



Manual de usuario: Página web en OpenStack

Moisés Pinto Rodriguez
Kevin Carrascal Causil
Camilo Aparicio Arciria
Brandon Hernández

ING. JORGE GOMEZ GOMEZ

UNIVERSIDAD DE CÓRDOBA
FACULTAD DE INGENIERÍAS
INGENIERÍA DE SISTEMAS Y TELECOMUNICACIONES
MONTERÍA, CÓRDOBA
2025

Contenido

Capítulo 1. Introducción General	4
1.1 ¿Qué es el cómputo en la nube?.....	4
1.2 Modelos de servicio (IaaS, PaaS, SaaS)	4
1.3 Modelos de despliegue (Privada, Pública, Híbrida, Comunitaria).....	5
1.4 ¿Dónde encaja OpenStack dentro del ecosistema cloud?	5
1.5 Ventajas del uso de la nube para servidores web.....	6
Capítulo 2. Historia y evolución de OpenStack.....	6
2.1 Origen del proyecto (Rackspace + NASA).....	6
2.2 Hitos importantes en su desarrollo.....	6
2.3 Lanzamientos destacados y su nombre clave.....	7
2.4 Evolución tecnológica de los módulos.....	7
2.5 La comunidad de OpenStack y su gobernanza	8
Capítulo 3. Arquitectura general de OpenStack	8
3.1 Conceptos fundamentales	8
3.2 Arquitectura modular	8
3.3 Componentes principales	9
3.4 Comunicación interna y buses de mensajería	9
3.5 Interacción por API y paneles de administración	9
Capítulo 4. Componentes esenciales de OpenStack	10
4.1 Keystone (gestión de identidades)	10
4.2 Nova (gestión de cómputo)	10
4.3 Neutron (redes virtuales).....	10
4.4 Glance (imágenes)	10
4.5 Cinder (almacenamiento en bloques).....	11
4.6 Swift (almacenamiento de objetos).....	11
4.7 Horizon (interfaz gráfica).....	11
4.8 Heat (orquestación)	11
4.9 Otros servicios relevantes:	12
Capítulo 5. Infraestructura necesaria para OpenStack	12
5.1 Requisitos hardware.....	12
5.2 Requisitos de red.....	14
5.3 Requisitos de software	15
5.4 Consideraciones de seguridad.....	16
5.5 Planificación de arquitectura cloud.....	17

5.6 Comparación entre instalaciones All-in-One y multinodo.....	18
Capítulo 6. Redes en OpenStack	19
6.1 Conceptos de redes virtuales.....	19
6.2 Interfaces, puertos y routers virtuales	20
6.3 Seguridad: Grupos de seguridad y reglas.....	21
6.4 Redes proveedor vs redes de proyecto	22
6.5 QoS, VLAN, VXLAN y modelos de encapsulación.....	23
6.6 Arquitectura de Neutron en profundidad	24
Capítulo 7. Almacenamiento en OpenStack	25
7.1 Diferencias entre Cinder, Swift y Glance	25
7.2 Backend de almacenamiento: LVM, Ceph, NFS	27
7.3 Snapshots, volúmenes y backups	27
7.4 Imágenes: creación y administración	28
7.5 Buenas prácticas para archivos, bases de datos y servidores web	29
Capítulo 8. Seguridad en OpenStack	29
8.1 Autenticación y autorización	29
8.2 Seguridad en la red.....	31
8.3 Políticas de acceso a servicios	32
8.4 Firewalls, TLS, certificados	34
8.5 Buenas prácticas para la nube privada	34
Capítulo 9. Casos de uso reales de OpenStack	35
9.1 Empresas que utilizan OpenStack.....	35
9.2 Comparación con AWS, Azure y Google Cloud	37
9.3 Ventajas de crear un servidor web en OpenStack.....	39
9.4 Escalabilidad y tolerancia a fallos.....	40
9.5 Microservicios, contenedores y Kubernetes	43
Capítulo 10. Preparación del entorno.....	44
10.1 Requisitos de la máquina virtual.....	45
Capítulo 11. Instalación de la máquina virtual (Kubuntu 24.04.3 (Noble Numbat))	45
Capítulo 12. Instalación de OpenStack.....	56
Capítulo 13. Configuración de OpenStack para la implementación de un servidor web.	62

Capítulo 1. Introducción General

1.1 ¿Qué es el cómputo en la nube?



El cómputo en la nube es un modelo tecnológico que permite acceder a recursos informáticos como servidores, almacenamiento, redes y aplicaciones a través de Internet sin necesidad de administrar directamente hardware físico. Este enfoque proporciona una gran flexibilidad, ya que los recursos pueden escalar dinámicamente y automáticamente según la demanda, y también reduce costos operativos al pagar solo por lo que se utiliza. Gracias a esto, la nube se ha convertido en la plataforma ideal para implementar servicios como servidores web, permitiendo crear instancias virtuales rápidamente y configurarlas según las necesidades del proyecto. Por esto nos lleva a comprender cómo el funcionamiento del cloud computing es fundamental para cualquier implementación de servicios web, ya que permite aprovechar al máximo sus capacidades para instalar y configurar servidores como Apache o Nginx de manera eficiente y sin depender de infraestructura física local.

1.2 Modelos de servicio (IaaS, PaaS, SaaS)

Los modelos de servicio en la nube determinan el nivel de responsabilidad que asume el proveedor y el usuario en la administración del entorno. En la Infraestructura como Servicio (IaaS), el proveedor ofrece recursos básicos como máquinas virtuales y redes, mientras que el usuario gestiona el sistema operativo, las configuraciones y las aplicaciones. En la Plataforma como Servicio (PaaS), el usuario recibe un entorno completo listo para desarrollar y ejecutar aplicaciones, sin preocuparse por el sistema subyacente. Finalmente, en el Software como Servicio (SaaS), el usuario accede a aplicaciones ya creadas desde un navegador sin necesidad de instalarlas o mantenerlas. Para un trabajo orientado a la instalación de servidores web, el modelo IaaS es el más relevante, nos permite configurar libremente una instancia virtual, instalar paquetes como Apache o Nginx y desplegar una aplicación web sencilla, tal como se haría en un entorno profesional real.

1.3 Modelos de despliegue (Privada, Pública, Híbrida, Comunitaria)



Los modelos de despliegue definen la forma en que la nube es administrada y utilizada por las organizaciones. La nube pública es operada por proveedores externos y ofrece acceso abierto a los recursos bajo demanda. La nube privada pertenece a una organización específica y permite un control total sobre la seguridad, la administración y la configuración del entorno. La nube híbrida combina características de ambas para obtener un balance entre flexibilidad y seguridad, mientras que la nube comunitaria es compartida entre instituciones con objetivos o necesidades comunes. Al realizar un trabajo práctico de implementación de servicios web, es común utilizar una nube privada basada en herramientas como OpenStack, que permite crear y administrar instancias personalizables y seguras. Esto nos facilita instalar y configurar servidores web y desplegar aplicaciones sin depender de hardware físico ni ser afectado por limitaciones externas.

1.4 ¿Dónde encaja OpenStack dentro del ecosistema cloud?

OpenStack es una plataforma de código abierto diseñada para construir y administrar infraestructuras de nube privada e híbrida bajo el modelo de Infraestructura como Servicio. Su arquitectura modular ofrece componentes especializados para el manejo de máquinas virtuales, redes, almacenamiento, imágenes y autenticación, permitiendo crear entornos totalmente controlables y escalables. OpenStack se integra dentro del ecosistema cloud como una alternativa flexible a soluciones comerciales, proporcionando todas las herramientas necesarias para que una organización gestione su propia nube interna. En un proyecto de implementación de un servidor web, OpenStack cumple un papel esencial al permitir crear una máquina virtual desde cero, asignarle redes, configurar recursos y disponer del entorno ideal para instalar Apache o Nginx y desplegar una aplicación web funcional.

1.5 Ventajas del uso de la nube para servidores web

Alojar servidores web en la nube ofrece beneficios significativos como escalabilidad automática, alta disponibilidad, reducción de costos y facilidad en el despliegue. Los recursos pueden aumentar o disminuir según el tráfico, lo que evita interrupciones en momentos de alta demanda. Además, las instancias pueden clonarse, restaurarse o migrar rápidamente, lo que facilita las pruebas y la recuperación ante fallos. La nube también ofrece mecanismos de seguridad integrados, monitoreo avanzado y actualizaciones centralizadas, lo que mejora la estabilidad del servicio. Estas características hacen que la nube sea un entorno ideal para la instalación de servidores web y el despliegue de aplicaciones sencillas, ya que permite trabajar con eficiencia, rapidez y confiabilidad, replicando las condiciones reales que se encuentran en entornos empresariales modernos.

Capítulo 2. Historia y evolución de OpenStack

2.1 Origen del proyecto (Rackspace + NASA)

OpenStack nació oficialmente en julio de 2010 como una iniciativa colaborativa entre Rackspace, una empresa líder en hosting y servicios en la nube, y la NASA, que en ese momento desarrollaba su propio proyecto interno llamado Nebula para gestionar grandes cantidades de datos científicos. La NASA necesitaba una plataforma abierta, flexible y escalable para manejar cómputos masivos asociados a misiones espaciales, mientras que Rackspace buscaba estandarizar y abrir al público algunas tecnologías propietarias que ya usaba en sus infraestructuras de almacenamiento. La combinación de estos intereses dio como resultado el lanzamiento de una plataforma completamente open source capaz de ofrecer administración de recursos en la nube al estilo de proveedores como AWS o Azure, pero sin costos de licencia y con total libertad de personalización.

2.2 Hitos importantes en su desarrollo

A medida que el proyecto avanzó, OpenStack logró consolidarse gracias a una serie de hitos que marcaron su crecimiento. Uno de los primeros eventos clave fue la liberación del código fuente inicial de los servicios Nova (cómputo) y Swift (almacenamiento), lo que permitió a desarrolladores de todo el mundo comenzar a contribuir. Posteriormente, en 2012 se creó la OpenStack Foundation, una organización independiente encargada de coordinar el desarrollo, fomentar la adopción y garantizar la neutralidad del proyecto. También se incorporaron al

ecosistema actores importantes como IBM, HP, Cisco, Intel, Red Hat, Canonical y SUSE, quienes aportaron recursos y soporte empresarial. A nivel técnico, OpenStack logró hitos como la integración de redes definidas por software (SDN), la compatibilidad con IPv6, mejoras en seguridad, soporte para contenedores, automatización avanzada y un mayor enfoque en la estabilidad para entornos de producción.

2.3 Lanzamientos destacados y su nombre clave

Una de las características distintivas de OpenStack es su sistema de lanzamientos semestrales, donde cada versión recibe un nombre clave inspirado en lugares cercanos a las sedes de las conferencias de desarrollo. Entre sus versiones más representativas se encuentran "Austin" (2010), que fue la primera muestra funcional del proyecto; "Essex" (2012), que introdujo mejoras críticas en estabilidad; "Grizzly" (2013), que marcó la expansión del número de componentes; y "Havana" (2013), que destacó por avances en automatización y dashboards. Posteriormente, versiones como "Juno", "Kilo", "Liberty", "Mitaka" y "Newton" consolidaron funciones empresariales como la alta disponibilidad, la integración con herramientas de orquestación y mejoras en rendimiento. Con el tiempo, el ecosistema adoptó nombres más maduros y un ritmo más estable, orientado a mejorar la calidad sobre la cantidad de funciones nuevas.

2.4 Evolución tecnológica de los módulos

OpenStack inició con solo dos módulos principales, pero con el paso del tiempo evolucionó hasta convertirse en un ecosistema de decenas de componentes especializados. Nova, encargado del cómputo, se amplió para soportar múltiples hipervisores como KVM, Xen, VMware y Hyper-V, además de permitir gestión avanzada de máquinas virtuales. Swift, el sistema de almacenamiento de objetos, mejoró su resiliencia y se convirtió en una alternativa real a soluciones como Amazon S3. Más tarde surgieron módulos como Neutron para redes virtuales con soporte para SDN y NFV; Cinder para almacenamiento en bloques; Glance para administración de imágenes; Keystone para autenticación y control de acceso; y Horizon, que proporcionó una interfaz web para gestionar todos los servicios de forma centralizada. La evolución no solo consistió en añadir más servicios, sino en refinar su interoperabilidad, escalabilidad y capacidad de adaptarse a ambientes híbridos, contenedores y nubes distribuidas.

2.5 La comunidad de OpenStack y su gobernanza

Desde sus inicios, OpenStack se ha caracterizado por tener una de las comunidades más activas y diversas dentro del mundo del software libre. Miles de desarrolladores, empresas, operadores de nubes públicas y privadas, instituciones académicas e investigadores participan activamente en la mejora del código, pruebas, documentación y soporte. La gobernanza del proyecto se organiza a través de la OpenInfra Foundation (antes OpenStack Foundation), que garantiza que el proyecto sea neutral, abierto y orientado al interés común. Esta fundación establece comités técnicos, equipos de mantenimiento y normas de contribución, asegurando que todas las decisiones se tomen de forma colaborativa y transparente. Además, los eventos mundiales como el OpenInfra Summit fomentan el intercambio de experiencias y la construcción de nubes empresariales estandarizadas.

Capítulo 3. Arquitectura general de OpenStack

3.1 Conceptos fundamentales

OpenStack se basa en el concepto de nube como un conjunto de recursos virtualizados cómputo, almacenamiento y redes administrados de forma centralizada y accesibles mediante APIs. A diferencia de una infraestructura tradicional, donde los servidores y servicios se gestionan manualmente, OpenStack utiliza un enfoque altamente automatizado que permite crear, eliminar y escalar recursos bajo demanda. Esto se logra mediante la abstracción de hardware físico, convirtiéndolo en recursos virtuales que los usuarios pueden consumir de manera flexible. Asimismo, la plataforma se sustenta en principios de software libre, interoperabilidad entre módulos, escalabilidad horizontal y administración distribuida.

3.2 Arquitectura modular

Una de las características más destacadas de OpenStack es su arquitectura modular, donde cada servicio cumple una función independiente, pero trabaja en conjunto con los demás mediante servicios API y una capa de mensajería. Esto permite que la plataforma sea flexible y adaptable: cada módulo puede instalarse, actualizarse o reemplazarse sin afectar por completo al sistema. El modularidad también facilita la implementación en ambientes híbridos, permitiendo integrar herramientas externas y ajustarse a diferentes casos de uso, como nubes públicas, privadas o de investigación académica. Gracias a este diseño, los

administradores pueden elegir únicamente los componentes que realmente necesitan, haciendo la plataforma más ligera o completa según el contexto.

3.3 Componentes principales

OpenStack está compuesto por varios servicios esenciales que permiten gestionar cada parte de la nube. Nova se encarga del cómputo y permite crear máquinas virtuales; Neutron gestiona las redes virtuales y la conectividad entre instancias; Cinder proporciona almacenamiento en bloques, útil para bases de datos y aplicaciones persistentes; Swift ofrece almacenamiento de objetos escalable para grandes cantidades de datos; Glance administra las imágenes de sistemas operativos; y Keystone maneja la autenticación y autorización de usuarios. También se incluyen otros componentes importantes como Horizon, el panel web de administración; Heat, para orquestación; y Ironic, para provisión de hardware bare-metal. Todos estos módulos trabajan de manera integrada para ofrecer una nube completa.

3.4 Comunicación interna y buses de mensajería

La comunicación entre los servicios de OpenStack se realiza típicamente mediante un sistema de mensajería, siendo RabbitMQ, Qpid o ZeroMQ los más utilizados. Estos buses permiten que los módulos intercambien comandos, solicitudes y eventos de forma asíncrona, asegurando que los servicios funcionen de manera coordinada incluso en entornos distribuidos. Por ejemplo, cuando un usuario solicita crear una máquina virtual, Nova coordina con Glance para recuperar la imagen, luego con Neutron para asignar una red y finalmente con Cinder si necesita almacenamiento adicional. Este flujo depende de un sistema de mensajería confiable y de alto rendimiento.

3.5 Interacción por API y paneles de administración

Todos los componentes de OpenStack exponen APIs REST que permiten el control programático de la nube. Esto es clave para integraciones con herramientas externas, automatización, desarrollo de aplicaciones y orquestación avanzada. Gracias a estas APIs, los administradores y usuarios pueden interactuar con la plataforma desde scripts, sistemas externos o herramientas de DevOps. Además, OpenStack dispone de Horizon, un panel de administración gráfico que facilita la gestión sin necesidad de usar la línea de comandos. Horizon permite crear instancias, configurar redes, asignar permisos y monitorear recursos mediante una interfaz amigable.

