:
          template:
    ↪ '[{"field":"display_name","op":"like","value":"autoscale-$STACK%"}]'
          params:
            $STACK: { get_param: 'OS::stack_name' }
      alarm_actions:
        - { get_attr: [scale_up_policy, alarm_url] }
      repeat_actions: true

  # Alarma de BAJADA (mismo filtrado)
  cpu_low_alarm:
    type: OS::Aodh::GnocchiAggregationByResourcesAlarm
    properties:
      description: Scale down when group CPU is low (display_name like
    ↪ autoscale-<stack>%)
      metric: cpu
      aggregation_method: rate:mean
      resource_type: instance
      threshold: { get_param: cpu_low_threshold }
      comparison_operator: lt
      granularity: { get_param: granularity }
      evaluation_periods: { get_param: evaluation_periods_down }
      query:
        str_replace:
          template:
    ↪ '[{"field":"display_name","op":"like","value":"autoscale-$STACK%"}]'
          params:
            $STACK: { get_param: 'OS::stack_name' }
      alarm_actions:
        - { get_attr: [scale_down_policy, alarm_url] }
      repeat_actions: true
```

Nota. Elaboración propia.

Las alarmas de CPU utilizan el recurso de Aodh y Gnocchi de agregación por recursos `OS::Aodh::GnocchiAggregationByResourcesAlarm` para monitorear el uso de CPU de las instancias en el grupo de autoescalado. La alarma de subida se activa cuando el uso de CPU supera un umbral

definido por el parámetro `cpu_high_threshold`, mientras que la alarma de bajada se activa cuando el uso de CPU cae por debajo de un umbral definido por el parámetro `cpu_low_threshold`. Ambas alarmas utilizan una consulta que filtra las instancias por su nombre para asegurarse de que solo las instancias del grupo de autoescalado sean monitoreadas. Cuando una alarma se activa, ejecuta la acción correspondiente para escalar hacia arriba o hacia abajo utilizando las URL proporcionadas por las políticas de escalado. El método de agregación utilizado es `rate:mean`, que calcula la tasa media de uso de CPU durante el período de evaluación definido. Los operadores de comparación utilizados son `gt` (greater than) para la alarma de subida y `lt` (less than) para la alarma de bajada.

La métrica de CPU con agregación `rate:mean` proporciona una visión más precisa del uso de CPU a lo largo del tiempo y de manera global, ya que utiliza la tasa media en lugar de valores instantáneos, lo que ayuda a evitar respuestas erráticas del sistema de autoescalado debido a picos momentáneos en el uso de CPU y ve el autoescalado como un sistema completo y no como instancias individuales. La dimensional que utiliza esta métrica es nanosegundos por segundo (ns/s), esta dimensional indica la cantidad de tiempo de CPU utilizado por segundo, expresado en nanosegundos. Esta métrica es la más adecuada para evaluar el rendimiento de la CPU en entornos virtualizados y es compatible con las políticas de autoescalado basadas en el uso de CPU. La otra opción común es utilizar la métrica de porcentaje de CPU (%), la opción utilizada es más precisa y adecuada ya que el uso de CPU depende de la cantidad de vCPU asignadas a cada instancia, y el porcentaje puede variar significativamente entre instancias con diferentes configuraciones de CPU, los ns/s proporcionan una medida más consistente y comparable del uso de CPU en entornos con múltiples instancias y configuraciones variadas al tener una unidad absoluta en lugar de relativa.

Salidas de la plantilla

Cuadro 110. Salidas de la plantilla de orquestación para autoscaling

```yaml
outputs:
  scale_up_url:
    description: URL CFN firmada para pruebas manuales (scale-up).
    value: { get_attr: [scale_up_policy, alarm_url] }
  scale_down_url:
    description: URL CFN firmada para pruebas manuales (scale-down).
    value: { get_attr: [scale_down_policy, alarm_url] }
```

Nota. Elaboración propia.

La sección de salidas define dos salidas que proporcionan las URL firmadas para activar manualmente las políticas de escalado hacia arriba y hacia abajo. Estas URL pueden ser utilizadas para probar las políticas de escalado sin necesidad de que se activen automáticamente mediante las alarmas de CPU. La descripción de cada salida indica su propósito, facilitando la identificación y el uso de estas URL en pruebas manuales o integraciones con otros sistemas. Estas URL son especialmente útiles para validar el comportamiento del grupo de autoescalado y las políticas definidas en la plantilla, además de permitir la activación manual en situaciones específicas o conexiones externas como servicios de monitoreo o automatización.

Plantilla final para orquestación de autoscaling

Cuadro 111. Plantilla de orquestación para autoscaling completa

```
heat_template_version: 2014-10-16
description: >
  AutoScalingGroup con politicas de escala y alarmas Aodh

parameters:
  image:
    type: string
    default: cirros
    description: Imagen Nova (nombre o ID).
  flavor:
    type: string
    default: m1.micro
    description: Flavor Nova.
  network:
    type: string
    default: autoscale-net
    description: Red Neutron.

  min_size:
    type: number
    default: 1
  max_size:
    type: number
    default: 5
  desired_capacity:
    type: number
    default: 1

  scale_up_adjustment:
    type: number
    default: 1
  scale_down_adjustment:
    type: number
    default: -1
  cooldown_up:
    type: number
    default: 120
  cooldown_down:
    type: number
    default: 180

```

```yaml
  # Gnocchi
  granularity:
    type: number
    default: 300
    description: Debe existir en la archive policy (ceilometer-low-rate soporta 300s).

  evaluation_periods_up:
    type: number
    default: 1
  evaluation_periods_down:
    type: number
    default: 1

  # Umbrales CPU en ns/s.
  cpu_high_threshold:
    type: number
    default: 800000000   # ~80% de 1 vCPU
  cpu_low_threshold:
    type: number
    default: 200000000   # ~20% de 1 vCPU

resources:
  asg:
    type: OS::Heat::AutoScalingGroup
    properties:
      min_size: { get_param: min_size }
      max_size: { get_param: max_size }
      desired_capacity: { get_param: desired_capacity }
      resource:
        type: OS::Nova::Server
        properties:
          name:
            list_join: ['-', ['autoscale', { get_param: 'OS::stack_name' }]]
          image: { get_param: image }
          flavor: { get_param: flavor }
          networks:
            - network: { get_param: network }
          config_drive: true
          metadata:
            asg: { get_param: 'OS::stack_name' }

  scale_up_policy:
    type: OS::Heat::ScalingPolicy
    properties:
```

```
      auto_scaling_group_id: { get_resource: asg }
      adjustment_type: change_in_capacity
      scaling_adjustment: { get_param: scale_up_adjustment }
      cooldown: { get_param: cooldown_up }

  scale_down_policy:
    type: OS::Heat::ScalingPolicy
    properties:
      auto_scaling_group_id: { get_resource: asg }
      adjustment_type: change_in_capacity
      scaling_adjustment: { get_param: scale_down_adjustment }
      cooldown: { get_param: cooldown_down }

  # Alarma de SUBIDA (consulta como STRING JSON; filtrado por display_name LIKE
    ↪ autoscale-<stack>%)
  cpu_high_alarm:
    type: OS::Aodh::GnocchiAggregationByResourcesAlarm
    properties:
      description: Scale up when group CPU is high (display_name like
    ↪ autoscale-<stack>%)
      metric: cpu
      aggregation_method: rate:mean
      resource_type: instance
      threshold: { get_param: cpu_high_threshold }
      comparison_operator: gt
      granularity: { get_param: granularity }
      evaluation_periods: { get_param: evaluation_periods_up }
      query:
        str_replace:
          template:
    ↪ '[{"field":"display_name","op":"like","value":"autoscale-$STACK%"}]'
          params:
            $STACK: { get_param: 'OS::stack_name' }
      alarm_actions:
        - { get_attr: [scale_up_policy, alarm_url] }
      repeat_actions: true

  # Alarma de BAJADA (mismo filtrado)
  cpu_low_alarm:
    type: OS::Aodh::GnocchiAggregationByResourcesAlarm
    properties:
      description: Scale down when group CPU is low (display_name like
    ↪ autoscale-<stack>%)
      metric: cpu
```

```
      aggregation_method: rate:mean
      resource_type: instance
      threshold: { get_param: cpu_low_threshold }
      comparison_operator: lt
      granularity: { get_param: granularity }
      evaluation_periods: { get_param: evaluation_periods_down }
      query:
        str_replace:
          template:
    ↪ '[{"field":"display_name","op":"like","value":"autoscale-$STACK%"}]'
          params:
            $STACK: { get_param: 'OS::stack_name' }
      alarm_actions:
        - { get_attr: [scale_down_policy, alarm_url] }
      repeat_actions: true

outputs:
  scale_up_url:
    description: URL CFN firmada para pruebas manuales (scale-up).
    value: { get_attr: [scale_up_policy, alarm_url] }
  scale_down_url:
    description: URL CFN firmada para pruebas manuales (scale-down).
    value: { get_attr: [scale_down_policy, alarm_url] }
```

Nota. Elaboración propia.

La plantilla completa combina todos los elementos descritos anteriormente, proporcionando una solución integral para el autoescalado de instancias en OpenStack basada en métricas de CPU. Esta plantilla puede ser utilizada para desplegar y gestionar automáticamente un grupo de instancias que se ajustan dinámicamente a la carga de trabajo, mejorando la eficiencia y la disponibilidad del servicio.

8.3.3. Modificación de archivos de configuración para Aodh y Heat

Antes de desplegar la plantilla, es importante modificar algunos archivos de configuración y asegurarse de que los servicios necesarios estén activos y funcionando correctamente en el entorno de OpenStack. Es necesario modificar la versión de la integración API con Keystone de los servicios de Aodh y Heat para que utilicen la versión v3, ya que las versiones anteriores no son compatibles con las funcionalidades requeridas para el autoescalado basado en métricas. Además, se debe asegurar que los servicios de Aodh y Gnocchi estén activos y funcionando correctamente, ya que son esenciales para la recopilación y monitoreo de métricas que activan las políticas de escalado.

Modificación de archivos de configuración de Aodh

Para esto es necesario modificar archivos de configuración. Para Aodh, los archivos a modificar se encuentran en las siguientes rutas:

- `/etc/kolla/aodh-api/aodh.conf`
- `/etc/kolla/aodh-evaluator/aodh.conf`
- `/etc/kolla/aodh-notifier/aodh.conf`
- `/etc/kolla/aodh-listener/aodh.conf`

En cada uno de estos archivos, se debe cambiar la línea que especifica la versión de la API de Keystone a `auth_version = v3`. Cada uno de estos archivos de configuración son iguales y deben quedar de la siguiente manera.

Cuadro 112. Archivo de configuración de Aodh modificado para usar Keystone v3