Capítulo 4. Componentes esenciales de OpenStack

4.1 Keystone (gestión de identidades)

Keystone es el servicio de autenticación y autorización de OpenStack. Su función principal es gestionar usuarios, proyectos, roles y tokens de acceso, permitiendo controlar qué recursos puede utilizar cada cliente o administrador dentro de la nube. Keystone actúa como un punto centralizado de seguridad, ya que todos los demás servicios dependen de él para validar credenciales y establecer permisos. También ofrece catálogos de servicios para que cada módulo de OpenStack sea localizado y consumido mediante APIs.

4.2 Nova (gestión de cómputo)

Nova es el componente responsable de la creación, administración y ciclo de vida de las máquinas virtuales. Es el “motor” de cómputo de OpenStack y se integra con hipervisores como KVM, QEMU, Xen o VMware. Nova coordina la ejecución de instancias, asigna recursos de CPU y memoria, gestiona su estado, controla migraciones en vivo y se comunica con otros módulos como Neutron y Cinder para completar el aprovisionamiento. Puede operar tanto en arquitecturas pequeñas como en grandes nubes distribuidas.

4.3 Neutron (redes virtuales)

Neutron es el módulo encargado de gestionar las redes virtuales en OpenStack. Permite crear subredes, routers, balanceadores de carga, firewalls y configurar reglas de acceso. Integra tecnologías como VLANs, VXLAN, GRE y SDN, lo que posibilita construir redes complejas y seguras para las instancias. Cada máquina virtual puede obtener una IP privada y pública controlada por políticas de seguridad. Neutron también permite la integración con plataformas externas como Open vSwitch y servicios avanzados de red.

4.4 Glance (imágenes)

Glance administra las imágenes de sistemas operativos utilizadas para crear instancias en Nova. Las imágenes pueden incluir Linux, Windows u otras distribuciones personalizadas, y pueden almacenarse en distintos backends como Swift, Cinder o sistemas de archivos locales.

Glance soporta formatos como QCOW2, RAW o ISO, permitiendo que los usuarios suban, modifiquen y repliquen imágenes fácilmente.

4.5 Cinder (almacenamiento en bloques)

Cinder proporciona almacenamiento en bloques persistente para las máquinas virtuales, similar al funcionamiento de un disco duro dedicado. Permite crear, adjuntar y gestionar volúmenes que pueden usarse para bases de datos, aplicaciones críticas o almacenamiento que debe mantenerse, aunque la instancia se reinicie o elimine. Cinder soporta múltiples backends como LVM, Ceph, NFS o sistemas empresariales.

4.6 Swift (almacenamiento de objetos)

Swift es un sistema de almacenamiento de objetos que permite guardar grandes volúmenes de datos no estructurados, como imágenes, videos, archivos de respaldo o datos de aplicaciones web. Su arquitectura distribuida garantiza disponibilidad y resiliencia, replicando objetos en distintos nodos. Swift permite crecimiento ilimitado sin afectar el rendimiento y es ideal para datos que no requieren estructura fija.

4.7 Horizon (interfaz gráfica)

Horizon es el panel web oficial de OpenStack. Permite a administradores y usuarios gestionar todos los recursos de la nube a través de una interfaz gráfica fácil de usar. Desde Horizon se pueden crear instancias, configurar redes, administrar volúmenes, manejar usuarios, asignar roles y monitorear recursos. Todo esto evita depender exclusivamente de la línea de comandos y facilita el uso para principiantes.

4.8 Heat (orquestación)

Heat es el componente encargado de la orquestación en OpenStack. Permite desplegar infraestructuras completas mediante plantillas (templates) escritas en YAML, conocidas como HOT (Heat Orchestration Templates). Estas plantillas describen redes, instancias, volúmenes, balanceadores y otros recursos, automatizando procesos que normalmente requieren muchas configuraciones manuales.

4.9 Otros servicios relevantes:

- Trove (bases de datos) : Trove es el servicio que permite desplegar y administrar bases de datos como MySQL, PostgreSQL o MongoDB dentro de OpenStack de forma automatizada. Simplifica tareas como backups, restauración y escalamiento.
- Ironic (bare metal) : Permite aprovisionar hardware físico real (bare metal) usando los mismos mecanismos de OpenStack. Esto es útil para cargas que requieren rendimiento total sin virtualización.
- Magnum (contenedores) : Gestiona clústeres de contenedores como Kubernetes, Docker Swarm o Mesos dentro de OpenStack. Automatiza su despliegue y administración. Magnum permite integrar tecnologías de contenedores con la infraestructura de OpenStack.
- Barbican (gestión de llaves) : Es el servicio de gestión de secretos como claves, certificados y contraseñas. Garantiza almacenamiento seguro y cifrado dentro de la plataforma. Barbican fortalece la seguridad de OpenStack al administrar de forma centralizada claves y secretos críticos.

Capítulo 5. Infraestructura necesaria para OpenStack

5.1 Requisitos hardware

Para desplegar una nube basada en OpenStack, es fundamental analizar los requisitos de hardware con el fin de garantizar disponibilidad, rendimiento, escalabilidad y mantenibilidad. Esta sección describe los componentes mínimos y recomendados en función del objetivo: desde entornos de prueba hasta producción.

En primer lugar, los nodos de control (control plane) deben disponer de CPU, memoria RAM y discos rápidos suficientes para alojar servicios críticos como el catálogo de identidades (Keystone), imágenes (Glance), API de cómputo (Nova), redes (Neutron) y colas de mensajería (ej. RabbitMQ). Por ejemplo, la documentación oficial sugiere reservar memoria adicional si en el mismo nodo coexisten servicios de almacenamiento o hipervisores (OpenStack Docs, s. f.).

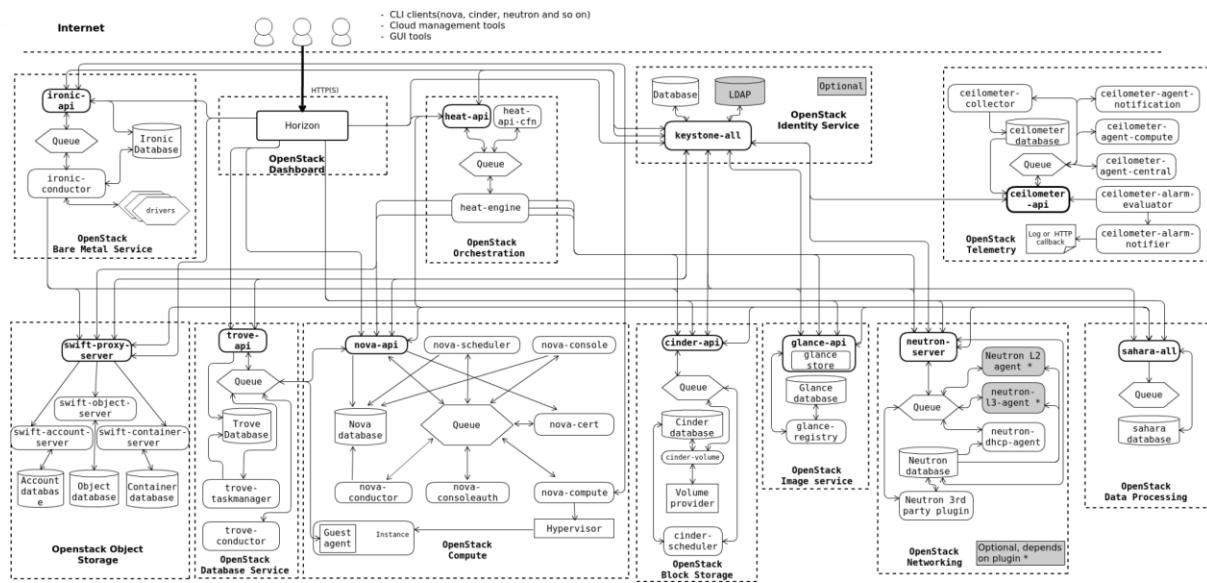


Fig 2. Arquitectura de una nube de Openstack

En segundo lugar, los nodos de cómputo requieren recursos para alojar instancias virtuales (VMs), lo cual implica una planificación rigurosa de CPU, RAM, capacidad de disco local o conectada en red y transferencia de red. La asignación debe permitir reservar una fracción suficiente de recursos para las instancias sin comprometer los servicios operativos del hipervisor. Conforme indican guías de despliegue de nivel de producción, en entornos con almacenamiento distribuido (como Ceph) la memoria y CPU empleada por demonios de almacenamiento debe considerarse en el dimensionamiento (Red Hat, s. f.).

En tercer lugar, el almacenamiento requiere atención especial: tanto los discos en los nodos de control como en los nodos de almacenamiento deben disponer de IOPS y latencia adecuados. En las arquitecturas en producción se recomienda el uso de múltiples unidades en RAID o un backend distribuido como Ceph, con separación de roles para base de datos, mensajería, logs y datos de VM (Red Hat, s. f.).

Finalmente, la planificación de red física debe contemplar la **separación de planos**: management, storage, public/external y guest/tenant. Esto permite aislar tráfico operativo del de usuario, mejorar rendimiento y la seguridad. Se recomienda utilizar interfaces físicas dedicadas o agregadas (LACP) con redundancia (OpenStack HA Guide, s. f.).

5.2 Requisitos de red

La red es, en muchos despliegues de OpenStack, el componente que marca la diferencia entre una solución robusta o una fuente de cuellos de botella. En esta sección se analizan los planos de red, tipos de segmentación, tecnologías de encapsulación y buenas prácticas.

El plano **management** alberga la comunicación entre los servicios internos (API, mensajería, base de datos). Debe estar sobre una red aislada que no comparta tráfico de usuarios finales.

El plano **guest/tenant** es el que utilizan las instancias virtuales para comunicarse entre ellas o con el exterior; en esta capa los modelos varían ampliamente y dependen de tecnologías como VLAN, VXLAN o GRE (OpenStack Neutron Admin Guide, s. f.).

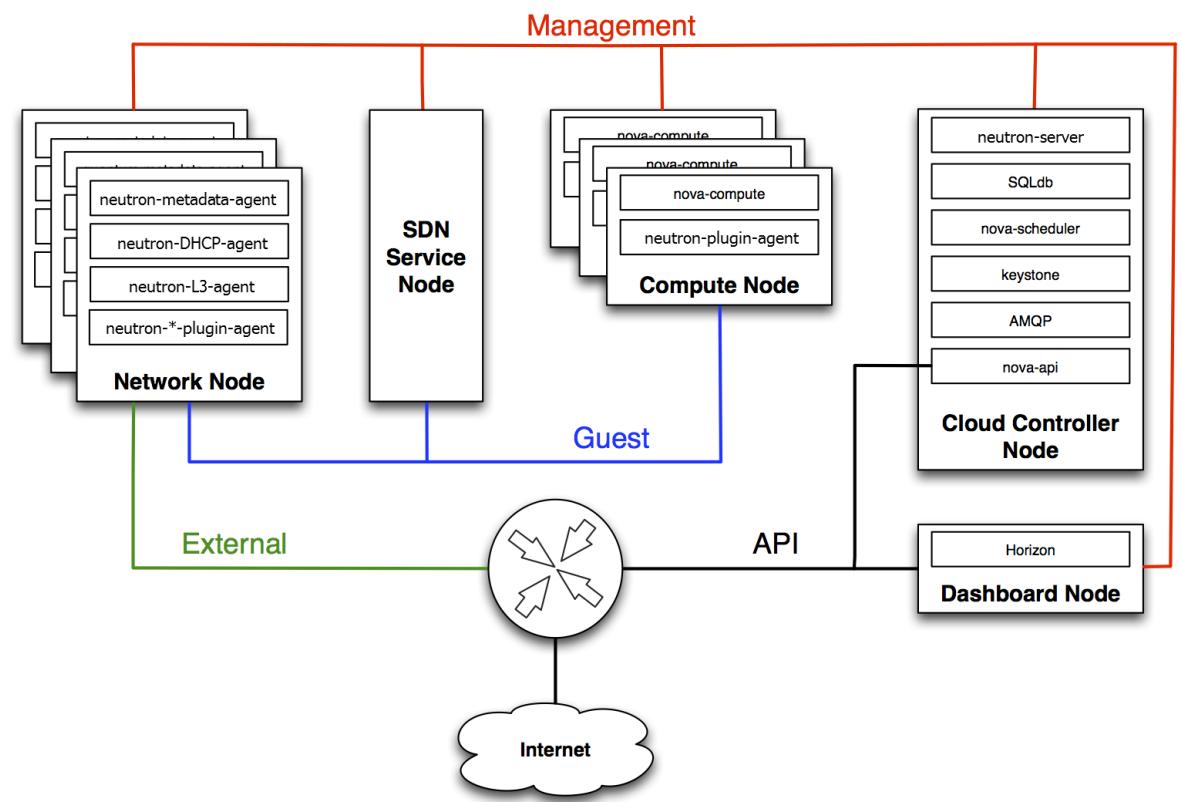


Fig 2.2 Neutron reference architecture (OVN)

Para entornos multi-tenant o de alto rendimiento, la utilización de **encapsulación** (por ejemplo VXLAN o GENEVE) permite un gran número de segmentos lógicos. No obstante, introduce sobrecarga de encapsulación/decapsulación, por lo que se debe evaluar el off-load del hardware (NICs compatibles) y ajustar el MTU de red para evitar fragmentación (OpenStack Neutron ML2 Plugin Docs, s. f.).

type driver / mech driver	Flat	VLAN	VXLAN	GRE	Geneve
Open vSwitch	yes	yes	yes	yes	yes
OVN	yes	yes	yes (requires OVN 20.09+)	no	yes
SRIOV	yes	yes	no	no	no
MacVTap	yes	yes	no	no	no
L2 population	no	no	yes	yes	yes

Figura 6.2 Comparación entre VLAN y VXLAN.

La segregación de tráfico también implica el plano **storage**, cuando se usa un backend como Ceph: requiere una red de baja latencia y alto ancho de banda, y lo ideal es que esté lo más separada posible de la red guest. De este modo, el tráfico de datos de almacenamiento no compite con el de instancias o de management. Adicionalmente, la redundancia de red (switches duales, enlaces agregados) y políticas de QoS pueden evitar congestión en escenarios de I/O intensivo (HA Guide, s. f.).

Por último, no debe descuidarse la seguridad de la red: segmentación, firewalls internos, control de acceso entre planos y monitorización del tráfico son componentes esenciales. El diseño debe prever tanto el rendimiento máximo como los escenarios de fallo y recuperación ante incidentes.

5.3 Requisitos de software

El software debe seleccionarse y versionarse cuidadosamente para asegurar compatibilidad, estabilidad y soporte de producción. Esta sección presenta los aspectos clave: sistema operativo, dependencias, hipervisor, bases de datos y colas de mensajería.

Las distribuciones soportadas oficialmente –como Ubuntu LTS, CentOS/RHEL (mediante RDO)– garantizan compatibilidad con OpenStack y sus componentes. Además, cada release de OpenStack publica una “matriz de compatibilidad” donde se listan las versiones aceptadas de Python, MySQL/MariaDB, RabbitMQ, KVM/libvirt, etc. (OpenStack Project Deploy Guide, s. f.). Se recomienda documentar estas versiones en el manual de usuario para reproducibilidad.

El hipervisor (normalmente KVM/libvirt) debe configurarse con performance optimizada (por ejemplo, virtio, SLAT, CPU pinning). Los agentes de red (como Open vSwitch o linuxbridge) requieren drivers y configuración compatibles. En producciones, los agentes de almacenamiento (Ceph OSDs, MONs) también demandan software específico. Por ello, se aconseja emplear herramientas de automatización (como Ansible) para gestionar actualizaciones, revisar parches y mantener un inventario de versiones.

Asimismo, los sistemas de base de datos (ej. MariaDB Galera) y cola de mensajes (RabbitMQ) deben dimensionarse y configurarse para HA: replicación, fail-over y monitorización. Por ejemplo, si Keystone, Nova, Glance y otros comparten la misma base de datos, la memoria y la CPU del servidor de base deben contemplar contingencias de carga máxima.

Finalmente, se deben definir procedimientos de actualización (upgrades) y mantenimiento: entornos de staging, pilas de pruebas, rollback plan y automatización del despliegue. Esto contribuye a mantener la nube estable, segura y con los mínimos tiempos de inactividad.

5.4 Consideraciones de seguridad

La seguridad en OpenStack debe abordarse desde múltiples frentes: físico, lógico, de red, almacenamiento y operaciones. Esta sección detalla los controles esenciales antes de la puesta en marcha.

Desde el punto de vista físico, el acceso a los nodos (controladores, almacenamiento, switches) debe estar restringido a personal autorizado. Los racks deben protegerse y los switches configurados con políticas de administración segura. En el plano lógico, todas las API deben usar TLS, se debe forzar la autenticación fuerte (idealmente con MFA) y rotar credenciales de servicio periódicamente (Red Hat Security and Hardening Guide, s. f.).

El servicio Keystone gestiona identidades, proyectos y roles. Es imprescindible aplicar el principio de privilegio mínimo: los usuarios solo reciben los roles imprescindibles para su trabajo y las credenciales de servicio tienen permisos mínimos. Asimismo, integrar Keystone con LDAP/AD o SSO mejora la gestión centralizada de identidades (Red Hat Security Guide, s. f.).

En la red, los grupos de seguridad (security groups) y ACLs en Neutron deben configurarse para limitar el tráfico entrante/saliente. Se recomienda una arquitectura de micro-segmentación (por ejemplo, mediante OVN) para aislar servicios y reducir superficie de ataque. Además, cifrar los volúmenes (mediante Cinder) y objetos (Swift/RGW) añade otra capa de protección (Red Hat Security Guide, s. f.).

Operacionalmente, se deben definir runbooks de incidentes, auditorías periódicas, pruebas de penetración y políticas de parcheo. La monitorización de logs (syslog, Ceph logs, Neutron agents) y la integración con un SIEM mejoran la detección de anomalías. La automatización de los procesos de seguridad reduce errores humanos y mejora la continuidad del servicio.

5.5 Planificación de arquitectura cloud

La planificación de la arquitectura es un paso estratégico: define dimensiones, topología, escalabilidad, roles de los nodos y crecimiento previsto. En esta sección se describe cómo abordar este proceso.

Primero, se debe establecer claramente el alcance del proyecto: número de usuarios o instancias estimadas, tipos de cargas de trabajo (web estática, bases de datos, contenedores), requisitos de SLA (tiempo de actividad, rendimiento, recuperación). Sobre esta base se dimensiona CPU, RAM, almacenamiento e interconexión de red (OpenStack HA Guide, s. f.).

Segundo, definir la topología del despliegue: número de controladores, nodos de red (en diseños que los separen), nodos de cómputo, nodos de almacenamiento. Hay que decidir si usarás un modelo distribuido (multinodo) o una instalación más ligera (All-in-One) y prever escalabilidad horizontal (añadir nodos) y vertical (mejorar nodos existentes). (OpenStack Project Deploy Guide, s. f.).

Tercero, integrar otros sistemas: identidad (LDAP/AD), backup y recuperación, orquestación (Heat), monitorización, facturación, seguridad. Cada integración tiene impacto en requisitos de hardware, red y software. Por ejemplo, un clúster Ceph integrado con OpenStack requiere redes dedicadas y planificación de réplica/erasure coding (OpenStack-Ansible Guide, s. f.).

Cuarto, definir la gobernanza y los procedimientos operativos: naming convention, políticas de recursos, límites por proyecto, runbooks para tareas rutinarias (creación de proyecto,

escalado, mantenimiento) y escalación de incidentes. Este aspecto operativo es fundamental para una nube mantenible (OpenStack Project Deploy Guide, s. f.).

Por último, probar la arquitectura antes del go-live: pruebas de estrés, fail-over, recuperación ante desastres, migraciones de carga. Documenta los resultados y ajusta la arquitectura según los hallazgos. Esta etapa reduce riesgos de fallo real y mejora la madurez operativa.

5.6 Comparación entre instalaciones All-in-One y multinodo

Un diseño All-in-One se caracteriza por reunir todos los servicios de OpenStack en un único nodo (por ejemplo, para PoC o entorno de desarrollo). Es rápido de desplegar, requiere pocos recursos y permite experimentar con la funcionalidad del sistema. No obstante, no refleja la complejidad o los retos de una nube de producción (Swift Deployment Guide, s. f.).

Por otra parte, la arquitectura multinodo separa los roles: múltiples controladores, nodos de cómputo, nodos de almacenamiento y, a menudo, nodos de red. Este diseño permite alta disponibilidad (HA), tolerancia a fallos, escalabilidad y rendimiento realista. Sin embargo, implica mayor inversión en hardware, red compleja y esfuerzo operativo (Red Hat Recommendations for Large Deployments, s. f.).

El manual debe mostrar tablas comparativas de requisitos, costes operativos, riesgos y casos de uso recomendados para cada modelo. Por ejemplo:

- All-in-One: ideal para pruebas, demos, aprendizaje.
- Multinodo: recomendado para producción, carga real, SLA críticos.

Finalmente, se recomienda un plan de migración: iniciar con un entorno All-in-One para validación, evolucionar a multinodo en staging, y luego desplegar en producción con una fase de pruebas y ejecución controlada (OpenStack Project Deploy Guide, s. f.).

Capítulo 6. Redes en OpenStack

6.1 Conceptos de redes virtuales

Las redes virtuales en OpenStack son la abstracción que permite crear, aislar y gestionar la conectividad entre máquinas virtuales sin necesidad de reconfigurar la red física cada vez que surge una necesidad nueva. El servicio responsable de estas abstracciones es Neutron, que expone APIs para crear Network, Subnet y Port, y coordina agentes y plugins que traducen las operaciones lógicas a acciones en el datapath (por ejemplo, en Open vSwitch u OVN). El modelo separa claramente el plano de control (APIs, base de datos, scheduler) del plano de datos (encaminamiento y forwarding) para ofrecer flexibilidad y escalabilidad.

Desde la perspectiva operativa, una red virtual típica en OpenStack se compone de: (1) una *network* (unidad lógica que agrupa direcciones y políticas); (2) una *subnet* (bloque IP asociado a esa network con su gateway y rango de DHCP); y (3) *ports* (puntos de conexión asignados a instancias (vNICs) o appliances). Estas entidades permiten asignar IPs fijas, aplicar security groups y gestionar bindings de interfaces al host físico adecuado. Entender esta jerarquía es esencial para modelar topologías de multi-tenant y diseñar políticas de acceso.

Además, el diseño de redes virtuales incluye la idea de planos lógicos (o *domains*): management, storage, provider/external y guest/tenant. Cada uno tiene objetivos y requisitos distintos: el plano management soporta la comunicación entre servicios (APIs, bases de datos), el plano storage transporta tráfico I/O de backends como Ceph, el plano provider/external conecta la nube con el mundo exterior y el plano guest es el que usan las VMs. Separar estos planos mejora rendimiento y seguridad.

OpenStack permite varios modelos de segmentación (flat, VLAN, VXLAN, GRE, GENEVE). La elección de uno u otro depende de la escala esperada, de la capacidad del underlay, y de los requisitos de aislamiento. Por ejemplo, VLAN es simple y eficiente en CPU, pero está limitada a ~4.094 IDs; VXLAN/Geneve escalan a millones de redes lógicas a costa de overhead en encapsulación. Estas decisiones deben tomarse en el diseño inicial de la nube.

Finalmente, desde la operación diaria, documentar naming conventions, rangos IP, y políticas de DHCP/MTU evita errores comunes (p. ej. fragmentación por MTU incorrecta en

overlays). También es recomendable incluir pruebas automatizadas de conectividad y validación de ACLs tras cada cambio de red. Un diagrama conceptual de la arquitectura de redes ayuda a que equipos no técnicos comprendan límites y responsabilidades.

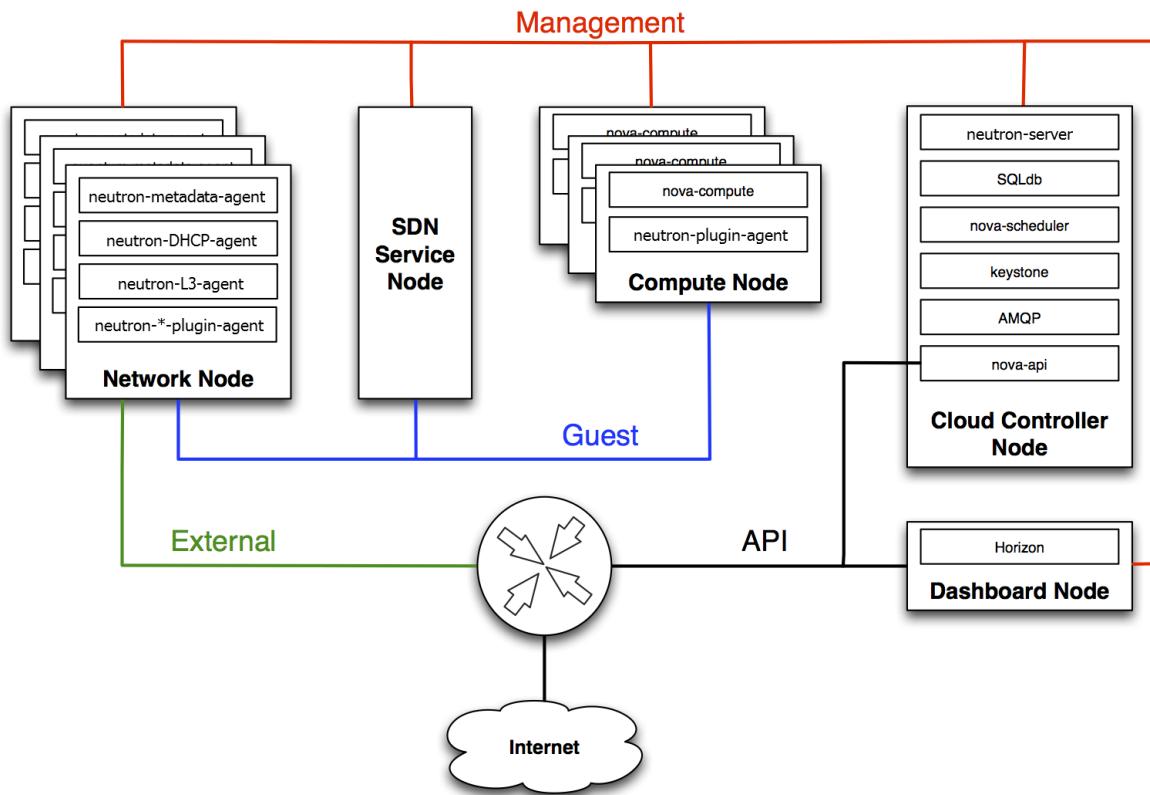


Fig 6.1 Diagrama conceptual de redes OpenStack

6.2 Interfaces, puertos y routers virtuales

Las interfaces virtuales (vNICs) que ven las instancias se representan en Neutron como ports. Un *port* puede tener una o varias IPs, uno o varios security groups, y atributos de binding que definen cómo se conecta el port al host físico o al mecanismo de datapath (por ejemplo, ovs, bridge o direct para SR-IOV). Conocer los atributos de los ports es imprescindible para tareas de troubleshooting (por ejemplo, identificar por qué una VM no tiene conectividad a pesar de tener IP asignada).

Los routers virtuales en OpenStack (servicio L3 de Neutron) proporcionan el enruteamiento entre subnets tenant y hacia la red externa mediante un *gateway* (external network). Los routers pueden gestionar NAT (SNAT) para tráfico saliente o enlazar floating IPs para exponer instancias al exterior. En arquitecturas modernas, el forwarding entre hosts para

tráfico Este-Oeste (E/W) se realiza de forma distribuida cuando se usa OVN, lo que reduce la necesidad de pasar todo por un gateway centralizado.

Además, Neutron soporta características avanzadas como router HA, distributed virtual routers (DVR) y *service chaining* (encadenamiento con appliances virtuales). DVR minimiza el salto por el nodo gateway para tráfico E/W y mejora latencias en entornos con muchos NATs. Documenta los pros/cons de cada modo y cuándo es adecuado habilitarlos (p. ej. pequeñas nubes: centralizado; nubes medianas/grandes: DVR/OVN).

Para depuración de nivel bajo, los administradores deben saber mapear un port a su namespace de Linux (en implementaciones que usan namespaces), revisar tablas ARP y rutas dentro del namespace, y usar herramientas como ip netns, ovs-vsctl, ovn-nbctl y tcpdump sobre la interfaz de la VM o el tap device vinculado. Incluir ejemplos de comandos en el manual acelera la resolución de incidentes.

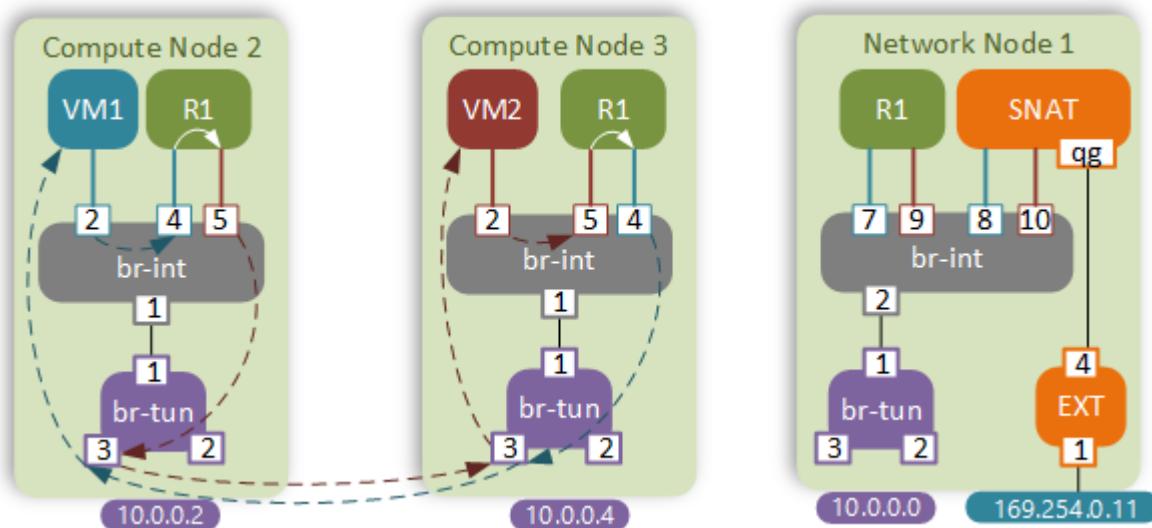


Fig 6.2 Distributed Virtual Routing(DVR)

6.3 Seguridad: Grupos de seguridad y reglas

Los security groups son firewalls stateful aplicados a nivel de instancia. Funcionan como conjuntos de reglas (ingress/egress) que especifican protocolo, puertos y origen/destino permitidos. Por defecto, un proyecto tiene un security group “por defecto” que comúnmente permite SSH y ICMP según política del operador; sin embargo, el principio de menor privilegio dicta cerrar todo y abrir solo lo necesario.

Comparado con firewalls perimetrales, los security groups permiten micro-segmentación dentro de la nube: por ejemplo, aplicar reglas distintas por rol (web, app, db) y evitar exposición innecesaria. No obstante, para inspección profunda (DPI) o cumplimiento estricto, se suelen integrar appliances (NGFW/IPS) mediante service chaining o funciones NFV, explicitando el impacto en latencia y throughput.

Además de security groups, Neutron ofrece otras técnicas de control: ACLs de routers, políticas de red en switches físicos y mecanismos SDN (OVN, NSX) que permiten definir políticas más granulares (etiquetas, grupos de seguridad basados en identidad). El manual debe detallar la jerarquía de aplicación de políticas para evitar conflictos: por ejemplo, qué ocurre si un security group y una ACL contradicen permisos.

Operativamente, incluye procedimientos para auditar reglas (búsqueda de reglas “any/any”), pruebas con `nmap/ping/curl`, y pasos para aislar una instancia comprometida (snapshot + detach + aplicar reglas restrictivas). También describe cómo conectar logs de Neutron y del hypervisor a un SIEM para correlación y alertas.

6.4 Redes proveedor vs redes de proyecto

En OpenStack existe una distinción operativa clave entre provider networks y tenant (o project) networks. Las provider networks (redes proveedor) son creadas por administradores y mapean directamente al underlay (se usan para acceder a recursos externos o para exponer IPs públicas); las tenant networks las crean los usuarios/proyectos para su aislamiento, normalmente sobre overlays (VXLAN/Geneve) o VLANs internas. La elección depende de políticas de direccionamiento, capacidad de la red física y necesidades de aislamiento.