```
[DEFAULT]
auth_strategy = keystone
log_dir = /var/log/kolla/aodh
debug = False
evaluation_interval = 300
transport_url =
    ↪ rabbit://openstack:mBbgdN3CTMvxmpxA58NQ8oAqcgrcmSEwh4kOq5Yx@192.168.74.5:5672//
[api]
port = 8042
host = 192.168.74.5

[database]
connection = mysql+pymysql://aodh:8MxSdDLSzEc8Yq7pjbZLVYYzy7XRPq1jBlCoZd3v
    ↪ @192.168.74.69:3306/aodh
connection_recycle_time = 10
max_pool_size = 1

[keystone_authtoken]
service_type = alarming
memcache_security_strategy = ENCRYPT
memcache_secret_key = 951HRpvVrK2NRDUYBmTPHf3hwI9nBfF5tVypHIJf
memcached_servers = 192.168.74.5:11211
www_authenticate_uri = http://192.168.74.69:5000/v3
project_domain_name = Default
project_name = service
user_domain_name = Default
username = aodh
password = 4sGPTdc1anMk3m7VnzFpy10gyzhdcj0Nx3v69Omy
auth_url = http://192.168.74.69:5000/v3
```

```
auth_type = password
cafile =
region_name = RegionOne

[oslo_middleware]
enable_proxy_headers_parsing = true

[service_credentials]
auth_url = http://192.168.74.69:5000/v3
region_name = RegionOne
password = 4sGPTdc1anMk3m7VnzFpy10gyzhdcj0Nx3v69Omy
username = aodh
project_name = service
project_domain_id = default
user_domain_id = default
auth_type = password
interface = internal
cafile =

[oslo_messaging_notifications]
transport_url =
    ↪ rabbit://openstack:mBbgdN3CTMvxmpxA58NQ8oAqcgrcmSEwh4kOq5Yx@192.168.74.5:5672//
driver = messagingv2
topics = notifications

[oslo_messaging_rabbit]
use_queue_manager = true
heartbeat_in_pthread = False
rabbit_quorum_queue = true
rabbit_stream_fanout = true
rabbit_qos_prefetch_count = 1
rabbit_transient_quorum_queue = true

[oslo_concurrency]
lock_path = /var/lib/aodh/tmp
```

Nota. Elaboración propia.

Después de realizar estos cambios, es necesario reiniciar los servicios de Aodh. Esto se puede hacer utilizando los comandos de gestión de servicios correspondientes en el entorno de OpenStack.

Cuadro 113. Reinicio de los servicios de Aodh

```
docker restart aodh_api aodh_evaluator aodh_notifier aodh_listener
```

Nota. Elaboración propia.

Una vez reiniciados los contenedores, es importante verificar que los servicios de Aodh estén activos y funcionando correctamente. Esto se puede hacer utilizando el comando **docker ps** para listar los contenedores en ejecución y asegurarse de que los contenedores de Aodh estén en estado *Up*. Se debe esperar unos minutos para que todos los servicios se inicien correctamente.

Cuadro 114. Verificación de los servicios de Aodh

```
docker ps --format '{{.Names}}' | grep aodh
```

Nota. Elaboración propia.

Cuadro 115. Respuesta esperada para verificación de los servicios de Aodh

```
aodh_api
aodh_evaluator
aodh_notifier
aodh_listener
```

Nota. Elaboración propia.

Modificación de archivos de configuración de Heat

Los archivos de configuración de Heat que necesitan ser modificados se encuentran en las siguientes rutas:

- `/etc/kolla/heat-api/heat.conf`
- `/etc/kolla/heat-api-cfn/heat.conf`
- `/etc/kolla/heat-engine/heat.conf`

En cada uno de estos archivos, se debe cambiar la línea que especifica la versión de la API de Keystone a `auth_version = v3`. Cada uno de estos archivos de configuración son iguales y deben quedar de la siguiente manera.

Cuadro 116. Archivo de configuración de Heat modificado para usar Keystone v3

```
[DEFAULT]
debug = False
log_dir = /var/log/kolla/heat
log_file = $log_dir/heat-engine.log
heat_metadata_server_url = http://192.168.74.69:8000
heat_waitcondition_server_url = http://192.168.74.69:8000/v1/waitcondition
heat_stack_user_role = heat_stack_user
stack_domain_admin = heat_domain_admin
stack_domain_admin_password = Uzfi1tJrtmcN841ZdeFKUXCjF8qgPH5K1ZPxrLIa
stack_user_domain_name = heat_user_domain
num_engine_workers = 5
```

```
transport_url =
    ↪ rabbit://openstack:mBbgdN3CTMvxmpxA58NQ8oAqcgrcmSEwh4kOq5Yx@192.168.74.5:5672//
region_name_for_services = RegionOne
server_keystone_endpoint_type = public

[database]
connection = mysql+pymysql://heat:
    ↪ meoJOcHN7k2Xxl04ba6ll7NJy0zSBnjlhhKl8NRX@192.168.74.69:3306/heat
connection_recycle_time = 10
max_pool_size = 1
max_retries = -1

[keystone_authtoken]
service_type = orchestration
www_authenticate_uri = http://192.168.74.69:5000/v3
auth_url = http://192.168.74.69:5000/v3
auth_type = password
project_domain_id = default
user_domain_id = default
project_name = service
username = heat
password = JroKIEw8dENhPMsaH7klOT8YyFgdntIT622w8yzL
cafile =
region_name = RegionOne
memcache_security_strategy = ENCRYPT
memcache_secret_key = 951HRpvVrK2NRDUYBmTPHf3hwI9nBfF5tVypHIJf
memcached_servers = 192.168.74.5:11211

[cache]
backend = oslo_cache.memcache_pool
enabled = true
memcache_servers = 192.168.74.5:11211

[trustee]
auth_url = http://192.168.74.69:5000/v3
auth_type = password
user_domain_id = default
username = heat
password = JroKIEw8dENhPMsaH7klOT8YyFgdntIT622w8yzL

[ec2authtoken]
auth_uri = http://192.168.74.69:5000/v3
auth_url = http://192.168.74.69:5000/v3

```

```
[oslo_messaging_notifications]
transport_url =
    ↪ rabbit://openstack:mBbgdN3CTMvxmpxA58NQ8oAqcgrcmSEwh4kOq5Yx@192.168.74.5:5672//
driver = messagingv2
topics = notifications

[oslo_messaging_rabbit]
use_queue_manager = true
heartbeat_in_pthread = False
rabbit_quorum_queue = true
rabbit_stream_fanout = true
rabbit_qos_prefetch_count = 1
rabbit_transient_quorum_queue = true

[clients]
endpoint_type = internalURL
ca_file =

[oslo_middleware]
enable_proxy_headers_parsing = true

[volumes]
backups_enabled = True

[oslo_concurrency]
lock_path = /var/lib/heat/tmp
```

Nota. Elaboración propia.

Después de realizar estos cambios, es necesario reiniciar los servicios de Heat para que los cambios surtan efecto. Esto se puede hacer utilizando los comandos de gestión de servicios correspondientes en el entorno de OpenStack.

Cuadro 117. Reinicio de los servicios de Heat

```
systemctl restart heat-api
systemctl restart heat-api-cfn
systemctl restart heat-engine
```

Nota. Elaboración propia.

Luego de reiniciados los servicios, es importante verificar que los servicios de Heat estén activos y funcionando correctamente. Esto se puede hacer utilizando el comando **docker ps** para listar los contenedores en ejecución y asegurarse de que los contenedores de Heat estén en estado *UP*, se debe esperar unos minutos para que todos los servicios se inicien correctamente.

Cuadro 118. Verificación de los servicios de Heat

```
docker ps --format '{{.Names}}' | grep heat
```

Nota. Elaboración propia.

Cuadro 119. Respuesta esperada a verificación de los servicios de Heat

```
heat_api
heat_api_cfn
heat_engine
```

Nota. Elaboración propia.

Creación de archivos de configuración personalizados

Es importante crear unos archivos de configuración de Heat y Aodh con los cambios realizados para futuras referencias o despliegues en otros entornos, esto debido a que los archivos de configuración originales pueden ser sobrescritos durante actualizaciones o reinstalaciones de los servicios y se perderán los cambios realizados. Se debe crear una carpeta para cada servicio donde se almacenarán los archivos de configuración modificados en la siguiente ruta `/etc/kolla/config`.

Al momento de desplegar los servicios de OpenStack utilizando Kolla-Ansible, se puede especificar la ruta de estos archivos de configuración personalizados para que sean utilizados en lugar de los archivos predeterminados. Esto garantiza que los cambios realizados se mantengan intactos y se apliquen correctamente durante el despliegue de los servicios.

Cuadro 120. Creación de carpeta para archivos de configuración de Aodh