El uso de provider networks simplifica la administración de direcciones IPv4 escasas y permite performance más directa (menos encapsulación), pero reduce la flexibilidad y el aislamiento entre proyectos. Al contrario, las tenant networks ofrecen aislamiento y escalabilidad (especialmente si se usan overlays), pero exigen mayor CPU por encapsulación y cuidado con MTU. El manual debe incluir criterios claros para escoger entre ambos en función del caso de uso.

En la práctica, muchas nubes híbridas mezclan ambos modelos: provider networks para servicios que requieren IP fija/alta performance (gateways, appliances, hosts de almacenamiento), y tenant networks para workloads de usuarios. Es importante documentar cómo se realiza la traducción entre ambos (routers, floating IPs, reglas SNAT/DNAT) y las implicaciones operativas (p. ej. troubleshooting de NAT).

Incluye ejemplos de comandos para crear cada tipo de red (`openstack network create --share --external`) y recomendaciones de naming conventions y rangos IP para evitar solapamientos en migraciones o integraciones con DC existentes.

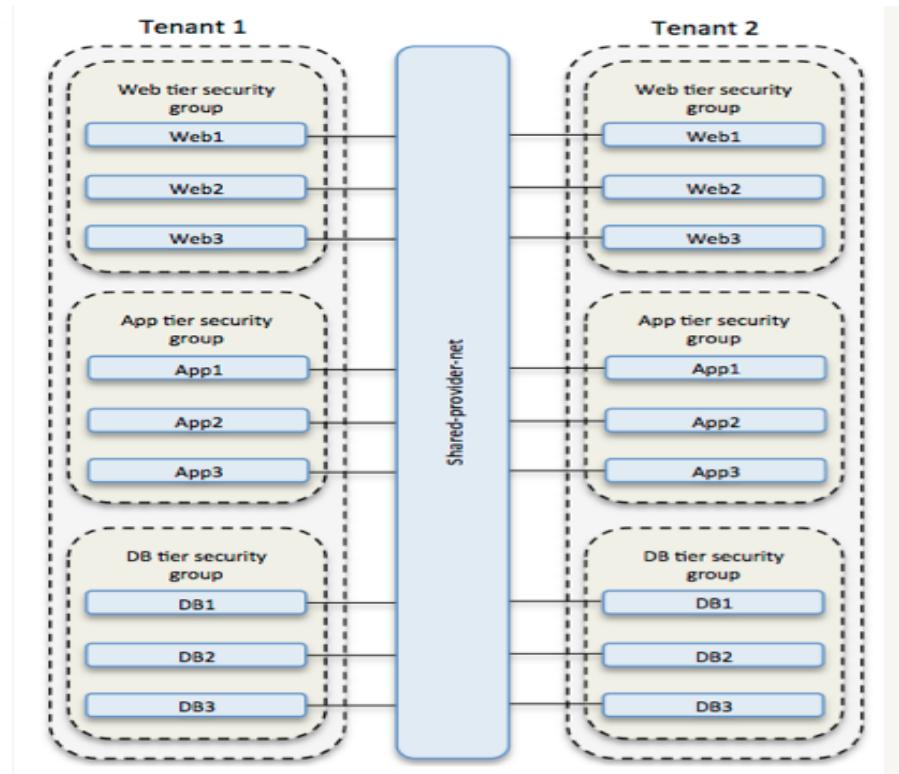


Fig. 6.4 Traditional application tiering

6.5 QoS, VLAN, VXLAN y modelos de encapsulación

La decisión sobre modelo de encapsulación (VLAN, VXLAN, GRE, Geneve) y la estrategia de QoS son determinantes para el rendimiento de la nube. VLAN es eficiente y simple, pero limitada en escala (≈ 4.094 IDs). VXLAN/Geneve permiten millones de redes lógicas (VNI de 24 bits) y son la opción de facto cuando se necesita multitenancy a gran escala; sin embargo, añaden overhead de encapsulación que puede impactar la CPU y el MTU.

QoS en Neutron permite aplicar reglas de limitación o prioridad a ports o networks (rate limiting, DSCP marking), pero su efectividad depende del soporte del underlay. Si los switches físicos no respetan las marcas o no están configurados para priorizar tráfico, las políticas de QoS en el overlay tendrán efecto limitado. Por eso, un diseño coherente entre overlay y underlay y pruebas de extremo a extremo son indispensables.

Desde el punto de vista operativo, se debe calibrar MTU (por ejemplo aumentar a 1600/9000 cuando se usa VXLAN o túneles), habilitar offloads en NICs si es posible, y evaluar SR-IOV/DPDK para cargas con requisitos de latencia o throughput muy exigentes (NFV, telecom). Documenta benchmarks esperables y pruebas de impacto.

También es importante contemplar la coexistencia de varios tipos de segmentos (ML2 permite múltiples type drivers simultáneamente). Por ejemplo, en la misma nube puede haber redes VLAN para appliances de rendimiento y VXLAN para tenant networks. El manual debe explicar cómo ML2 gestiona el mapeo y binding de ports a segmentos concretos.

6.6 Arquitectura de Neutron en profundidad

Neutron está diseñado como un conjunto de componentes: **neutron-server** (API y control), ML2 plugin (framework de type/mechanism drivers), y agentes distribuidos en hosts (L2 agent, L3 agent, DHCP agent, metadata agent). El ML2 framework permite mezclar *type drivers* (VLAN/VXLAN/GRE) con *mechanism drivers* (OVS, linuxbridge, OVN) para adaptar la solución al entorno físico y a los requisitos de performance. Este modularidad es la que permite que Neutron funcione tanto en pequeños PoC como en nubes a escala.

En implementaciones modernas, OVN (Open Virtual Network) surge como el backend preferido por su arquitectura distribuida: mantiene la información de la red lógica en una base de datos central (OVN Northbound) y programa el datapath de forma automática en los hipervisores (OVS), lo que reduce la complejidad de agentes centralizados y mejora escalabilidad. OVN facilita funcionalidades nativas como routers distribuidos y logical switching sin necesidad de tantos procesos en nodos de red.

Otro aspecto esencial es el colocamiento de servicios (service placement). En arquitecturas de referencia se separan roles: controlador(s), gateways, compute nodes y storage nodes. Con OVN, mucho del forwarding E/W se mantiene en los compute nodes, mientras que los gateway nodes se usan para N/S (internet) y SNAT. El manual debe incluir recomendaciones sobre cuántos agentes/instances desplegar por escala (por ejemplo, balancear gateways para N/S y distribuir control plane para tolerancia a fallos).

Para operaciones y troubleshooting, documenta métricas clave de Neutron (latencia API, número de ports por host, tasa de churn de ports, uso CPU de agentes), y procedimientos de recuperación (reconectar agentes, reparar base de datos OVN, limpiar bindings huérfanos). Incluir scripts básicos y comandos ovn-sbctl, ovn-nbctl, ovs-vsctl, y ejemplos de openstack network show acelerará la respuesta operacional.

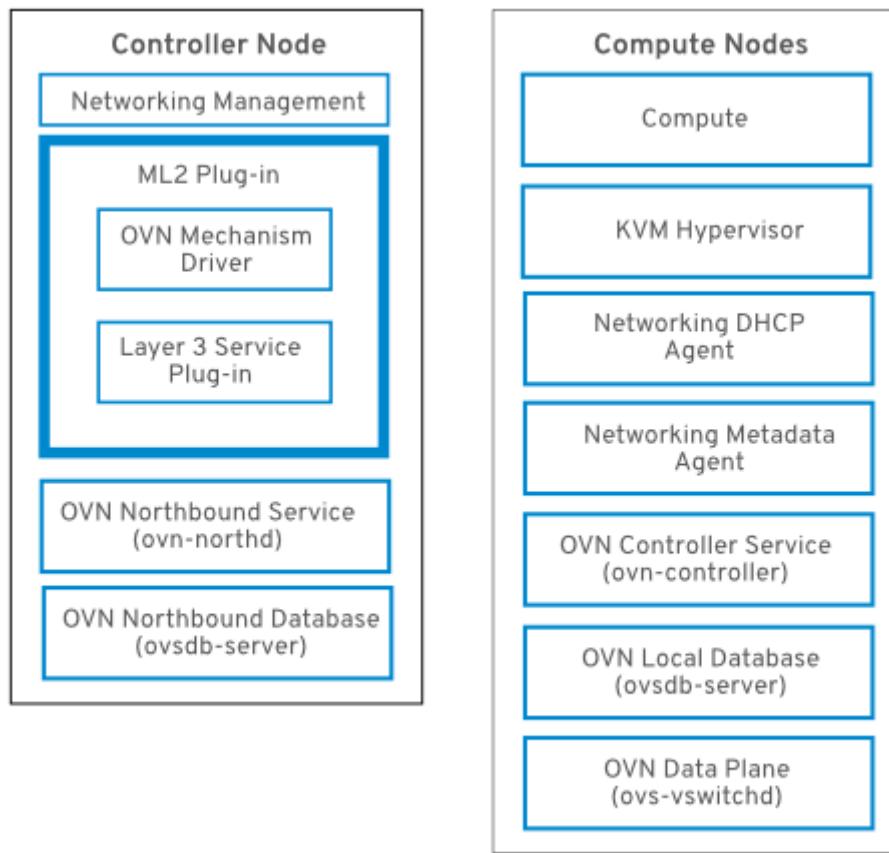


Fig. 6.6 Open Virtual Network (OVN)

Capítulo 7. Almacenamiento en OpenStack

El almacenamiento en OpenStack constituye uno de los pilares fundamentales para la operación de infraestructuras en la nube. Tanto las instancias, como las imágenes, los volúmenes persistentes, las copias de seguridad y los objetos estáticos dependen directamente de los servicios de almacenamiento que ofrece OpenStack. Esta sección ofrece un análisis detallado y claro de cómo funciona el almacenamiento, los componentes involucrados, sus diferencias clave, los tipos de backend compatibles y las mejores prácticas para su implementación en entornos de producción o educativos.

7.1 Diferencias entre Cinder, Swift y Glance

OpenStack incorpora tres servicios de almacenamiento principales, cada uno orientado a una función diferente dentro de la arquitectura de nube: **Cinder**, **Swift** y **Glance**. Aunque operan bajo un mismo ecosistema, sus objetivos, funcionamiento y modo de uso difieren considerablemente. Es fundamental comprender estas diferencias para diseñar correctamente una infraestructura estable, segura y escalable.

Cinder es el servicio de *Block Storage* o almacenamiento por bloques. Este tipo de almacenamiento es equivalente a un disco duro virtual conectado a una instancia. Los volúmenes creados en Cinder son persistentes, pueden desconectarse de una instancia para conectarse a otra y permiten crear copias de seguridad. Cinder es ideal para bases de datos, aplicaciones que manejan grandes cantidades de escritura, o cualquier servicio que requiera almacenamiento duradero y de alto rendimiento.

Swift, por otro lado, es un sistema de almacenamiento de objetos distribuido. A diferencia del almacenamiento por bloques, Swift no estructura la información en discos virtuales, sino en objetos almacenados dentro de contenedores. Su arquitectura distribuida permite una alta disponibilidad, escalabilidad horizontal y resistencia a fallos. Swift es ideal para almacenar archivos grandes, medios estáticos, copias de seguridad, información no estructurada y contenido web.

Glance es el servicio encargado de manejar imágenes de máquina virtual. Su propósito es almacenar, catalogar y distribuir imágenes que servirán para lanzar instancias. A diferencia de Cinder y Swift, Glance no está pensado para almacenar datos de usuario, sino imágenes de sistema operativo, snapshots y plantillas de máquinas. Glance puede usar diferentes backends, incluyendo almacenamiento local, Swift o Ceph.

Estas diferencias permiten a los administradores de OpenStack seleccionar el tipo de almacenamiento adecuado según la necesidad: persistencia transaccional (Cinder), contenido distribuido (Swift) o gestión de imágenes (Glance).

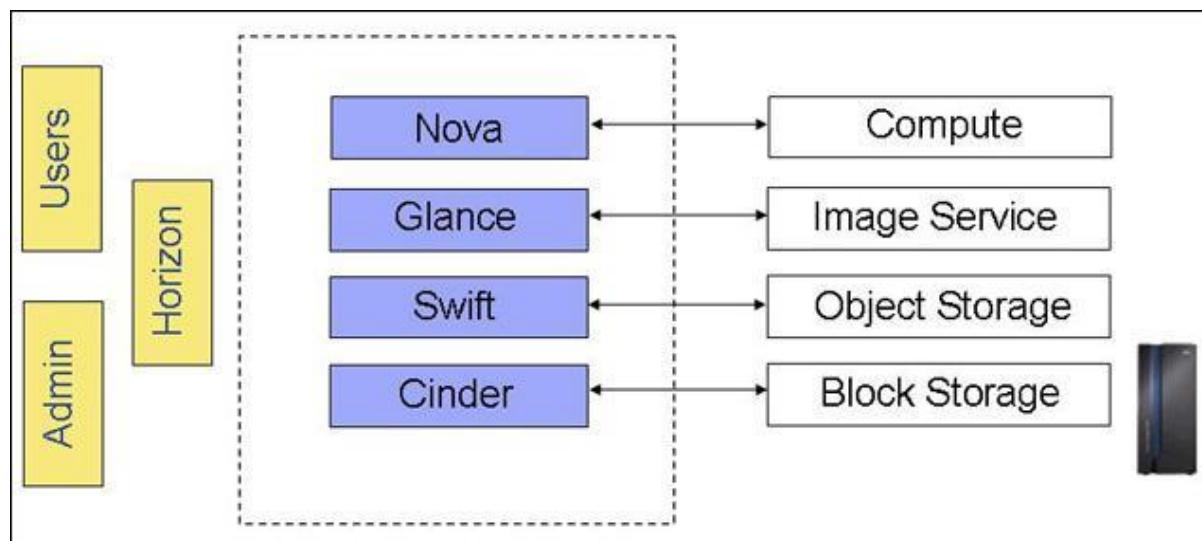


Fig 7.1 DS8870 en una infraestructura OpenStack

7.2 Backend de almacenamiento: LVM, Ceph, NFS

El almacenamiento en OpenStack puede operar sobre diferentes tipos de backend. Cada backend aporta capacidades distintas en cuanto a rendimiento, disponibilidad, escalabilidad y facilidad de administración. Los más utilizados son LVM, Ceph y NFS.

El backend LVM (Logical Volume Manager) es el más usado en instalaciones pequeñas o de laboratorio debido a su simpleza. LVM permite dividir un disco físico en volúmenes lógicos que Cinder puede usar como almacenamiento por bloques. Si bien es fácil de instalar y configurar, su escalabilidad es limitada y no provee replicación nativa, lo que lo hace más adecuado para entornos de prueba.

Ceph es considerado el backend más completo y robusto para OpenStack. Es un sistema de almacenamiento distribuido que ofrece almacenamiento por bloques, por objetos y por archivos. Ceph se integra perfectamente con Cinder, Swift y Glance, permitiendo crear una plataforma unificada de almacenamiento. Su arquitectura basada en replicación o codificación de borrado permite una alta disponibilidad y tolerancia a fallos. Es el backend recomendado para implementaciones en producción y nubes de gran escala.

El backend NFS (Network File System) es sencillo de implementar y permite compartir directorios a través de la red para su uso tanto por Cinder como por Glance. Aunque es fácil de administrar, su rendimiento depende del servidor NFS y su escalabilidad no es comparable con Ceph. Es útil para entornos donde la simplicidad es prioridad sobre la tolerancia a fallos.

Cada backend ofrece ventajas específicas, y la elección depende del tamaño de la infraestructura, el presupuesto, las necesidades de redundancia y el propósito de la instalación.

7.3 Snapshots, volúmenes y backups

El manejo de snapshots, volúmenes y copias de seguridad es fundamental para mantener la continuidad operativa en un entorno de nube. OpenStack proporciona herramientas integradas para gestionar estos elementos en los servicios Cinder y Glance.

Un snapshot es una captura del estado de un volumen o de una instancia en un momento específico. Los snapshots permiten crear nuevas instancias basadas en ese estado o restaurar una instancia existente. En Cinder, los snapshots se usan principalmente para volúmenes; en Glance se usan para generar imágenes a partir de máquinas en ejecución. Los snapshots son útiles para pruebas, actualizaciones de software o recuperación ante errores de usuario.

Los volúmenes son dispositivos de almacenamiento por bloques proporcionados por Cinder. Pueden conectarse, desconectarse y moverse entre instancias sin afectar su contenido. Los volúmenes pueden actuar como discos duros de alta persistencia y pueden escalarse según las necesidades del usuario. También pueden usarse para almacenar bases de datos y archivos de gran tamaño.