```
mkdir -p /etc/kolla/config/aodh
nano /etc/kolla/config/aodh/aodh.conf
```

Nota. Elaboración propia.

El archivo a crear debe de contener únicamente los cambios realizados respecto al archivo original, debido a que Kolla-Ansible fusiona los archivos de configuración personalizados con los archivos predeterminados durante el despliegue de los servicios. El archivo a crear debe quedar de la siguiente manera.

Cuadro 121. Archivo de configuración personalizado de Aodh para usar Keystone v3

```
[keystone_authtoken]
www_authenticate_uri = http://192.168.74.69:5000/v3
auth_url = http://192.168.74.69:5000/v3

[service_credentials]
auth_url = http://192.168.74.69:5000/v3
```

Nota. Elaboración propia.

Cuadro 122. Creación de carpeta para archivos de configuración de Heat

```
mkdir -p /etc/kolla/config/heat
nano /etc/kolla/config/heat/heat.conf
```

Nota. Elaboración propia.

De misma forma, el archivo de configuración personalizado de Heat debe contener únicamente los cambios realizados respecto al archivo original y debe quedar de la siguiente manera.

Cuadro 123. Archivo de configuración personalizado de Heat para usar Keystone v3

```
[keystone_authtoken]
www_authenticate_uri = http://192.168.74.69:5000/v3
auth_url = http://192.168.74.69:5000/v3

[trustee]
auth_url = http://192.168.74.69:5000/v3

[ec2authtoken]
auth_uri = http://192.168.74.69:5000/v3
auth_url = http://192.168.74.69:5000/v3
```

Nota. Elaboración propia.

Una vez realizados estos cambios y reiniciados los servicios, el entorno de OpenStack estará preparado para desplegar la plantilla de orquestación que implementa el autoescalado basado en métricas de CPU utilizando Aodh y Gnocchi.

8.3.4. Lanzamiento y verificación de la plantilla

Para lanzar la plantilla de orquestación se utiliza el comando de **openstack stack create**, indicando el nombre del archivo con la definición de la plantilla y el nombre del stack a crear. La lista completa de stacks creados pueden ser visualizados mediante el comando **openstack stack list**.

Cuadro 124. Lanzamiento de la plantilla de orquestación para autoscaling

```
openstack stack create -t asg-aodh-gnocchi.yaml asg-demo
```

Nota. Elaboración propia.

Luego de lanzar la plantilla, se mostrará un mensaje indicando que la creación del stack ha comenzado. Es importante monitorear el progreso de la creación del stack para asegurarse de que todos los recursos se creen correctamente. Esto se puede hacer utilizando el comando **openstack stack show**, que proporciona detalles sobre el estado del stack y sus recursos asociados.

Cuadro 125. Detalles del stack creado asg-demo

```
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| id                  | fac243b0-04e9-487e-9fb3-         |
|                     | 971a24972a8f                     |
| stack_name          | asg-demo                         |
| description         | AutoScalingGroup en red interna, |
|                     | con politicas y etiqueta de      |
|                     | filtrado.                        |
| creation_time       | 2025-08-23T02:35:06Z             |
| updated_time        | None                             |
| stack_status        | CREATE_IN_PROGRESS               |
| stack_status_reason | Stack CREATE started             |
+---------------------+----------------------------------+
```

Nota. Elaboración propia.

Para ver una lista de todos los stacks creados en el entorno de OpenStack, se utiliza el comando **openstack stack list**. Este comando muestra una tabla con información básica sobre cada stack, incluyendo su ID, nombre, proyecto asociado, estado y tiempos de creación y actualización. Además, para obtener detalles específicos sobre los recursos dentro de un stack, se puede utilizar el comando **openstack stack resource list** junto con el nombre del stack y la opción –nested-depth para mostrar recursos anidados.

Cuadro 126. Listado de stacks en OpenStack

```
openstack stack list
```

Nota. Elaboración propia.

Cuadro 127. Detalles de un stack en OpenStack

```
+----------+------------+----------+--------------+---------------+--------------+
| ID       | Stack Name | Project  | Stack Status | Creation Time | Updated Time |
+----------+------------+----------+--------------+---------------+--------------+
| fac243b0 | asg-demo   | 431b4b0e | CREATE_COMPL | 2025-08-      | None         |
| -04e9-   |            | 9d594207 | ETE          | 23T02:35:06Z  |              |
| 487e-    |            | a4ef1df4 |              |               |              |
| 9fb3-    |            | bbe8463a |              |               |              |
| 971a2497 |            |          |              |               |              |
| 2a8f     |            |          |              |               |              |
+----------+------------+----------+--------------+---------------+--------------+
```

Nota. Elaboración propia.

Cuadro 128. Listado de recursos en el stack asg-demo

```
openstack stack resource list asg-demo --nested-depth 5
```

Nota. Elaboración propia.

Cuadro 129. Respuesta esperada recortada a listado de recursos en el stack asg-demo

```
+------------------+----------------------------------------------+
| resource_name    | resource_type              | resource_status |
+------------------+----------------------------------------------+
| scale_down_policy | OS::Heat::ScalingPolicy    | CREATE_COMPLETE |
| scale_up_policy   | OS::Heat::ScalingPolicy    | CREATE_COMPLETE |
| asg               | OS::Heat::AutoScalingGroup | CREATE_COMPLETE |
| izzzicu66go6      | OS::Nova::Server           | CREATE_COMPLETE |
+------------------+----------------------------------------------+
```

Nota. Elaboración propia.

También es posible visualizar los eventos del stack en tiempo real mediante el comando **openstack stack event list**. Este nos permite monitorear el progreso de la creación del stack y detectar posibles errores visualizando el estado de los recursos de manera más granular y detallada.

Cuadro 130. Listado de eventos en el stack asg-demo

```
openstack stack event list asg-demo --nested-depth 5 --limit 10
```

Nota. Elaboración propia.

Cuadro 131. Respuesta esperada a listado de eventos en el stack asg-demo

```
2025-08-23 02:35:07Z [asg-demo]: CREATE_IN_PROGRESS  Stack CREATE started
2025-08-23 02:35:07Z [asg-demo.asg]: CREATE_IN_PROGRESS  state changed
2025-08-23 02:35:07Z [asg-demo.asg]: CREATE_IN_PROGRESS  Stack CREATE started
2025-08-23 02:35:07Z [asg-demo.asg.fgsopvfoboth]: CREATE_IN_PROGRESS  state changed
2025-08-23 02:35:21Z [asg-demo.asg.fgsopvfoboth]: CREATE_COMPLETE  state changed
2025-08-23 02:35:21Z [asg-demo.asg]: CREATE_COMPLETE  Stack CREATE completed
    ↪ successfully
2025-08-23 02:35:23Z [asg-demo.asg]: CREATE_COMPLETE  state changed
2025-08-23 02:35:23Z [asg-demo.scale_up_policy]: CREATE_IN_PROGRESS  state changed
2025-08-23 02:35:23Z [asg-demo.scale_down_policy]: CREATE_IN_PROGRESS  state changed
2025-08-23 02:35:24Z [asg-demo.scale_up_policy]: CREATE_COMPLETE  state changed
```

Nota. Elaboración propia.

La respuesta esperada muestra la instancia creada por *default* en el grupo de autoescalado, con su ID, nombre y estado actual. En este caso, la instancia debería estar en estado `ACTIVE` y con el `Power State` en `Running`, indicando que la instancia está funcionando correctamente.

8.3.5. Verificación de funcionamiento de plantilla de autoscaling

Una vez que el stack ha sido creado exitosamente, es importante verificar que las políticas de escalado y las alarmas estén funcionando correctamente. Esto se puede hacer obteniendo las URL de señalización generadas por las políticas de escalado y utilizando estas URL para activar manualmente las políticas de escalado. Además, se pueden consultar las métricas de CPU de las instancias en el grupo de autoescalado para asegurarse de que las alarmas se disparen correctamente cuando los umbrales definidos sean alcanzados.

Obtención de URL de señalización y métricas de instancias

Para obtener las URL de señalización que permiten activar las políticas de escalado, se utiliza el comando **openstack stack output show**. Este comando muestra las salidas definidas en la plantilla de orquestación, incluyendo las URL generadas para las políticas de escalado hacia arriba y hacia abajo. Estas URL representan señalizaciones seguras tipo *webhook* que cuando se acceden mediante HTTP POST, activan las políticas de escalado correspondientes.

Cuadro 132. Consulta de URL de escalado hacia arriba

```
openstack stack output show asg-demo scale_up_url
```

Nota. Elaboración propia.

Cuadro 133. Respuesta esperada a consulta de URL de escalado hacia arriba

```
+--------------+------------------------------------------------------------------+
| Field        | Value                                                            |
+--------------+------------------------------------------------------------------+
| output_key   | scale_up_url                                                     |
| description  | URL CFN firmada para pruebas manuales (scale-up).                |
| output_value | http://192.168.74.69:8000/v1/signal/arn%3Aopenstack%3Aheat%3     |
|              | A%3A5260c7767b584c58a1da728373b2d8f3%3Astacks/asg-demo/13ba5     |
|              | 26b-5046-40d2-aac9-                                              |
|              | 4aed7709381c/resources/scale_up_policy?SignatureMethod=HmacS     |
|              | HA256&SignatureVersion=2&AWSAccessKeyId=b808742eff8f4819999b     |
|              | f641cb65243e&Signature=VCCUYkiSz2uW0hCXOS9ptgUtSM9BBSUqaE4qb     |
|              | s6IzR4%3D                                                        |
+--------------+------------------------------------------------------------------+
```

Nota. Elaboración propia.

Cuadro 134. Consulta de URL de escalado hacia abajo

```
openstack stack output show asg-demo scale_down_url
```

Nota. Elaboración propia.

Cuadro 135. Respuesta esperada a consulta de URL de escalado hacia abajo

```
+--------------+------------------------------------------------------------------+
| Field        | Value                                                            |
+--------------+------------------------------------------------------------------+
| output_key   | scale_down_url                                                   |
| description  | URL CFN firmada para pruebas manuales (scale-down).              |
| output_value | http://192.168.74.69:8000/v1/signal/arn%3Aopenstack%3Aheat%3 |
|              | A%3A5260c7767b584c58a1da728373b2d8f3%3Astacks/asg-demo/13ba5 |
|              | 26b-5046-40d2-aac9-                                              |
|              | 4aed7709381c/resources/scale_down_policy?SignatureMethod=Hma |
|              | cSHA256&SignatureVersion=2&AWSAccessKeyId=a7dfbd7bd6894a74b5 |
|              | 16d624f20b5156&Signature=I5T6QcGomG0iEjN%2B1gycixEYYpxxjuk%2 |
|              | BdUWxjCyWUHA%3D                                                  |
+--------------+------------------------------------------------------------------+
```

Nota. Elaboración propia.

Para un mejor manejo de las URL, se pueden guardar en variables de entorno para su uso posterior. Esto facilita la activación de las políticas de escalado mediante comandos curl o desde alarmas de Aodh. Se utiliza el comando **openstack stack output show** para extraer las URL y almacenarlas en las variables de entorno **SCALE_UP_URL** y **SCALE_DOWN_URL**.

Cuadro 136. Guardado URL de escalado en variables de entorno

```
SCALE_UP_URL=$(openstack stack output show asg-demo scale_up_url -f value -c
    ↪ output_value)
echo "SCALE_UP_URL=$SCALE_UP_URL"

SCALE_DOWN_URL=$(openstack stack output show asg-demo scale_down_url -f value -c
    ↪ output_value)
echo "SCALE_DOWN_URL=$SCALE_DOWN_URL"
```

Nota. Elaboración propia.

Escalado manual de instancias

Una vez que se tienen las URL de señalización, se puede proceder a activar manualmente las políticas de escalado para verificar su funcionamiento. Esto se puede hacer utilizando el comando **openstack stack resource signal** o mediante comandos curl para enviar una solicitud HTTP POST a las URL obtenidas anteriormente. Al activar la política de escalado hacia arriba, se debería observar un aumento en el número de instancias en el grupo de autoescalado, mientras que al activar la política de escalado hacia abajo, se debería observar una disminución en el número de instancias.

El primero método de escalado manual utiliza el comando **openstack stack resource signal** para enviar una señal a la política de escalado hacia arriba, a partir de esto Heat ejecutará la política correspondiente para aumentar la capacidad del grupo de autoescalado y tomar todas las acciones

que están definidas en la política. Al inicar la plantilla se crea una instancia inicial en el grupo de autoescalado, se puede verificar el estado del grupo utilizando el comando **openstack server list** para observar la capacidad deseada del grupo antes de realizar cualquier acción de escalado.

Cuadro 137. Verificación de instancias antes de escalado

```
openstack server list
```

Nota. Elaboración propia.

Cuadro 138. Respuesta esperada a verificación de instancias antes de escalado

```
+----------------+----------------+--------+----------------+--------+----------+
| ID             | Name           | Status | Networks       | Image  | Flavor   |
+----------------+----------------+--------+----------------+--------+----------+
| 7bd846f4-7d5b- | autoscale-asg- | ACTIVE | autoscale-net= | cirros | m1.micro |
| 4aa5-a031-     | demo           |        | 192.168.100.56 |        |          |
| 4c2ca20c73a6   |                |        |                |        |          |
+----------------+----------------+--------+----------------+--------+----------+
```

Nota. Elaboración propia.

La respuesta al comando nos muestra que hay una instancia activa en el grupo de autoescalado. Ahora, se puede proceder a activar la política de escalado hacia arriba utilizando el comando **openstack stack resource signal**, al ejecutar este comando, Heat ejecutará la política de escalado definida para aumentar la capacidad del grupo de autoescalado y creará una nueva instancia.

Cuadro 139. Escalado manual hacia arriba utilizando comando de Heat

```
openstack stack resource signal asg-demo scale_up_policy
```

Nota. Elaboración propia.

Al volver a correr el comando **openstack server list**, se debería observar que ahora hay dos instancias activas en el grupo de autoescalado, indicando que la política de escalado hacia arriba ha sido ejecutada correctamente.

Cuadro 140. Verificación de instancias después de escalado hacia arriba

```
openstack server list
```

Nota. Elaboración propia.

Cuadro 141. Respuesta esperada a verificación de instancias después de escalado hacia arriba

```
+----------------+----------------+--------+----------------+--------+----------+
| ID             | Name           | Status | Networks       | Image  | Flavor   |
+----------------+----------------+--------+----------------+--------+----------+
| 62c089bf-b26e- | autoscale-asg- | ACTIVE | autoscale-net= | cirros | m1.micro |
| 4c46-bca8-     | demo           |        | 192.168.100.17 |        |          |
```

```
| 6e3732df524a   |                | ACTIVE | 8              | cirros | m1.micro |
| 7bd846f4-7d5b- | autoscale-asg- |        | autoscale-net= |        |          |
| 4aa5-a031-     | demo           |        | 192.168.100.56 |        |          |
+----------------+----------------+--------+----------------+--------+----------+
```

Nota. Elaboración propia.

De manera similar, para activar la política de escalado hacia abajo, se utiliza el comando **openstack stack resource signal** nuevamente, pero esta vez apuntando a la política de escalado hacia abajo. Al ejecutar este comando, Heat ejecutará la política de escalado definida para disminuir la capacidad del grupo de autoescalado y eliminará una instancia. Se debe de tomar en cuenta el tiempo de *cooldown* definido en la política para evitar que se realicen múltiples acciones de escalado en un corto período de tiempo, ya que si se hace dentro de la misma ventana del *cooldown*, la acción no se ejecutará.

Cuadro 142. Escalado manual hacia abajo utilizando comando de Heat

```
openstack stack resource signal asg-demo scale_down_policy
```

Nota. Elaboración propia.

Al volver a correr el comando **openstack server list**, se debería observar que ahora hay una sola instancia activa en el grupo de autoescalado, indicando que la política de escalado hacia abajo ha sido ejecutada correctamente.

Cuadro 143. Verificación de instancias después de escalado hacia abajo

```
openstack server list
```

Nota. Elaboración propia.

Cuadro 144. Respuesta esperada a verificación de instancias después de escalado hacia abajo

```
+----------------+----------------+--------+----------------+--------+----------+
| ID             | Name           | Status | Networks       | Image  | Flavor   |
+----------------+----------------+--------+----------------+--------+----------+
| 62c089bf-b26e- | autoscale-asg- | ACTIVE | autoscale-net= | cirros | m1.micro |
| 4c46-bca8-     | demo           |        | 192.168.100.17 |        |          |
+----------------+----------------+--------+----------------+--------+----------+
```

Nota. Elaboración propia.

El segundo método de escalado manual utiliza comandos curl para enviar solicitudes HTTP POST directamente a las URL de señalización obtenidas anteriormente. Esto simula la activación de las políticas de escalado como si fueran disparadas por las alarmas de Aodh. Un comando curl es un llamado HTTP que permite interactuar con URL desde la línea de comandos. Este método es más acertado a lo que harían las alarmas de Aodh al activarse, ya que las alarmas envían solicitudes HTTP POST a las URL de señalización para activar las políticas de escalado.

Estos comandos curl envían una solicitud POST a las URL de escalado hacia arriba y hacia abajo, respectivamente. Al ejecutar estos comandos, se debería observar el mismo comportamiento que con el método anterior, donde la capacidad del grupo de autoescalado aumenta o disminuye según la política activada. La diferencia entre ambos métodos radica en la forma en que se envía la señal para activar las políticas de escalado, pero el resultado final es el mismo. Los comandos para realizar el escalado manual utilizando curl son los siguientes.

Cuadro 145. Comandos de escalado manual utilizando curl

```
curl -s -X POST "$SCALE_UP_URL"
curl -s -X POST "$SCALE_DOWN_URL"
```

Nota. Elaboración propia.

8.3.6. Escalado automático de instancias

Para verificar el funcionamiento del escalado automático basado en métricas se debe de tener en cuenta tres puntos de carga de CPU para ajustar los umbrales tanto de subida como de bajada y hasta los valores de capacidad mínima y máxima del grupo de autoescalado así como el *cooldown* de las políticas de escalado. Estos tres puntos clave son los siguientes:

- **Punto de carga baja:** este punto se define por debajo del umbral de CPU bajo (cpu_low_threshold) establecido en la plantilla. Cuando la carga de CPU de las instancias en el grupo de autoescalado cae por debajo de este umbral durante el período de evaluación definido, se activará la política de escalado hacia abajo para reducir la capacidad del grupo.
- **Punto de carga alta:** este punto se define por encima del umbral de CPU alto (cpu_high_threshold) establecido en la plantilla. Cuando la carga de CPU de las instancias en el grupo de autoescalado supera este umbral durante el período de evaluación definido, se activará la política de escalado hacia arriba para aumentar la capacidad del grupo.
- **Punto de carga media:** este punto se encuentra entre los umbrales de CPU bajo y alto. Cuando la carga de CPU se encuentra en este rango, no se activarán las políticas de escalado, y el grupo mantendrá su capacidad actual.

En este caso, se colocará una carga simulada mediante CLI, mediante un comando muy sencillo que genera una carga de CPU en la instancia. Este comando se ejecuta dentro de la instancia y utiliza un bucle infinito para consumir recursos de CPU. El comando utilizado es el siguiente:

Cuadro 146. Comando para generar carga de CPU en la instancia