Los backups permiten generar copias externas de un volumen para almacenarlas en un backend secundario, normalmente Swift. A diferencia de los snapshots, que permanecen en el mismo backend, los backups permiten recuperar la información incluso en caso de fallo del nodo principal. Esto convierte a los backups en una herramienta esencial de continuidad operativa.

Una buena estrategia de almacenamiento incluye combinar snapshots frecuentes con backups periódicos y volúmenes dedicados para datos críticos.

7.4 Imágenes: creación y administración

El servicio Glance permite gestionar imágenes de máquinas virtuales, que son esenciales para desplegar instancias dentro del entorno OpenStack. Las imágenes pueden ser de diferentes formatos, como RAW, QCOW2, VMDK o ISO, dependiendo del tipo de sistema operativo y el hipervisor utilizado.

El proceso de creación de imágenes comienza con la preparación del sistema operativo base. Una vez optimizado, se empaqueta en un formato compatible con Glance. Posteriormente, el administrador la sube a Glance, donde se catalogan sus metadatos, como arquitectura, tamaño y estado. Estas imágenes pueden actualizarse, versionar o duplicarse para distintos usos.

La administración de imágenes consiste en asignar permisos, definir propiedades, optimizar tamaño, convertir formatos y mantener un inventario actualizado. Glance permite compartir

imágenes entre proyectos o mantenerlas privadas para grupos específicos. También soporta la creación de imágenes personalizadas basadas en instancias ya configuradas, permitiendo automatizar despliegues repetitivos.

Una adecuada organización del catálogo de imágenes garantiza un proceso de despliegue más rápido, eficiente y coherente.

7.5 Buenas prácticas para archivos, bases de datos y servidores web

Para mantener un entorno de almacenamiento optimizado en OpenStack, es necesario aplicar buenas prácticas que garanticen la eficiencia, el rendimiento y la seguridad de los datos. Estas recomendaciones abarcan archivos, bases de datos y servidores web.

Para el manejo de archivos, se recomienda separar los volúmenes según su propósito: logs, contenido multimedia, configuraciones y datos persistentes. Esto simplifica las copias de seguridad y evita que fallos en un directorio afecten toda la aplicación. También es recomendable habilitar compresión cuando el backend lo permita.

En cuanto a bases de datos, se deben almacenar en volúmenes de Cinder dedicados, preferiblemente en backends de alto rendimiento como Ceph. Deben realizarse snapshots frecuentes antes de actualizaciones y backups completos de forma periódica. Además, es aconsejable asignar QoS a los volúmenes para garantizar operaciones estables.

Para servidores web, se recomienda almacenar el contenido estático (imágenes, videos, documentos) en Swift, debido a su escalabilidad y tolerancia a fallos. El servidor web debe usar un volumen Cinder para configuraciones y archivos internos, pero el contenido público debe descargarse desde Swift para reducir carga y mejorar disponibilidad.

Estas prácticas garantizan mayor rendimiento general, facilidad de mantenimiento y reducción de fallos críticos.

Capítulo 8. Seguridad en OpenStack

8.1 Autenticación y autorización

OpenStack utiliza Keystone como su núcleo para la gestión de identidades. Keystone funciona como un directorio centralizado donde se autentican usuarios, se asignan permisos y

se emiten tokens. Este proceso es fundamental porque todos los servicios de OpenStack dependen de Keystone para validar quién accede al sistema y qué acciones puede realizar.

El mecanismo de autenticación se basa principalmente en tokens. Cuando un usuario ingresa sus credenciales, Keystone verifica esta información y genera un token temporal. Este token actúa como una “llave digital” que permite acceder a los demás servicios sin necesidad de reenviar la contraseña. Esta arquitectura reduce el riesgo de exposición de datos sensibles, ya que las contraseñas no se transmiten frecuentemente por la red.

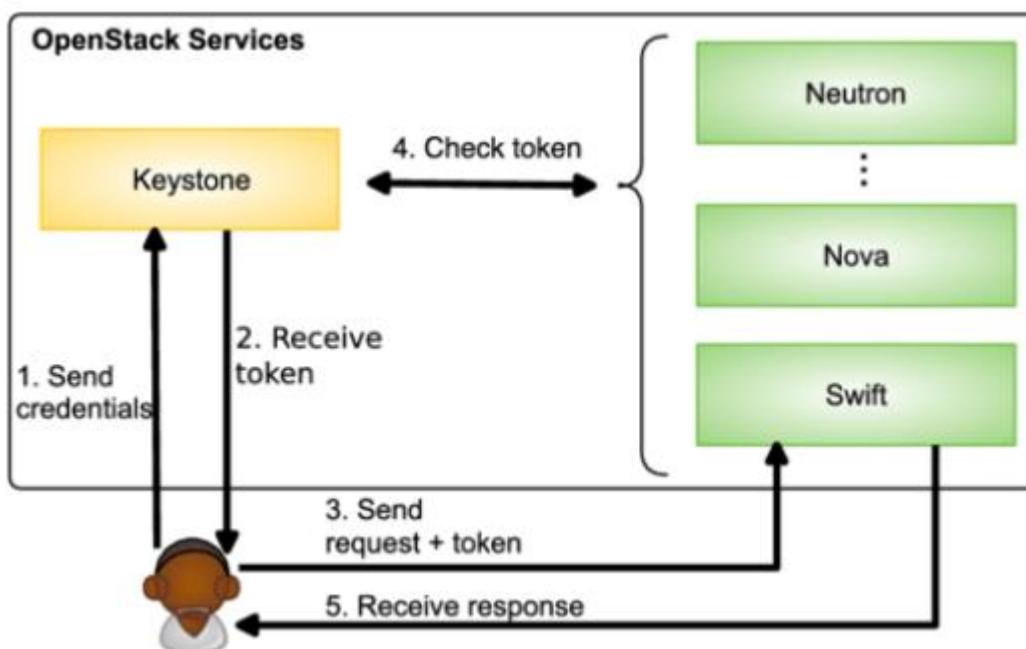


Fig. 8.1 Arquitectura Keystone

Keystone no solo maneja autenticación básica por usuario y contraseña; también permite integraciones corporativas. Por ejemplo, admite autenticación mediante LDAP, directorios Active Directory y federación con proveedores externos. Esto resulta esencial para organizaciones grandes que requieren control centralizado de su personal y auditoría del acceso.

La autorización, por otro lado, se vincula directamente a la asignación de roles dentro de proyectos específicos. Los roles determinan el nivel de poder que tiene un usuario sobre los recursos. Roles como *admin*, *member* o *reader* determinan qué acciones están permitidas. El principio de privilegios mínimos indica que un usuario debe tener exactamente los permisos necesarios, y nada más.

Para dar mayor claridad visual, en esta sección se recomienda incluir un diagrama que ilustre cómo Keystone maneja usuarios, roles y tokens.

8.2 Seguridad en la red

Uno de los aspectos más vulnerables de cualquier infraestructura en la nube es la red. OpenStack utiliza Neutron para gestionar redes virtuales, subredes, routers, reglas de seguridad y direcciones flotantes. La flexibilidad de Neutron permite construir redes de alta complejidad, pero también exige una arquitectura de seguridad sólida para evitar intrusiones.

Un componente central en este aspecto son los Security Groups. Estos actúan como un firewall interno asignado a cada instancia. Las reglas definidas en un Security Group controlan qué tráfico puede entrar o salir de una máquina virtual. Si estas reglas no se establecen correctamente, una instancia puede quedar completamente expuesta a ataques externos. Por ejemplo, permitir todos los puertos abiertos es una práctica extremadamente insegura; por el contrario, deben habilitarse solo los puertos estrictamente necesarios como SSH (22), HTTP (80) o HTTPS (443).

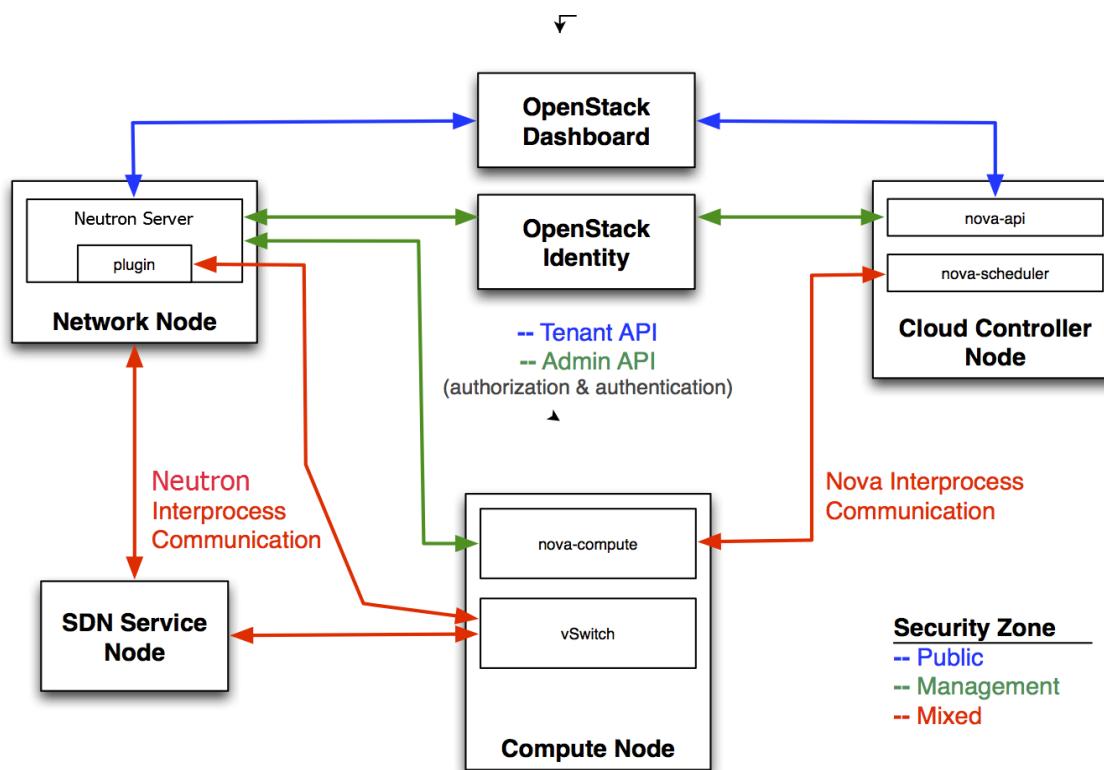


Fig 8.2 Diagrama de grupos de seguridad Neutron de OpenStack

Además del firewall a nivel de instancia, Neutron proporciona aislamiento de redes mediante VLAN, VXLAN o GRE. Estos métodos encapsulan el tráfico de los usuarios, evitando que un proyecto pueda interceptar información de otro. Este aislamiento es vital para entornos multi inquilino, como universidades o empresas que manejan múltiples áreas internas.

También es importante destacar las direcciones IP flotantes, que son la forma en que OpenStack expone instancias hacia el exterior. Las IP flotantes deben asignarse con cuidado, ya que representan el punto donde la nube privada interactúa con el internet público. Deben acompañarse de reglas de firewall restrictivas y monitoreo constante del flujo de paquetes.

Para mejorar la comprensión de estos conceptos, la sección debe incluir un esquema visual del flujo de tráfico en Neutron, mostrando routers, grupos de seguridad y redes internas.

8.3 Políticas de acceso a servicios

OpenStack incorpora mecanismos avanzados de control mediante el archivo de políticas conocido como policy.json. Cada uno de los servicios (Cinder, Nova, Glance, Neutron, etc.) tiene su propia configuración de políticas que definen qué acciones pueden realizar los usuarios según su rol.

Barbican reference deployment:
SafeNet Luna HSM (NTL is a SafeNet Protocol)
PKCS#11 Key Management Protocol
Best Practice Deployment

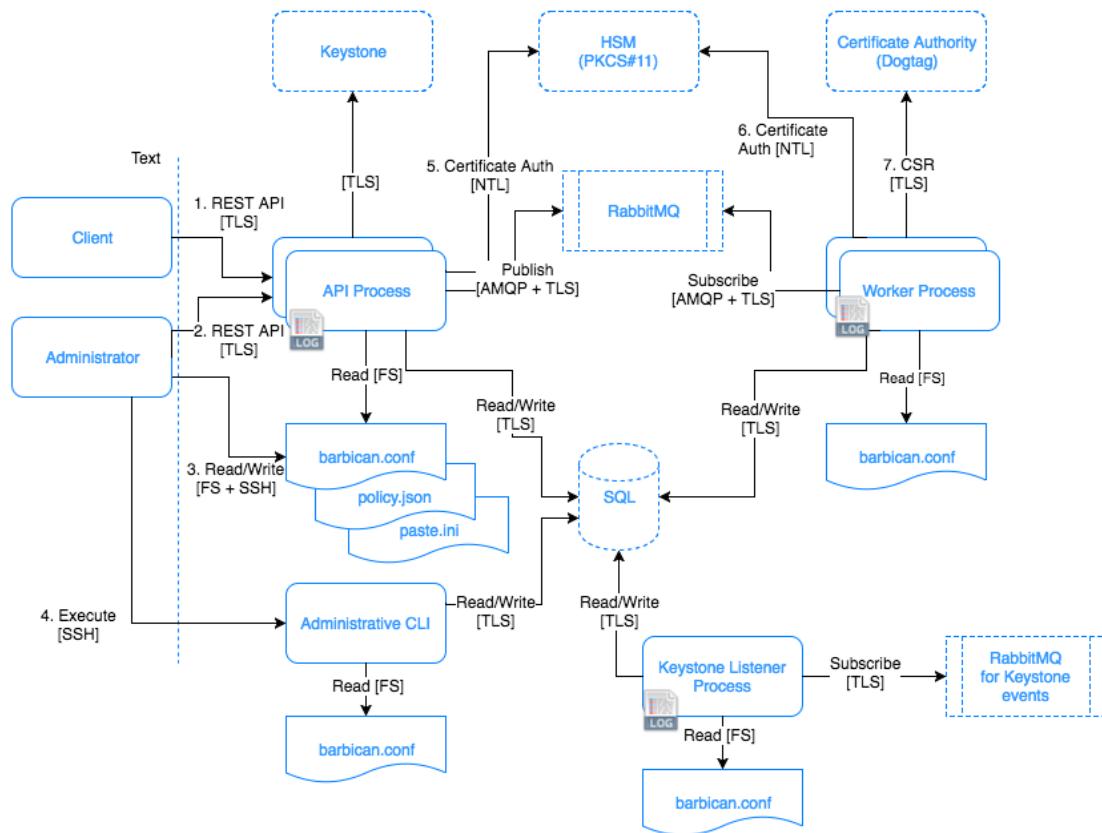


Fig 8.3 Diagrama de la arquitectura de servicio

Estas políticas funcionan como reglas internas que complementan el sistema de roles de Keystone. Permiten bloquear acciones críticas, incluso si un usuario tiene un rol relativamente elevado. Por ejemplo, es posible restringir la eliminación de instancias únicamente a administradores, o impedir que un usuario modifique reglas de red aunque sea miembro del proyecto.

Una característica importante del sistema de políticas es su granularidad. Las reglas pueden especificar permisos basados en información contextual, como el proyecto, el usuario o incluso la propiedad de un recurso. Esta flexibilidad hace que el sistema sea robusto, pero también requiere una administración cuidadosa. Un error en la configuración podría dejar a un servicio demasiado expuesto o inutilizable.

En versiones recientes, OpenStack está migrando hacia el modelo llamado **policy-in-code**, donde las políticas por defecto están integradas directamente en el software. Esto evita que errores de configuración en archivos externos rompan el sistema y mejora la seguridad general.

Para ofrecer una mejor visualización, es recomendable incluir un diagrama conceptual que muestre cómo funcionan las políticas desde Keystone hacia los demás servicios.

8.4 Firewalls, TLS, certificados

La seguridad no solo se basa en reglas de acceso y políticas internas; también requiere mecanismos de protección como firewalls, conexiones cifradas mediante TLS y el uso adecuado de certificados. Todos los servicios de OpenStack se comunican por API, por lo que es esencial que estas conexiones estén protegidas.

Los firewalls pueden estar configurados a nivel del sistema operativo, del hipervisor o a través de servicios específicos como *Firewall-as-a-Service (FWaaS)*. Este tipo de firewall permite controlar tráfico entre redes internas de OpenStack, evitando accesos no autorizados entre segmentos de red.