```
sh -c 'while :; do :; done' &
echo $! > /tmp/cpu.pid
```

Nota. Elaboración propia.

Este comando inicia un bucle infinito que consume CPU y guarda el ID del proceso en un archivo temporal para poder detenerlo más tarde mediante el comando **kill** seguido del ID del proceso. Para detener la carga de CPU generada, se utiliza el siguiente comando:

Cuadro 147. Comando para detener la carga de CPU en la instancia

```
kill $(cat /tmp/cpu.pid)
```

Nota. Elaboración propia.

Es importante monitorear las métricas de CPU de las instancias en el grupo de autoescalado para observar cómo varían en respuesta a la carga generada. Esto se puede hacer utilizando el comando **openstack metric resource show** para obtener las métricas de CPU de una instancia específica. Al generar carga de CPU, se debería observar un aumento en la métrica; cuando la carga se detiene, la métrica debería disminuir nuevamente. Monitorear estas métricas es crucial para verificar que las alarmas de Aodh se disparen correctamente y que las políticas de escalado se ejecuten según lo esperado en función de la carga de CPU o bien modificar los umbrales de las alarmas para ajustarlos a la carga generada.

Para esto, se puede utilizar el siguiente *script* que consulta las métricas de CPU de todas las instancias en el grupo de autoescalado, calcula el promedio de la métrica y determina si se debe escalar hacia arriba, abajo, o mantener la capacidad actual del grupo en función de los umbrales definidos. Es importante que las variables vayan acorde a los valores definidos en la plantilla de orquestación.

Cuadro 148. Script para monitorear métricas de CPU y determinar acciones de escalado

```
STACK=asg-demo
GRAN=300
LOW_TH=3070000000
HIGH_TH=10000000000

RESIDS=$(for id in $(openstack server list --name "autoscale-$STACK" -f value -c ID);
    ↪ do
  if gnocchi resource show -t instance "$id" >/dev/null 2>&1; then
    echo "$id"
  fi
done)
echo "RESIDS:"; echo "$RESIDS" | sed 's/^/ - /'

for rid in $RESIDS; do
  echo "===== resource_id $rid ====="
  gnocchi measures show cpu --resource-id "$rid" --aggregation 'rate:mean'
    ↪ --granularity "$GRAN" \
    -f value -c timestamp -c value | tail -n 12
done

AVG=$(for rid in $RESIDS; do
```

```
  gnocchi measures show cpu --resource-id "$rid" --aggregation 'rate:mean'
    ↪ --granularity "$GRAN" \
    -f value -c value | tail -n 1
done | awk '{s+=$1;n++} END{if(n>0) printf("%.0f", s/n); else print "NaN"}')

echo "avg_nsps_last=${AVG}"

awk -v v="$AVG" -v low="$LOW_TH" -v high="$HIGH_TH" 'BEGIN{
  printf("avg_nsps_last=%s  LOW=%d  HIGH=%d  => ", v, low, high);
  if (v=="NaN") print "SIN_DATOS";
  else if (v+0 > high) print "UP";
  else if (v+0 < low)  print "DOWN";
  else                 print "OK";
}'

for rid in $RESIDS; do
  ts=$(gnocchi measures show cpu --resource-id "$rid" --aggregation 'rate:mean'
    ↪ --granularity "$GRAN" \
        -f value -c timestamp | tail -n 1)
  printf "%s  %s\n" "$rid" "$ts"
done
```

Nota. Elaboración propia.

Este *script* realiza las siguientes acciones:

- Define las variables necesarias, incluyendo el nombre del stack, la granularidad de las métricas y los umbrales de CPU bajo y alto.
- Obtiene los ID de las instancias en el grupo de autoescalado utilizando el comando **openstack server list** y verifica que cada instancia tenga métricas disponibles en Gnocchi.
- Para cada instancia, consulta las métricas de CPU utilizando el comando **gnocchi measures show** y muestra las últimas 12 mediciones.
- Calcula el promedio de la métrica de CPU para todas las instancias en el grupo.
- Compara el promedio calculado con los umbrales definidos y determina si se debe escalar hacia arriba, hacia abajo o mantener la capacidad actual del grupo.
- Muestra la última marca de tiempo de las métricas para cada instancia.

Al ejecutar este *script*, se podrá monitorear las métricas de CPU de las instancias en el grupo de autoescalado y verificar si las políticas de escalado se están activando correctamente en función de la carga generada. Es importante ajustar los umbrales y la carga generada según sea necesario para probar diferentes escenarios de escalado automático.

Para este caso los umbrales definidos son 3.07 GHz para el umbral bajo y 10 GHz para el umbral alto, con una capacidad mínima de 1 instancia y una capacidad máxima de 5 instancias en el grupo

de autoescalado. El período de evaluación está definido en 300 segundos (5 minutos) y el tiempo de *cooldown* para las políticas de escalado es de 180 segundos para la subida de instancias y 240 segundos para la bajada.

La respuesta que se obtuvo al ejecutar el *script* en este punto donde se tiene una instancia activa y sin carga de CPU es la siguiente:

Cuadro 149. Respuesta esperada al ejecutar el script sin carga de CPU

```
 - 62c089bf-b26e-4c46-bca8-6e3732df524a
===== resource_id 62c089bf-b26e-4c46-bca8-6e3732df524a =====
2025-10-29T13:15:00-06:00 990000000.0
2025-10-29T13:20:00-06:00 980000000.0
2025-10-29T13:25:00-06:00 990000000.0
2025-10-29T13:30:00-06:00 1010000000.0
2025-10-29T13:35:00-06:00 970000000.0
2025-10-29T13:40:00-06:00 980000000.0
2025-10-29T13:45:00-06:00 990000000.0
2025-10-29T13:50:00-06:00 1030000000.0
2025-10-29T13:55:00-06:00 990000000.0
2025-10-29T14:00:00-06:00 980000000.0
2025-10-29T14:05:00-06:00 990000000.0
2025-10-29T14:10:00-06:00 1040000000.0
avg_nsps_last=1040000000
avg_nsps_last=1040000000  LOW=3070000000  HIGH=10000000000  => DOWN
62c089bf-b26e-4c46-bca8-6e3732df524a  2025-10-29T14:10:00-06:00
```

Nota. Elaboración propia.

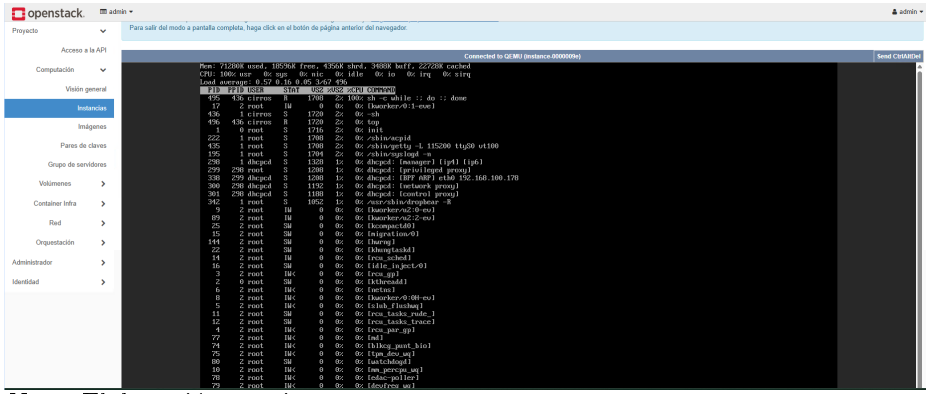
La respuesta muestra que la métrica de CPU promedio para la instancia es de 1040000000 (1.04 GHz), lo cual está por debajo del umbral bajo definido (3070000000). Por lo tanto, el *script* indica que se debe escalar hacia abajo (DOWN), que en este caso no se puede ejecutar debido a que la capacidad mínima del grupo es de una instancia.

Ahora para fines demostrativos aplicaremos la carga de CPU en la instancia para observar cómo varían las métricas y verificar si se activa la política de escalado hacia arriba. Después de aplicar la carga de CPU, se vuelve a ejecutar el *script* para monitorear las métricas, tomar en cuenta la granularidad y el tiempo de evaluación definidos para la toma de métricas.

El comando **top** dentro de la instancia muestra el uso de CPU en tiempo real, y se puede observar que la carga de CPU ha aumentado significativamente debido al comando que genera la carga. La métrica de CPU promedio debería reflejar este aumento y superar el umbral alto definido (10000000000) para activar la política de escalado hacia arriba.

En la imagen podemos observar que el PID 495, que fue el asignado al comando que genera la carga de CPU, está utilizando el 100 % de la CPU, lo que indica que la carga se ha aplicado correctamente. Al volver a ejecutar el *script* para monitorear las métricas de CPU, se obtiene la siguiente respuesta:

Figura 13. Uso de CPU durante la prueba de carga.



Nota. Elaboración propia.

Cuadro 150. Respuesta esperada al ejecutar el script con carga de CPU