El uso de TLS garantiza que las comunicaciones entre clientes, paneles de administración (Horizon) y APIs estén cifradas. Esto es especialmente importante cuando los administradores gestionan OpenStack de forma remota. Sin TLS, un atacante podría interceptar tokens, contraseñas o incluso manipular solicitudes.

La administración de certificados requiere especial atención. Deben generarse, renovarse y almacenarse de forma segura. Un certificado expirado puede bloquear el acceso a los servicios, mientras que un certificado comprometido puede permitir ataques de suplantación. Para este propósito, OpenStack ofrece Barbican, un servicio diseñado específicamente para gestionar secretos, llaves y certificados de manera segura dentro de la nube.

8.5 Buenas prácticas para la nube privada

Para asegurar de manera efectiva un entorno OpenStack, es necesario seguir buenas prácticas que abarquen todos los niveles de la arquitectura. La primera recomendación es aplicar el

principio de privilegios mínimos, lo cual implica otorgar a cada usuario únicamente los permisos necesarios para desempeñar su función. Esto reduce la probabilidad de que una cuenta comprometida cause un daño significativo.

Otra práctica esencial es implementar un sistema de monitoreo, tanto para servicios internos como para la red. Herramientas como Prometheus, Grafana o Elastic Stack pueden integrarse para recolectar métricas de seguridad, analizar logs y generar alertas en tiempo real. Una nube sin monitoreo constante es vulnerable a ataques prolongados que pasan desapercibidos.

Las actualizaciones periódicas también son cruciales. Los servicios de OpenStack reciben parches de seguridad con frecuencia, por lo que aplicar estas actualizaciones reduce significativamente el riesgo de explotación de vulnerabilidades conocidas. Del mismo modo, los sistemas operativos de los nodos deben mantenerse al día con parches y paquetes actualizados.

El uso de redes aisladas, firewalls estrictos, certificados válidos y mecanismos de copia de seguridad componen la base técnica de un entorno seguro. Sin embargo, también es vital capacitar al personal, documentar los procedimientos internos y auditar el acceso de forma periódica. La seguridad no es solo tecnológica, sino también organizacional.

Capítulo 9. Casos de uso reales de OpenStack

Este capítulo presenta aplicaciones reales de OpenStack en el entorno empresarial, comparaciones con nubes públicas comerciales y análisis sobre por qué la plataforma es una alternativa sólida para desplegar servicios web, arquitecturas escalables y entornos modernos basados en microservicios o contenedores. Se incluyen además recomendaciones prácticas y ejemplos para comprender cómo OpenStack se implementa en escenarios de producción.

9.1 Empresas que utilizan OpenStack

OpenStack es utilizado por organizaciones de diversos sectores que requieren controlar sus recursos, gestionar grandes volúmenes de datos y mantener independencia de proveedores externos. Empresas de telecomunicaciones, bancos, universidades y compañías tecnológicas han adoptado OpenStack para construir nubes privadas y públicas, optimizar costos y aumentar la seguridad de sus operaciones.



Fig 9.1 Principales empresas de servicios OpenStack

Algunas organizaciones que utilizan OpenStack incluyen:

- Empresas de telecomunicaciones que requieren una plataforma estable y escalable para alojar servicios de red, infraestructura virtualizada y aplicaciones de misión crítica.
- Instituciones financieras que buscan un mayor control sobre su infraestructura, especialmente en entornos donde la regulación exige soberanía de datos.
- Universidades y centros de investigación que utilizan OpenStack para laboratorios virtuales, simulaciones científicas y plataformas de cómputo de alto rendimiento.
- Empresas tecnológicas que necesitan flexibilidad para automatizar el despliegue de servicios, escalar rápidamente y adoptar metodologías DevOps.

La adopción de OpenStack se debe principalmente al equilibrio entre control, flexibilidad y reducción de costos operativos, lo que lo convierte en una alternativa atractiva frente a proveedores de nube pública.

9.2 Comparación con AWS, Azure y Google Cloud

OpenStack no compite directamente con los proveedores de nube pública, sino que ofrece un enfoque alternativo: la capacidad de construir una nube privada totalmente personalizable, con niveles más altos de control sobre el hardware, los datos y la seguridad. Existen diferencias clave que deben considerarse al seleccionar la plataforma adecuada.

				
LAUNCHING YEAR	2004	2010	2008	
AVAILABILITY	84 availability zone & 24 Geographical locations	60+ region across all over the country	24 region & 74 total Zones	
SERVICES	212+ SERVICES	200+ SERVICES	100+ Services	
CLOUD SHARE	33% of the Market	21% of the Market	8% of the Market	
COMPUTE ENGINE	EC2 (Elastic Compute System)	Virtual Machine	Compute Engine	
NETWORKING	VIRTUAL PRIVATE CLOUD	VIRTUAL NETWORK (VNET)	CLOUD VIRTUAL NETWORK	
CLIENTS	Netflix, BMW, Samsung, Unilever, Expedia	HP, Apple, Polycom, Honeywell, Johnson	LG, Toyota, Vodafone, Spotify, New York Times	
PRICING	Based on Charge per hour	Based on Charge per minute	Based on Charge per minute	

Fig 9.2 Comparación OpenStack vs AWS vs Azure vs Google Cloud

Ventajas de OpenStack

- Control total sobre la infraestructura física y virtual.
- Costo más predecible, sin tarifas variables mensuales por consumo.
- Ideal para organizaciones que requieren privacidad, aislamiento o cumplimiento normativo estricto.
- Extensible y adaptable mediante módulos y servicios internos.

Ventajas de las nubes públicas (AWS, Azure, GCP)

- Disponibilidad global sin necesidad de administrar hardware.
- Servicios administrados avanzados como bases de datos, inteligencia artificial y machine learning preconstruidos.
- Alta tolerancia a fallos garantizada por el proveedor.
- Modelo de pago por uso que facilita escalar sin inversión inicial.

En proyectos donde la privacidad, el control o la reducción de costos en el largo plazo son prioritarios, OpenStack se posiciona como una opción sólida. Por el contrario, si se requiere escalabilidad global inmediata o servicios avanzados ya preconfigurados, las nubes públicas suelen ser más convenientes.

9.3 Ventajas de crear un servidor web en OpenStack

Implementar un servidor web dentro de OpenStack ofrece beneficios significativos en términos de rendimiento, seguridad y adaptabilidad. La capacidad de ajustar el tamaño de las instancias, añadir平衡adores de carga o replicar configuraciones permite construir un servicio optimizado para producción.

Las ventajas clave al crear un servidor web en OpenStack son:

- **Aislamiento total:** Cada instancia se ejecuta en un entorno independiente, lo que reduce riesgos de seguridad y conflictos entre aplicaciones.
- **Flexible asignación de recursos:** CPU, RAM y almacenamiento pueden ampliarse según el crecimiento del sitio web o aplicación.
- **Automatización de despliegues:** Herramientas como Heat o Ansible permiten crear plantillas reproducibles para servidores web robustos.
- **Redes personalizadas:** El administrador puede crear subredes internas, redes públicas, floating IPs y reglas de seguridad específicas.
- **Escalabilidad local:** Si el tráfico aumenta, se pueden lanzar nuevas instancias rápidamente sin depender de recursos externos.

Esta capacidad hace que OpenStack sea ideal para organizaciones que desean manejar servicios críticos sin renunciar al control sobre su infraestructura.

9.4 Escalabilidad y tolerancia a fallos

OpenStack ofrece una arquitectura altamente modular que permite construir entornos escalables y tolerantes a fallos, lo cual es esencial para aplicaciones de misión crítica o servicios que requieren alta disponibilidad. Gracias a su diseño distribuido, los servicios principales como Nova (cómputo), Neutron (redes) y Cinder (almacenamiento) pueden desplegarse en múltiples nodos para asegurar que, si uno de ellos falla, los demás continúen ejecutándose sin interrupciones. Esto reduce notablemente los tiempos de caída y garantiza que los usuarios puedan acceder a sus recursos de manera continua. Además, la plataforma permite la escalabilidad horizontal, de modo que si la demanda de procesamiento o almacenamiento aumenta, es posible añadir más nodos al clúster sin afectar los servicios en ejecución. Esta flexibilidad es especialmente útil en organizaciones donde el tráfico o la carga de trabajo varía constantemente.

Otra ventaja importante es la capacidad de implementar balanceadores de carga que distribuyen solicitudes entre diferentes instancias, lo que no solo mejora el rendimiento general de las aplicaciones, sino que también evita que un único servidor se convierta en un punto de fallo. En entornos de almacenamiento, OpenStack puede integrarse con soluciones distribuidas como Ceph, que replican automáticamente los datos en múltiples discos y nodos, aumentando la resiliencia ante pérdidas de hardware. Asimismo, OpenStack permite la recuperación automática de instancias: si un host físico falla, el sistema puede reiniciar sus máquinas virtuales en otro nodo disponible sin intervención manual. Este conjunto de características convierte a OpenStack en una opción confiable para la construcción de infraestructuras robustas, capaces de mantenerse operativas incluso ante fallas imprevistas.

Open vSwitch - High-availability with VRRP Components and Connectivity

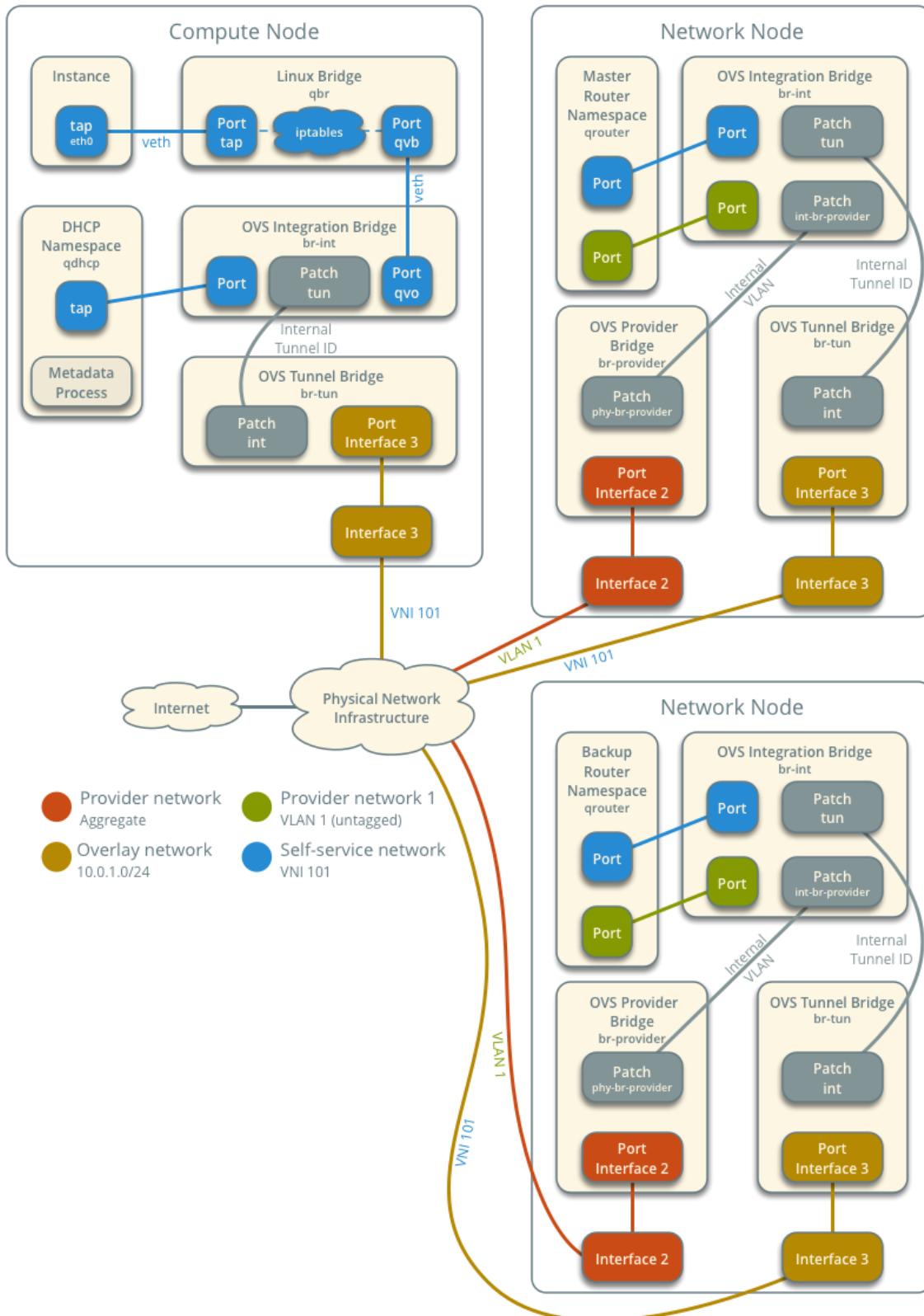


Fig 9.4 Arquitectura de alta disponibilidad OpenStack

9.5 Microservicios, contenedores y Kubernetes

OpenStack se ha convertido en una plataforma ideal para entornos modernos basados en microservicios y contenedores, gracias a su capacidad de integrarse de forma nativa con tecnologías como Docker y Kubernetes. A través del servicio Magnum, OpenStack permite desplegar clústeres completos de Kubernetes, Docker Swarm o Mesos directamente sobre instancias virtuales, convirtiendo la nube privada en una base sólida para ejecutar arquitecturas distribuidas. Esto facilita que los desarrolladores creen aplicaciones modulares, escalables y fáciles de mantener, aprovechando al mismo tiempo todas las ventajas de la virtualización y el control de red que ofrece OpenStack. Además, al poder asignar volúmenes persistentes de Cinder a pods dentro de Kubernetes, se garantiza que los datos esenciales permanezcan almacenados incluso si los contenedores se reemplazan o se escalan.

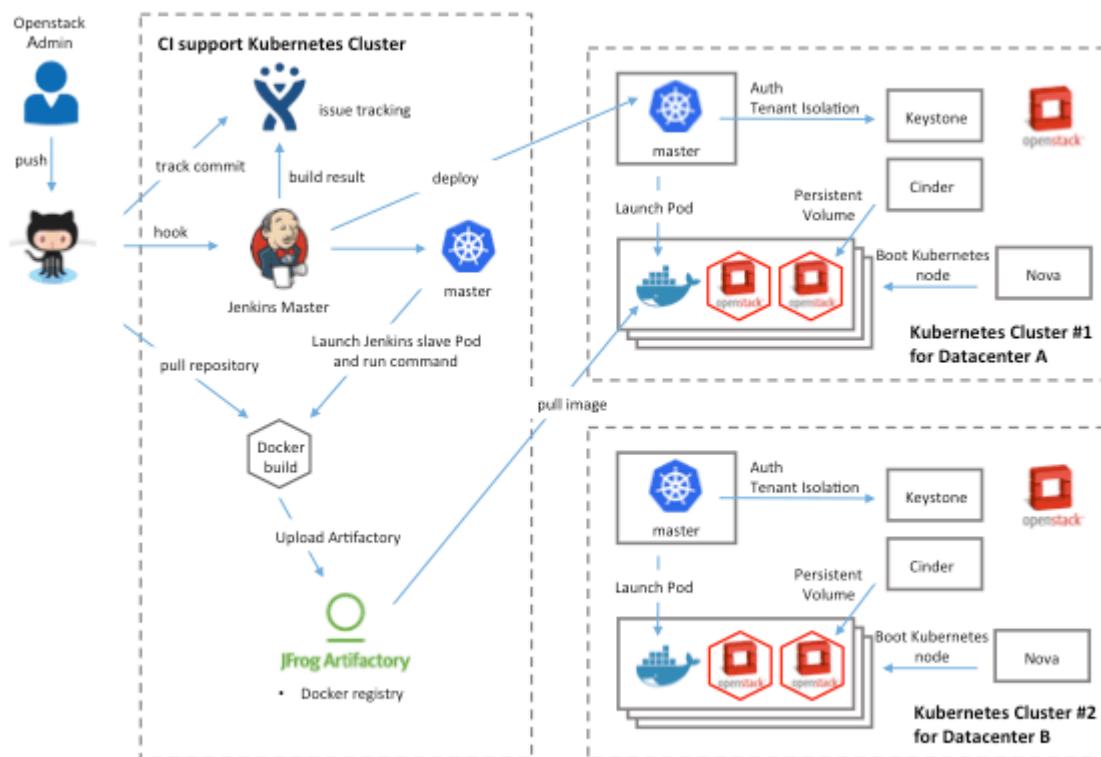


Fig 9.5.1 Diagrama de integración Kubernetes Openstack

En términos de redes, Neutron proporciona un entorno adaptable para conectar contenedores con instancias, redes internas, redes externas y balanceadores de carga. Esto permite construir

aplicaciones híbridas donde conviven máquinas virtuales y microservicios en un mismo entorno, manteniendo una comunicación estable y segura entre todos los componentes.

OpenStack también se utiliza frecuentemente como plataforma subyacente para nubes híbridas y arquitecturas más complejas de orquestación, ya que su flexibilidad permite conectar diferentes entornos sin sacrificar rendimiento. Gracias a estas capacidades, OpenStack no solo soporta aplicaciones nativas de la nube, sino que también impulsa la adopción de metodologías DevOps, automatización avanzada y despliegues continuos en entornos empresariales modernos.

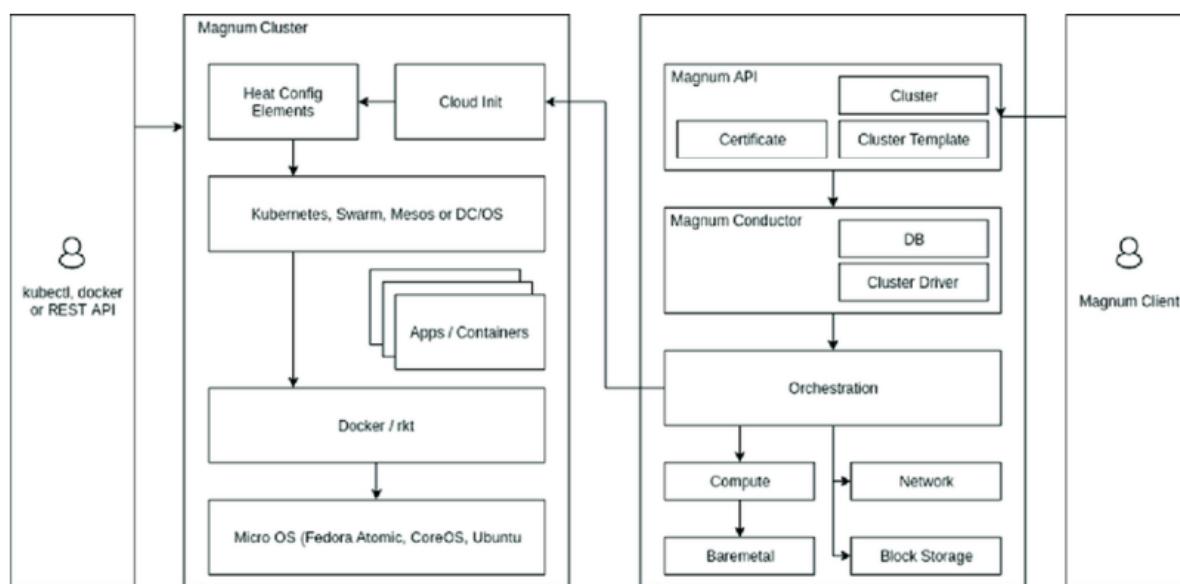


Fig 9.5.2 Diagrama Magnum Openstack

Capítulo 10. Preparación del entorno

La preparación del entorno es una de las fases más importantes antes de realizar la instalación de OpenStack mediante DevStack. Un entorno correctamente configurado garantiza estabilidad, rendimiento y evita la mayoría de errores comunes durante el deployment. En este capítulo se explican los requisitos necesarios para la máquina virtual, la descarga de la imagen base del sistema operativo, la configuración inicial, los ajustes de red y la instalación de dependencias previas. Este proceso se realizará utilizando Oracle VirtualBox como hipervisor, Kubuntu 24.04 LTS como sistema operativo base, y DevStack como método de instalación, ejecutado sobre una interfaz adaptador puente y una conexión física Ethernet para evitar problemas de conectividad entre servicios internos de OpenStack.

10.1 Requisitos de la máquina virtual

Antes de comenzar la instalación, es fundamental asegurarse de que el entorno virtual cumple con los requisitos recomendados para ejecutar OpenStack de manera estable. DevStack es una herramienta que facilita la instalación de una nube OpenStack completa en un solo nodo, pero aun así requiere recursos considerables debido a los distintos servicios que se desplegarán (Nova, Neutron, Glance, Cinder, Keystone, entre otros).

Se recomienda que la máquina virtual en VirtualBox cuente con al menos 4 vCPUs asignadas. Aunque es posible que DevStack funcione con menos núcleos, ejecutar múltiples servicios simultáneamente puede generar lentitud e incluso fallos en la instalación si no se dispone de capacidad computacional suficiente. Respecto a la memoria RAM, se sugiere un mínimo de 8 GB, siendo 10 o 12 GB lo ideal para un desempeño fluido. Esto permite que los servicios permanezcan activos sin que el sistema operativo comience a utilizar memoria de intercambio, circunstancia que ralentiza drásticamente el funcionamiento de OpenStack. En cuanto a almacenamiento, se recomienda asignar un disco virtual de 40 GB o más, preferiblemente en formato VDI con asignación dinámica para optimizar el uso real del espacio. Este almacenamiento será necesario para las imágenes de máquinas virtuales, logs, paquetes de instalación y volúmenes internos utilizados por DevStack.

Capítulo 11. Instalación de la máquina virtual (Kubuntu 24.04.3 (Noble Numbat))

Damos clic en el siguiente enlace para descargar la imagen:

<https://cdimage.ubuntu.com/kubuntu/releases/noble/release/kubuntu-24.04.3-desktop-amd64.iso>

Una vez terminada la descarga abrimos virtualbox y damos clic en Nueva para crear una máquina virtual nueva [10].

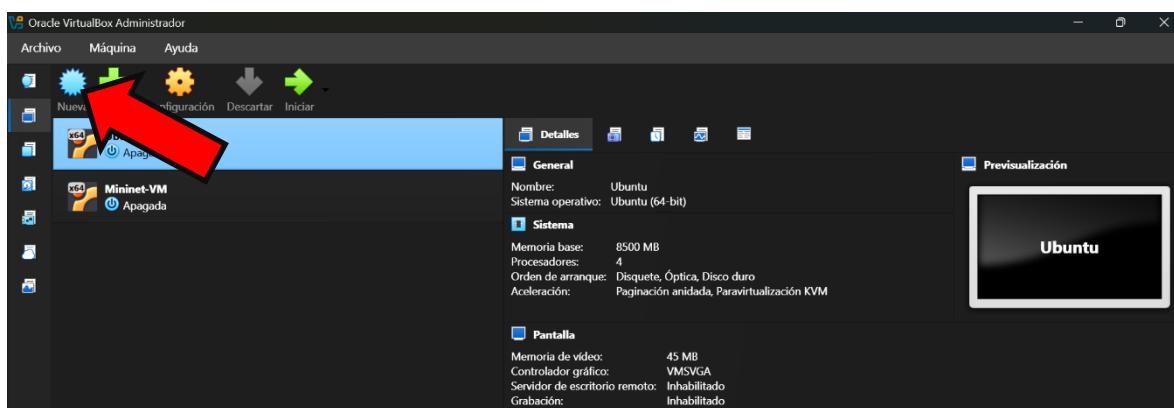


Figura 10: Creación de máquina virtual.

Luego se nos abrirá una ventana donde pondremos el nombre de nuestra máquina virtual y la ruta donde se descargó nuestra imagen de Kubuntu, adicional quitamos la opción que dice “Proceed with Unattended Installation” con el fin de poder especificar los recursos de nuestro hardware que va a utilizar la máquina virtual [11].

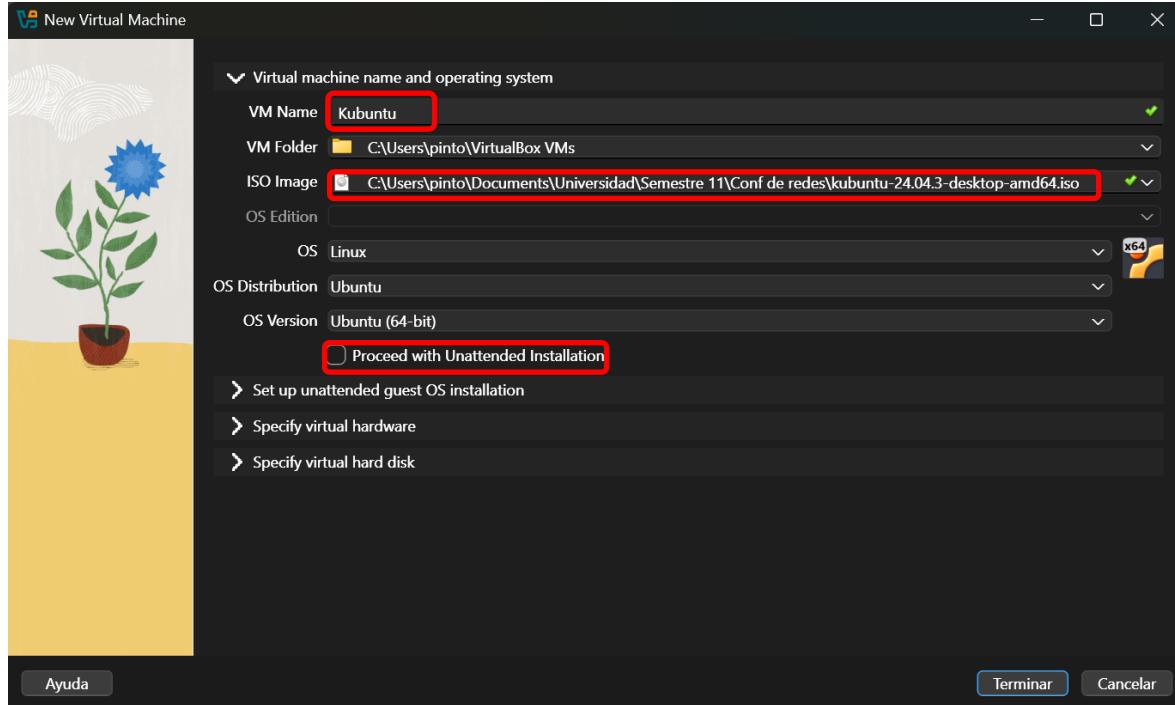


Figura 11: Configuración de nuestra máquina virtual.

Desplegamos el apartado “Specify virtual hardware” donde le asignaremos los recursos de nuestra computadora que van a ser usado para correr la máquina virtual [12].

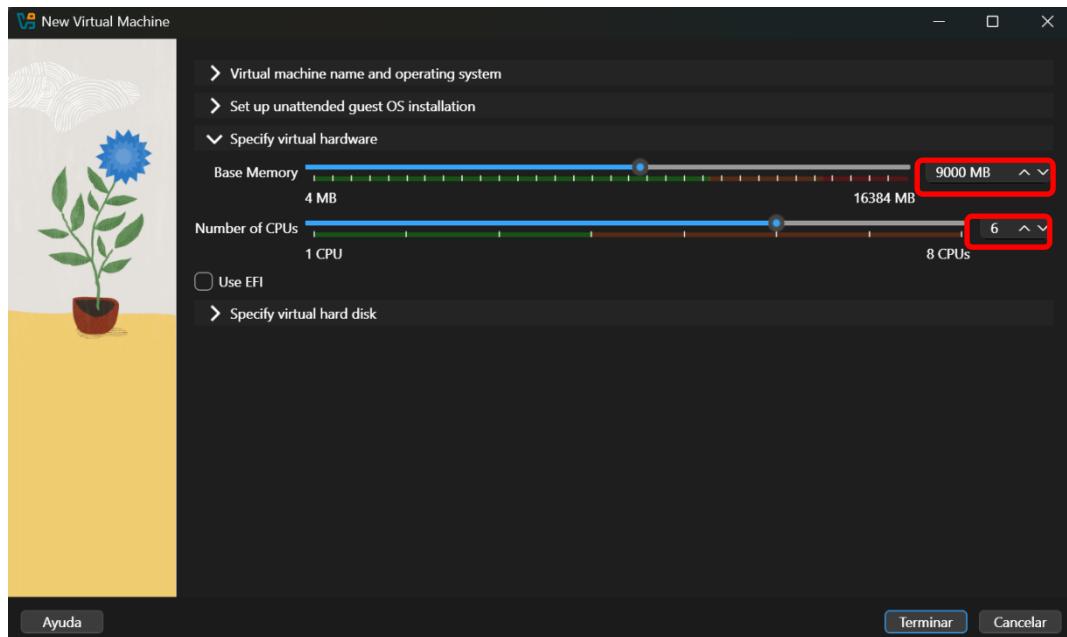


Figura 12: Asignación de hardware.

Importante: Si queremos obtener el mejor desempeño posible la mejor opción es instalar kubuntu en nuestro equipo de manera normal, así obtendremos el mejor rendimiento posible y acortaremos tiempos de ejecución, a partir de que terminemos con la instalación de la VM las siguientes capturas de pantalla serán tomadas de Kubuntu instalado directamente en nuestro pc, pero no te preocupes, puedes seguir usando tu máquina virtual, solo debes ser paciente y seguir todos los pasos.

Desplegamos el apartado “Specify virtual hard disk” donde le asignaremos el tamaño de almacenamiento con el que va a contar la máquina virtual y damos clic en terminar [13].

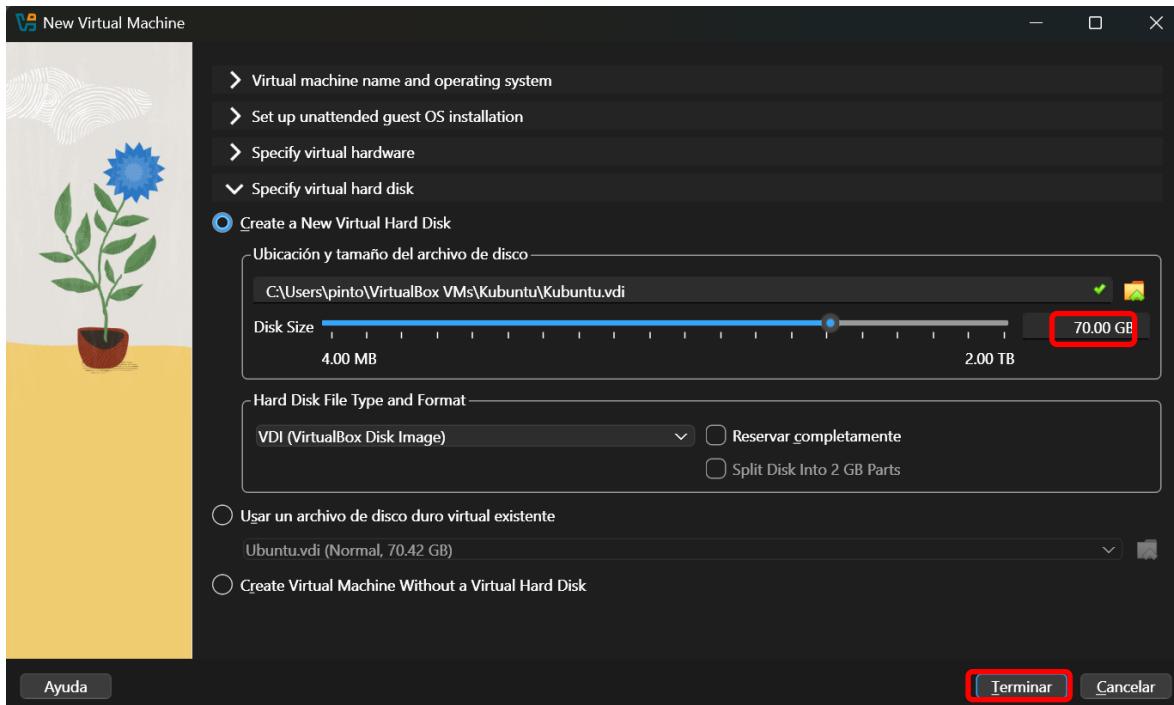


Figura 13: Asignación de almacenamiento.

Se nos agregará nuestra máquina virtual, damos clic derecho sobre ella y luego clic en Configuración [14].