```
RESIDS:
 - 6e0bc64d-2b6b-4dd6-8732-1d1fbf559bf0
 - 62c089bf-b26e-4c46-bca8-6e3732df524a
===== resource_id 6e0bc64d-2b6b-4dd6-8732-1d1fbf559bf0 =====
===== resource_id 62c089bf-b26e-4c46-bca8-6e3732df524a =====
2025-10-29T13:30:00-06:00 1010000000.0
2025-10-29T13:35:00-06:00 970000000.0
2025-10-29T13:40:00-06:00 980000000.0
2025-10-29T13:45:00-06:00 990000000.0
2025-10-29T13:50:00-06:00 1030000000.0
2025-10-29T13:55:00-06:00 990000000.0
2025-10-29T14:00:00-06:00 980000000.0
2025-10-29T14:05:00-06:00 990000000.0
2025-10-29T14:10:00-06:00 1040000000.0
2025-10-29T14:15:00-06:00 980000000.0
2025-10-29T14:20:00-06:00 61270000000.0
2025-10-29T14:25:00-06:00 300630000000.0
avg_nsps_last=300630000000
avg_nsps_last=300630000000  LOW=3070000000  HIGH=10000000000  => UP
```

Nota. Elaboración propia.

La respuesta muestra que ahora hay dos instancias en el grupo de autoescalado, y la métrica de CPU promedio ha aumentado significativamente a 300630000000 (300.63 GHz), lo cual está por encima del umbral alto definido (10000000000). Por lo tanto, el *script* indica que se debe escalar hacia arriba (UP), y Heat debería haber ejecutado la política de escalado para crear una nueva instancia en el grupo de autoescalado. La instancia creada aún no cuenta con métricas de CPU debido a que es muy reciente, pero en las siguientes ejecuciones del *script* se podrán observar sus métricas.

Después de verificar que la política de escalado hacia arriba se ha activado correctamente, se puede proceder a detener la carga de CPU generada en la instancia utilizando el comando **kill** mencionado anteriormente. Una vez que la carga se detiene, se vuelve a ejecutar el *script* para monitorear las métricas de CPU y verificar si la política de escalado hacia abajo se activa correctamente cuando la métrica cae por debajo del umbral bajo definido.

Se realizará un ciclo de escalado completo, es decir, un escalado de hasta cinco instancias y luego un escalado de regreso a una sola instancia, para verificar que ambas políticas de escalado funcionen correctamente en diferentes escenarios de carga. Es importante monitorear las métricas de CPU y ajustar los umbrales según sea necesario para probar diferentes condiciones de escalado automático.

En la demostración que se presenta a continuación se realizó un ciclo completo de cinco instancias con un *threshold* de bajada de 15 miles de millones de ns/s y uno de subida de 28 miles de millones de ns/s, colocandoles una carga artificial de CPU al momento de ser creadas, simulando un período de alta demanda y manteniendo la carga por un tiempo antes de liberarla por completo y observar el escalado gradual hacía abajo de regreso a una sola instancia, completando un ciclo completo de escalado automático basado en métricas de CPU.

Cuadro 151. Serie temporal de uso de CPU por ID y promedio, expresado en miles de millones de ns/s, con conteo de instancias activas.

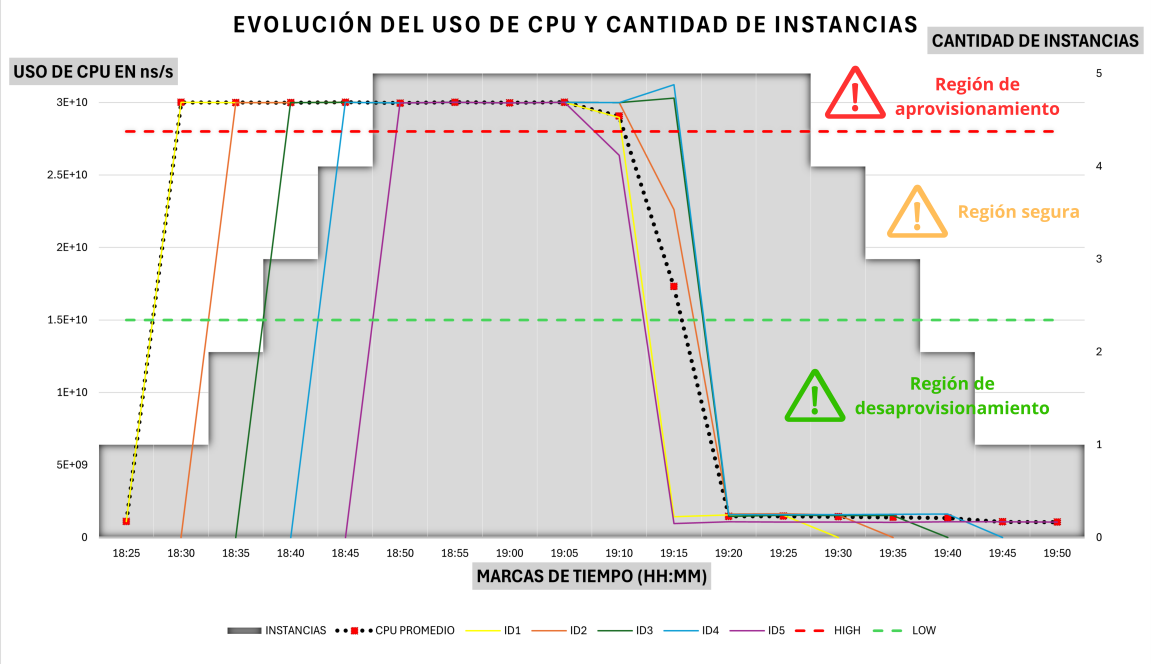
TIMESLOT	ID1	ID2	ID3	ID4	ID5	CPU PROMEDIO	INSTANCIAS
18:25	1.110	–	–	–	–	1.110	1
18:30	30.007	–	–	–	–	30.007	1
18:35	29.998	29.991	–	–	–	29.994	2
18:40	29.984	29.981	29.988	–	–	29.984	3
18:45	30.012	30.011	30.013	30.013	–	30.012	4
18:50	29.961	29.961	29.961	29.961	29.961	29.961	5
18:55	30.016	30.013	30.013	30.013	30.015	30.014	5
19:00	29.974	29.976	29.972	29.977	29.976	29.975	5
19:05	30.013	30.010	30.013	30.015	30.011	30.012	5
19:10	28.956	29.991	29.998	29.998	26.357	29.060	5
19:15	1.440	22.620	30.300	31.230	0.970	17.312	5
19:20	1.560	1.620	1.460	1.580	1.080	1.460	5
19:25	1.540	1.650	1.530	1.570	1.060	1.470	5
19:30	–	1.560	1.510	1.580	1.070	1.430	4
19:35	–	–	1.520	1.600	1.050	1.390	3
19:40	–	–	–	1.620	1.080	1.350	2
19:45	–	–	–	–	1.080	1.080	1
19:50	–	–	–	–	1.060	1.060	1

Nota. Elaboración propia.

En el cuadro 151 se muestra la serie temporal del uso de CPU por ID de instancia y el promedio, expresado en miles de millones de ns/s, junto con el conteo de instancias activas en cada período de tiempo. Se puede observar cómo la métrica de CPU varía en función de la carga aplicada y cómo

el número de instancias cambia en respuesta a las políticas de escalado definidas en la plantilla de orquestación. Para una mejor comprensión, se creó un gráfico que ilustra estos cambios de manera visual.

Figura 14. Gráfico de la serie temporal de uso de CPU por ID y promedio con conteo de instancias activas.



Nota. Elaboración propia.

La Figura 14 ilustra el ciclo demostrativo realizado, la linea punteada representa el promedio de uso de CPU, mientras que las barras indican el número de instancias activas en cada período de tiempo. Las líneas punteadas horizontales representan los umbrales de CPU bajo y alto definidos en la plantilla de orquestación, las líneas contínuas representan el uso de CPU por ID de instancia. Analizando cada marca de tiempo se puede observar lo siguiente:

- **18:25:** se tiene la instancia inicial del grupo, la métrica de CPU está por debajo del umbral bajo, por lo que no se realiza ninguna acción de escalado y se mantiene una sola instancia activa.
- **18:30:** se coloca la carga simulada en la instancia creada, la métrica de CPU supera el umbral alto, lo que activa la política de escalado hacia arriba. Se tiene en proceso de creación una nueva instancia y antes de la siguiente marca de tiempo se le coloca su carga artificial.
- **18:35** se tiene la segunda instancia activa en conjunto a la primera, la métrica de CPU promedio continúa superando el umbral alto debido a las cargas colocadas, lo que provoca la creación de más instancias en el grupo de autoescalado. Se coloca en estado de creación la tercera instancia que ya aparecerá activa para la siguiente marca de tiempo.

- **18:40:** se tiene la tercera instancia activa junto a las dos anteriores, la métrica de CPU promedio sigue superando el umbral alto, lo que lleva a la creación de una cuarta instancia. Se coloca en estado de creación la cuarta instancia que ya aparecerá activa para la siguiente marca de tiempo.
- **18:45:** se tiene la cuarta instancia activa junto a las tres anteriores, la métrica de CPU promedio continúa superando el umbral alto debido a las cargas continuas agregadas a cada instancia conforme su expansión, lo que provoca la creación de una quinta instancia. Se coloca en estado de creación la quinta instancia que ya aparecerá activa para la siguiente marca de tiempo.
- **18:50:** se tiene la quinta instancia activa junto a las cuatro anteriores, la métrica de CPU promedio sigue superando el umbral alto, alcanzando el máximo de instancias permitidas en el grupo de autoescalado (5 instancias). No se pueden crear más instancias debido a la capacidad máxima definida.
- **18:55 a 19:05:** durante este período las cargas artificiales aún no han sido retiradas y la métrica de CPU promedio se mantiene por encima del umbral alto, pero no se pueden crear más instancias debido a la capacidad máxima alcanzada.
- **19:10:** se comienza a retirar la carga de CPU de la instancia ID5, la métrica de CPU promedio disminuye pero aún supera el umbral alto, lo que indica que el grupo sigue en un estado de alta demanda por lo que es necesario retirar la carga del resto de instancias y esperar la bajada de uso de CPU.
- **19:15:** se retira la carga de CPU de las instancias ID1, ID2 y ID5, la métrica de CPU promedio cae por debajo del umbral alto, pero aún está por encima del umbral bajo, lo que indica que el grupo está en un estado de demanda moderada. No se realizan acciones de escalado en este punto ya que se encuentra dentro de la zona segura.
- **19:20 a 19:25:** se retira la carga de CPU de todas las instancias, la métrica de CPU promedio cae por debajo del umbral bajo, lo que activa la política de escalado hacia abajo. Se comienza a eliminar instancias del grupo de autoescalado luego del segundo período de evaluación ya que dentro de la plantilla se definió un período de evaluación de 5 minutos (300 segundos) para las alarmas.
- **19:30:** se elimina la instancia ID1, quedando cuatro instancias activas en el grupo. La métrica de CPU promedio continúa por debajo del umbral bajo, lo que lleva a la eliminación de otra instancia en el siguiente período de evaluación.
- **19:35:** se elimina la instancia ID2, quedando tres instancias activas en el grupo. La métrica de CPU promedio sigue por debajo del umbral bajo, lo que provoca la eliminación de otra instancia en el siguiente período de evaluación.
- **19:40:** se elimina la instancia ID3, quedando dos instancias activas en el grupo. La métrica de CPU promedio continúa por debajo del umbral bajo, lo que lleva a la eliminación de la última instancia en el siguiente período de evaluación.
- **19:45:** se elimina la instancia ID4, quedando una sola instancia activa en el grupo. La métrica de CPU promedio sigue por debajo del umbral bajo, pero no se pueden eliminar más instancias debido a la capacidad mínima definida (1 instancia).

- **19:50:** se mantiene una sola instancia activa en el grupo, la métrica de CPU promedio continúa por debajo del umbral bajo, pero no se realizan acciones de escalado ya que se ha alcanzado la capacidad mínima del grupo.

CAPÍTULO 9

Conclusiones

- Se configuró y estableció el servicio Glance para conversión de imágenes a QCOW2 mediante un proceso estandarizado usando la herramienta de `QEMU-img`. Se definieron los pasos necesarios para la conversión de imágenes, asegurando la compatibilidad y optimización del formato QCOW2 para su uso en entornos de virtualización. Este formato es óptimo para OpenStack/KVM debido a sus características de eficiencia en el almacenamiento y soporte para *snapshots*.
- Se demostró la implementación exitosa de imágenes para una amplia gama de sistemas, incluyendo distribuciones de Linux tanto servidor como escritorio, así como sistemas operativos Windows servidor y profesional. La versatilidad del servicio Glance permitió la gestión centralizada de estas imágenes, facilitando su despliegue y administración en entornos virtualizados.
- Se brindó un catálogo de imágenes y *flavors* predefinidos para la creación rápida de instancias en OpenStack. Este catálogo incluye configuraciones optimizadas para diferentes casos de uso, permitiendo a los usuarios seleccionar rápidamente las opciones que mejor se adapten a sus necesidades sin tener que configurar manualmente cada instancia desde cero.
- Se introdujo el paradigma de computación elástica a través de servicios de orquestación, telemetría y alarmas, creando un circuito de retroalimentación autónomo denominado *closed-loop elasticity* haciendo uso de los servicios de Heat, Ceilometer y Aodh. Este enfoque permite la adaptación dinámica de los recursos computacionales en función de la demanda, lo que mejora la eficiencia operativa y reduce costos asociados al uso de infraestructura.
- Se implementó el servicio Heat para definir infraestructuras complejas como código mediante plantillas YAML denominadas HOT *Heat Orchestration Templates*. Estas plantillas permiten la automatización del despliegue y gestión de recursos en OpenStack, lo que facilita la replicación y la escalabilidad de entornos completos con configuraciones específicas.

- Se configuraron los servicios de Gnocchi para almacenamiento de series temporales y Ceilometer para la recolección de métricas de uso y rendimiento de los recursos en la nube. Estos servicios proporcionan una visión detallada del comportamiento del sistema, permitiendo la monitorización efectiva y la toma de decisiones informadas sobre la gestión de recursos.
- Se estableció el servicio Aodh para la gestión de alarmas basadas en las métricas recolectadas por Ceilometer y almacenadas en Gnocchi. Este servicio permite la configuración de alertas automáticas que pueden desencadenar acciones específicas, como el escalado de recursos o notificaciones, mejorando la capacidad de respuesta ante cambios en la demanda o condiciones operativas.
- Se diseñó y validó una política de escalado horizontal basada en la métrica de uso de CPU donde se realizaron pruebas de carga artificial lo que permitió observar el comportamiento del sistema ante incrementos y decrementos en la demanda. Esta política asegura que los recursos se ajusten dinámicamente, manteniendo un rendimiento óptimo y evitando tanto la sobrecarga como el desperdicio de capacidad.
- El proyecto dota al Departamento de Electrónica de una nube privada con alta disponibilidad y resilencia. La versatilidad y escalado automático implementado permite simular un entorno de nube moderno que puede ser utilizado para fines educativos, de investigación y desarrollo, así como para la experimentación con nuevas tecnologías y servicios en la nube.

CAPÍTULO 10

Recomendaciones

- Implementar un *dashboard* de monitoreo y visualización centralizado. Para mejorar la observabilidad del proceso de escalado automático, se recomienda instalar y configurar Grafana junto con Gnocchi y Ceilometer para consumir y visualizar las series temporales almacenadas en Gnocchi. Esto permitirá crear paneles personalizados e interactivos que muestren métricas clave del rendimiento del sistema, lo que facilitará la identificación de tendencias y posibles cuellos de botella. El *dashboard* podría incluir gráficos de uso de CPU, memoria, red, un contador de instancias activas y alertas visuales para eventos críticos con notificaciones por correo.
- Desarrollar un sistema de reportes automatizados que genere informes periódicos sobre el rendimiento y la eficiencia del escalado automático. Estos reportes podrían incluir análisis de tendencias, recomendaciones para optimización de recursos y resúmenes de eventos de escalado ocurridos durante el período. La automatización del proceso de generación y distribución de estos informes ayudará a mantener informados a los administradores del sistema y facilitará la toma de decisiones basadas en datos históricos.
- Complementar la escalabilidad horizontal añadiendo capacidades de escalabilidad vertical, lo que ofrecerá aún más flexibilidad, sobre todo para cargas de trabajo que no se benefician del escalado horizontal. Desarrollar procedimientos y scripts que utilicen la API de Nova para redimensionar los *flavors* de una instancia activa en función de métricas de uso de CPU, memoria y red. Crear políticas que definan umbrales para el escalado vertical, permitiendo aumentar o disminuir los recursos asignados a una instancia sin necesidad de reiniciarla o crear una nueva.

- Integrar el sistema de almacenamiento distribuido Ceph. Esto aportaría un backend de almacenamiento de bloques y objetos altamente disponible. Esto es fundamental para asegurar la persistencia de datos en entornos de autoescalado donde las instancias pueden ser creadas y destruidas dinámicamente.
- Implementar un sistema de balanceo de carga avanzado utilizando Octavia o HAProxy para distribuir el tráfico de red entre las instancias de manera eficiente. Esto garantizará que las nuevas instancias creadas durante el proceso de escalado automático sean incorporadas al balanceo de carga, lo que mejorará la disponibilidad y el rendimiento del servicio.
- Desarrollar un sistema de pruebas automatizadas para validar las políticas de escalado y el comportamiento del sistema bajo diferentes cargas de trabajo. Esto incluiría la creación de scripts que simulen cargas de trabajo variables y midan el tiempo de respuesta del sistema, la eficiencia del escalado y la estabilidad general del entorno.

CAPÍTULO 11

Referencias

[1] F. J. L. Turcios, *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*, 2024.

[2] T. Bell, *10 Years of OpenStack at CERN: From 0 to 300k Cores*, https://www.openstack.org/videos/summits/virtual/10-years-of-OpenStack-at-CERN-From-0-to-300k-cores, OpenInfra Summit Virtual, OpenInfra Foundation, 2020.

[3] F. D. C. Aragón, «Módulo para la gestión de imágenes de Sistema Operativo en Nova-LTSP 2.0,» Tesis de mtría., Universidad de las Ciencias Informáticas, La Habana, Cuba, 2020.

[4] F. A. Hashmi, «OpenStack-Elastic Cloud Service Orchestration with Openstack,» Supervised by Muhammad Durrani, Pakistan, inf. téc., 2017.

[5] N. Trippler, «Using Heat and Ceilometer to Create an Elastic OpenStack Grid,» Tesis de mtría., University of Bergen, Bergen, Norway, 2017.

[6] M. Armbrust et al., *A View of Cloud Computing*, https://dl.acm.org/doi/pdf/10.1145/1721654.1721672, Communications of the ACM, Vol. 53, No. 4, 2010.

[7] P. Mell y T. Grance, *The NIST Definition of Cloud Computing*, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf, NIST Special Publication 800-145, 2011.

[8] P. Barham et al., *Xen and the Art of Virtualization*, https://dl.acm.org/doi/pdf/10.1145/361011.361073, 2003.

[9] Red Hat, *KVM Architecture Overview*, https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/7/html/virtualization_tuning_and_optimization_guide/sect-virtualization_tuning_optimization_guide-introduction-kvm_architecture_overview, 2015.

[10] Linux Foundation, *Linux Distributions*, https://www.linux.org/learn/linux-distributions, 2025.

[11] T. Mirzoev, *Employing Virtualization for Information Technology Education*, https://arxiv.org/abs/1404.2167, arXiv preprint arXiv:1404.2167, 2014.

[12] OpenStack Foundation, *OpenStack Documentation 2025.1*, https://docs.openstack.org/2025.1/, 2025.

[13] OpenStack Foundation, *OpenStack Project Navigator*, https://www.openstack.org/software/project-navigator/, https://www.openstack.org/software/project-navigator/, 2024.

[14] ResearchGate, *OpenStacks main components and their interaction*, https://www.researchgate.net/figure/OpenStacks-main-components-and-their-interaction_fig1_333943258, Figura consultada en línea, 2019.

[15] OpenStack Foundation, *OpenStack Heat Documentation*, https://docs.openstack.org/heat/latest/, 2025.

[16] OpenStack Foundation, *OpenStack Ceilometer Documentation*, https://docs.openstack.org/ceilometer/latest/, 2025.

[17] OpenStack Foundation, *OpenStack Gnocchi Documentation*, https://docs.openstack.org/python-openstackclient/latest/cli/plugin-commands/gnocchi.html, 2025.

[18] OpenStack Foundation, *OpenStack Operations Guide*, https://docs.openstack.org/operations-guide/, 2014.

[19] OpenStack Foundation, *OpenStack Command-Line Client (OSC) Documentation*, https://docs.openstack.org/python-openstackclient/latest/, Accedido el 7 de junio de 2025, 2025.

[20] G. (Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure.Com LLC, 2009, ISBN: 9780979958717.

[21] P. Mell y T. Grance, «The NIST Definition of Cloud Computing,» National Institute of Standards y Technology, inf. téc. SP 800-145, 2011, [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf.

[22] Amazon Web Services, *What is Elastic Computing?* [Online]. Available: https://aws.amazon.com/elastic-compute-cloud/, 2024.

[23] Amazon Web Services, *Elastic Load Balancing and Geographic Elasticity*, [Online]. Available: https://aws.amazon.com/elasticloadbalancing/, 2024.

[24] R. Buyya, C. Vecchiola y S. T. Selvi, *Mastering Cloud Computing: Foundations and Applications Programming*. Morgan Kaufmann, 2013.

[25] N. Trippler, «Using Heat and Ceilometer to Create an Elastic OpenStack Grid,» Department of Informatics, Master's thesis, University of Bergen, Bergen University College, jun. de 2017. dirección: https://hdl.handle.net/1956/16256.

[26] B. Bhatta, *Auto Scaling in OpenStack using Heat, Gnocchi and Aodh*, https://bhujaykbhatta.wordpress.com/2018/01/18/auto-scaling-in-openstack-using-heat-gnocchi-and-aodh/, Publicado en el blog de Bhujay Bhatta, 2018.

[27] A. Barnawi, S. Sakr, W. Xiao y A. Al-Barakati, «The views, measurements and challenges of elasticity in the cloud: A review,» *Computer Communications*, vol. 154, págs. 111-117, 2020. DOI: 10.1016/j.comcom.2020.02.010.

[28] E. A. C. García, *Automatización de despliegue de plataforma de Cloud Computing Openstack: Una estrategia escalable empleando Infraestructura como código*, 2025.

[29] Canonical Ltd., *Ubuntu Downloads*, https://ubuntu.com/download, Canonical - Official Ubuntu Downloads Page, 2025.

[30] Fedora Project, *Fedora Linux Downloads*, https://www.fedoraproject.org/server/download, Official Fedora Linux Downloads Page, 2025.

[31] Manjaro, *Manjaro Downloads*, https://manjaro.org/download/, Official Manjaro Downloads Page, 2025.

[32] Microsoft Corporation, *Descargar Windows Server 2022 – Centro de evaluación*, https://www.microsoft.com/es-es/evalcenter/download-windows-server-2022, 2025.

[33] Microsoft Corporation, *Descargar Windows 11*, https://www.microsoft.com/es-es/software-download/windows11, 2025.

CAPÍTULO 12

Glosario

Alarma: evento generado cuando se cumple una condición sobre métricas monitorizadas.

Alta disponibilidad: diseño y operación de sistemas para minimizar el tiempo fuera de servicio mediante redundancia, tolerancia a fallos y recuperación rápida.

Auto scaling group: conjunto lógico de instancias gestionadas como unidad para escalar horizontalmente según políticas y métricas.

Backoff exponencial: estrategia de reintentos que aumenta progresivamente el tiempo de espera entre intentos para aliviar congestión o fallas.

Balanceo de carga: distribución del tráfico entre réplicas usando estrategias como *round robin*, menor número de conexiones o hash.

Cardinalidad de métricas: número de combinaciones únicas de etiquetas en una serie temporal; una cardinalidad alta complica almacenamiento y consultas.

cgroups: mecanismo del kernel de Linux que aplica límites y contabilidad de recursos a grupos de procesos.

***Closed-loop elasticity*:** esquema de retroalimentación donde monitoreo, decisión y acción se encadenan para ajustar recursos de manera autónoma ante cambios de carga.

Contención de recursos: competencia entre procesos por CPU, memoria, disco o red que degrada el rendimiento observable.

Periodo de enfriamiento (*cooldown*): intervalo tras una acción automática durante el cual no se permiten nuevas acciones para estabilizar el sistema.

***Dashboard*:** panel visual que agrupa métricas, gráficos y alertas para observar el estado y rendimiento de sistemas.

Elasticidad geográfica: habilidad de distribuir o mover cargas entre regiones para reducir latencia y aumentar la disponibilidad ante fallas locales.

Elasticidad horizontal: aumento o reducción del número de instancias que componen un servicio para repartir la carga y mantener el rendimiento.

Elasticidad temporal: programación del ajuste de capacidad según patrones previsibles de demanda (diurna/semanal) para optimizar costos.

Elasticidad vertical: ajuste de recursos dentro de una misma instancia (CPU, RAM, disco) para absorber picos sin crear nuevas máquinas.

Endpoint HTTP: punto de acceso (URL) expuesto por un servicio para publicar o recolectar datos mediante peticiones HTTP.

Escalabilidad: capacidad potencial de un sistema de crecer en recursos o rendimiento; se diferencia de la elasticidad en que no implica ajuste automático.

Escalado predictivo: ajuste proactivo de capacidad usando pronósticos de demanda, evitando saturación y sobrecostos por reacción tardía.

Escalado reactivo: ajuste de capacidad disparado por métricas que ya superaron umbrales; simple de configurar, pero responde con retardo.

Flavor: plantilla de recursos (vCPU, RAM, disco y metadatos) que define los recursos asignados a una instancia.

Heat Orchestration Template (HOT): lenguaje declarativo de Heat para describir recursos, dependencias y políticas de escalamiento.

Hipervisor: capa de virtualización que abstrae hardware físico para ejecutar múltiples máquinas virtuales aisladas.

Histéresis: margen entre umbrales de subida y bajada que evita oscilaciones de escalado cuando la métrica está cerca del límite.

Imagen base: archivo maestro inmutable a partir del cual se aprovisionan nuevas instancias.

Latencia: tiempo que tarda un paquete o una operación en ir desde el origen al destino; afecta la experiencia en aplicaciones sensibles al tiempo.

Limitación de tasa (throttling): mecanismo que restringe solicitudes a una API en un intervalo para proteger recursos.

Métrica: medida cuantitativa de un aspecto del sistema (ej., uso de CPU, latencia) utilizada para monitoreo, análisis y toma de decisiones.

Microservicios: estilo arquitectónico que descompone una aplicación en servicios pequeños, autónomos y desplegables de forma independiente, que colaboran mediante interfaces ligeras.

Métricas: valores cuantificables recolectados periódicamente que describen utilización y desempeño.

Network Address Translation (NAT): traducción de direcciones IP privadas a públicas para conectividad externa controlada.

Orquestación: coordinación automatizada de despliegue, configuración y relaciones entre recursos de infraestructura.

Patrones de carga: formas repetitivas de demanda (picos de horario laboral, estacionalidad) que guían la planificación de capacidad.

Process Identifier (PID): identificador único asignado a cada proceso en ejecución en un sistema operativo, utilizado para gestionar y controlar procesos.

Plano de control: conjunto de procesos que deciden cómo se gestionan y configuran los recursos del sistema, separado del plano de datos que transporta la información.

Calidad de servicio (QoS): conjunto de técnicas para priorizar tráfico, reservar ancho de banda y controlar colas, garantizando niveles de servicio.

Saturación: estado en que un recurso alcanza o supera su capacidad útil, degradando el desempeño y motivando acciones de mitigación.

Series temporales: modelo de datos en el que las observaciones se almacenan como pares (marca de tiempo, valor), permitiendo consultas y análisis a lo largo del tiempo.

Snapshot: captura en un punto en el tiempo del estado de un disco o instancia para revertir o clonar.

Sobreaprovisionamiento: asignación de más capacidad de la necesaria para reducir riesgo de saturación a costa de eficiencia y costo.

Stack: conjunto de recursos desplegado a partir de una plantilla y gestionado de forma unitaria por Heat.

Subaprovisionamiento: capacidad inferior a la demanda que provoca cuellos de botella, latencia elevada y errores por saturación.

Telemetría: recolección continua de métricas y eventos operativos para análisis y automatización.

Marca de tiempo (timestamp): registro temporal asociado a una medición o evento que indica cuándo ocurrió en formato estándar.

Token (autenticación): credencial temporal emitida por Keystone que autoriza solicitudes a servicios.

Tormenta de alertas: explosión simultánea de notificaciones, usualmente por dependencia común; requiere deduplicación y correlación de eventos.

Umbral: valor límite a partir del cual se dispara una alerta o acción automática, por ejemplo, escalar recursos ante carga elevada.

Ventana de evaluación: intervalo de tiempo sobre el cual se calcula una métrica o condición antes de disparar una alerta o acción.

Ventana rodante: ventana temporal que se desplaza en el tiempo para calcular métricas agregadas y suavizar la variabilidad instantánea.

Volumen efímero: almacenamiento temporal ligado al ciclo de vida de la instancia.

Volumen persistente: dispositivo de bloque gestionado por Cinder que persiste tras eliminar la instancia.