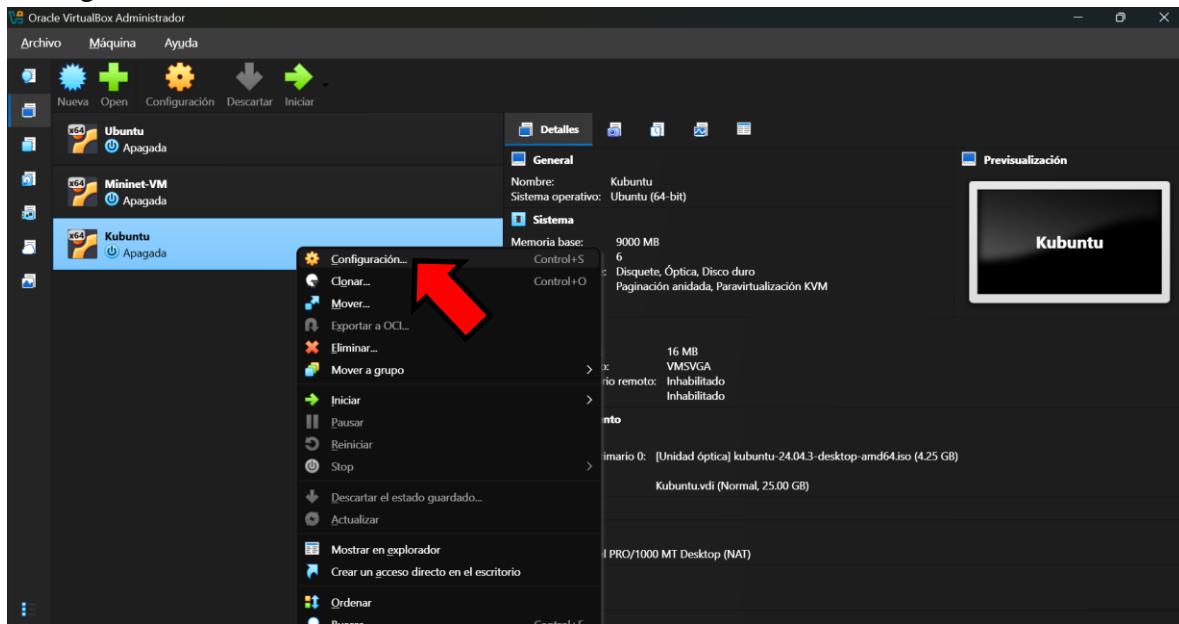


Figura 14: Configuración de la VM.

Accedemos al apartado Red, en “Conectar a:” seleccionamos la opción Adaptador puente, luego damos clic en Aceptar [15].

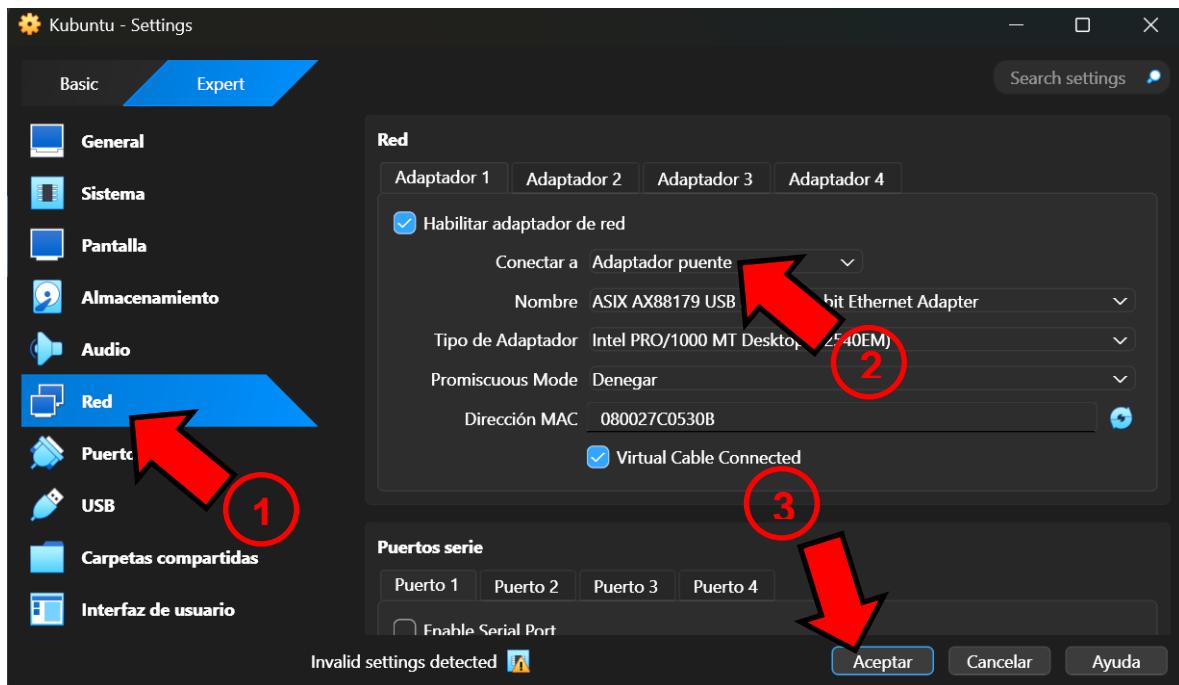


Figura 15: Configuración de la Red.

Clic en iniciar para arrancar nuestra VM [16].

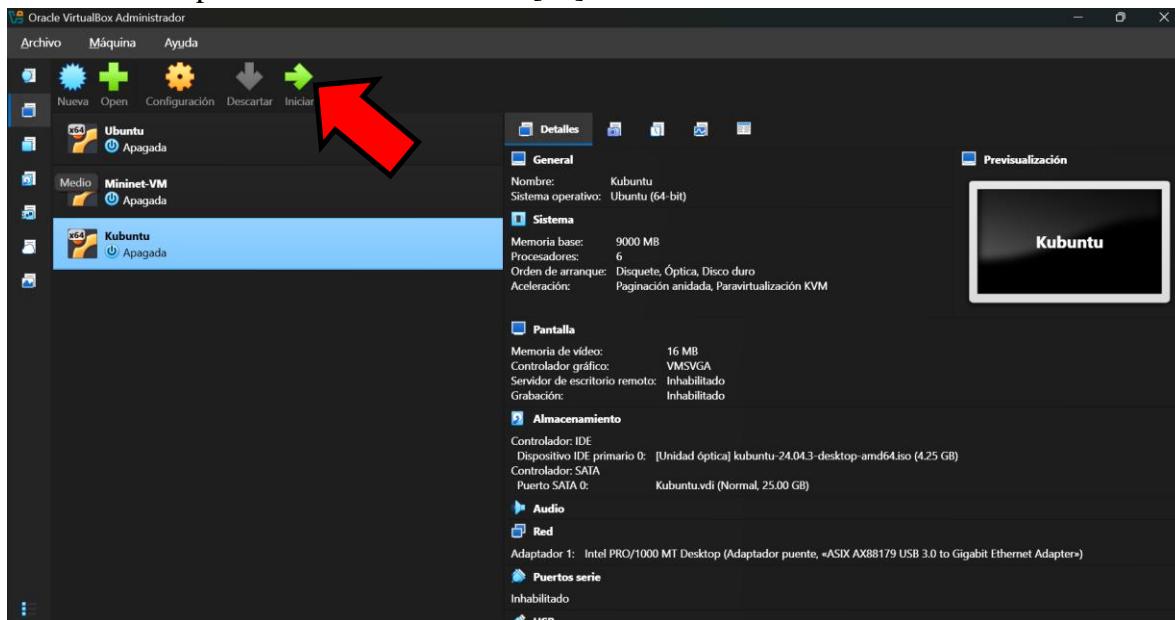


Figura 16: Iniciar máquina virtual.

Una vez inicia con esta opción resaltada en gris presionamos la tecla Enter [17]

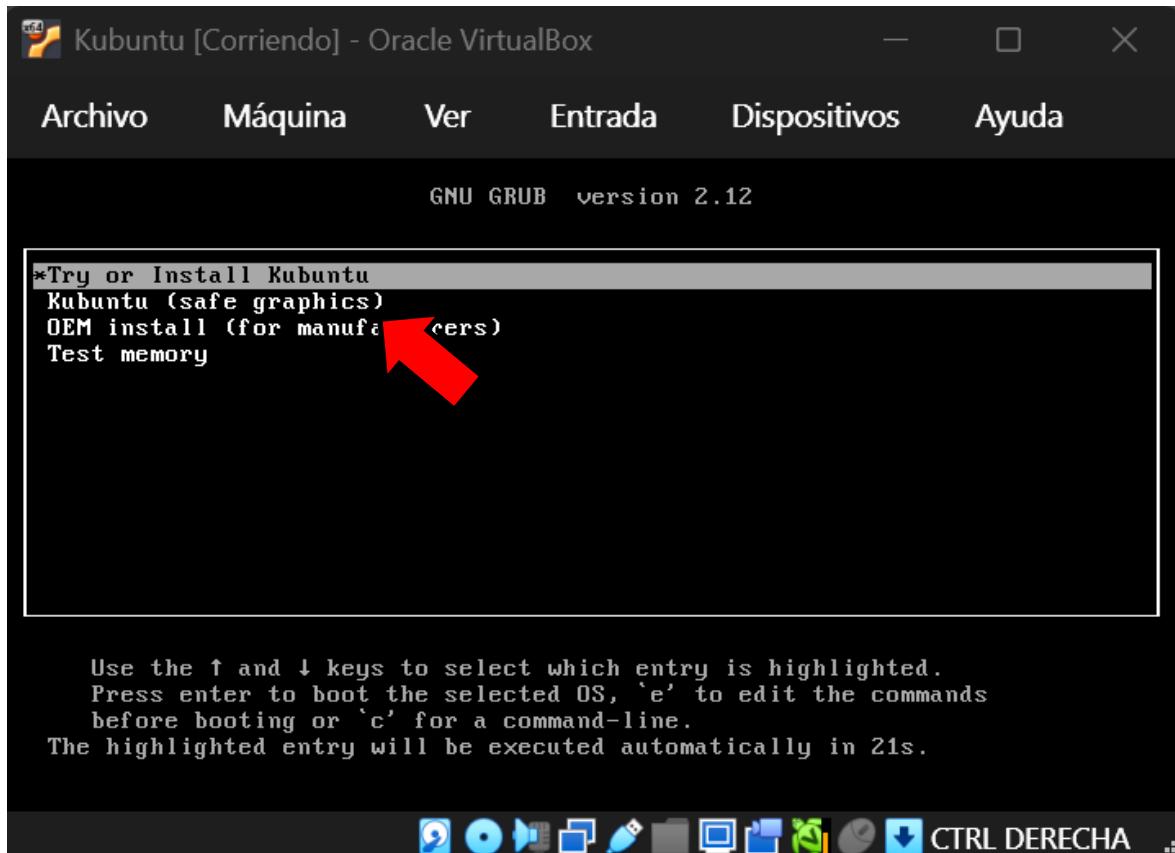


Figura 17: Instalar kubuntu.

Seleccionamos el lenguaje español y damos clic en Install Kubuntu [18]

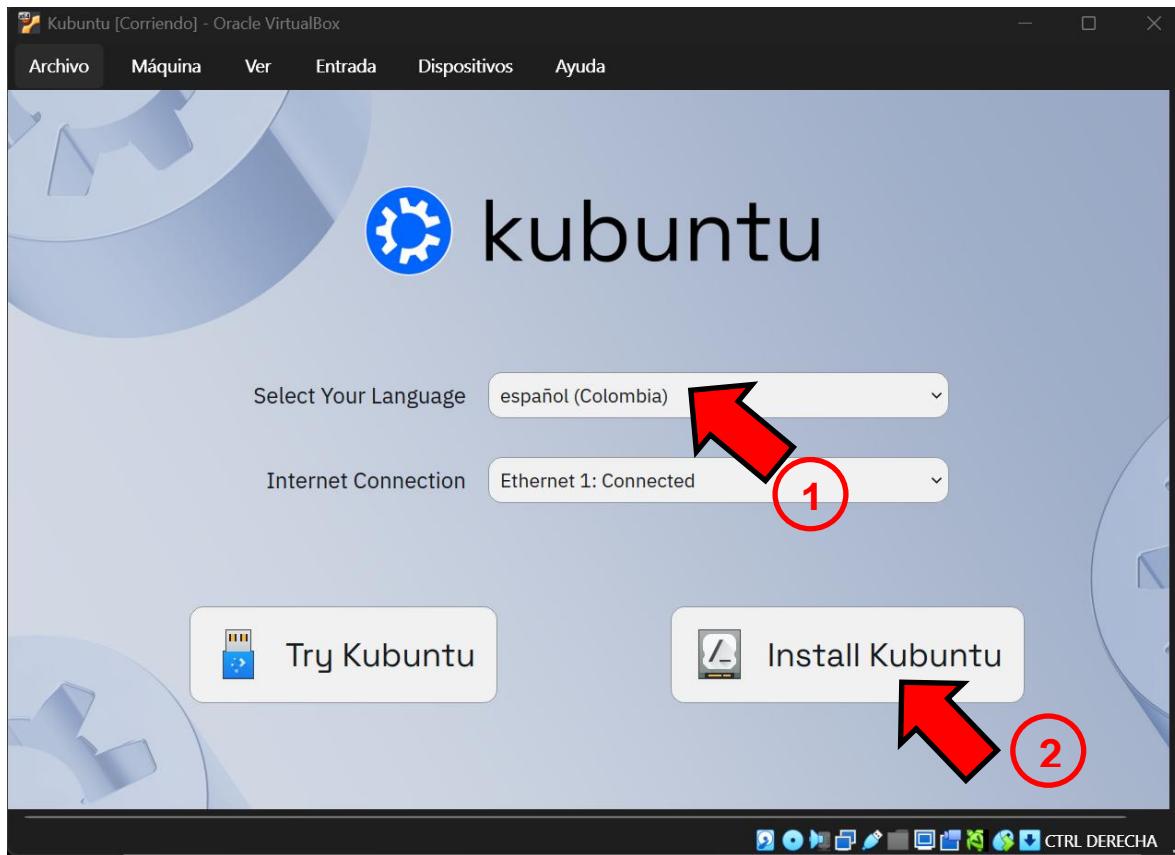


Figura 18: Selección de idioma.

Seleccionamos nuestra región horaria y damos clic en siguiente [19].

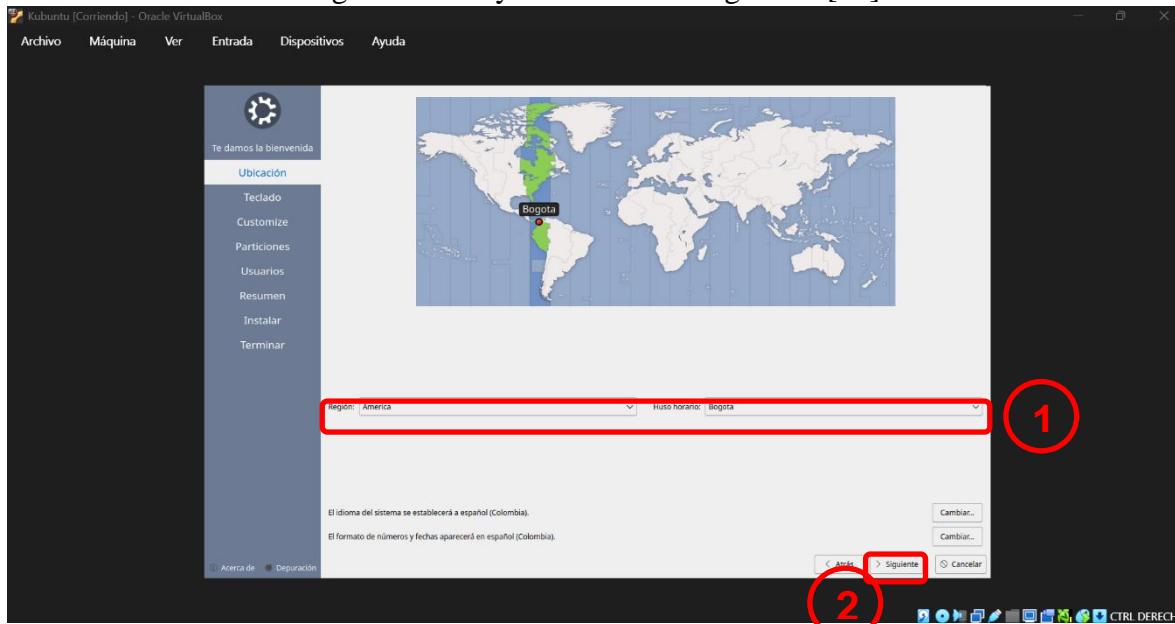


Figura 19: Selección de zona horaria.

Seleccionamos nuestra distribución de teclado y damos clic en siguiente [20].

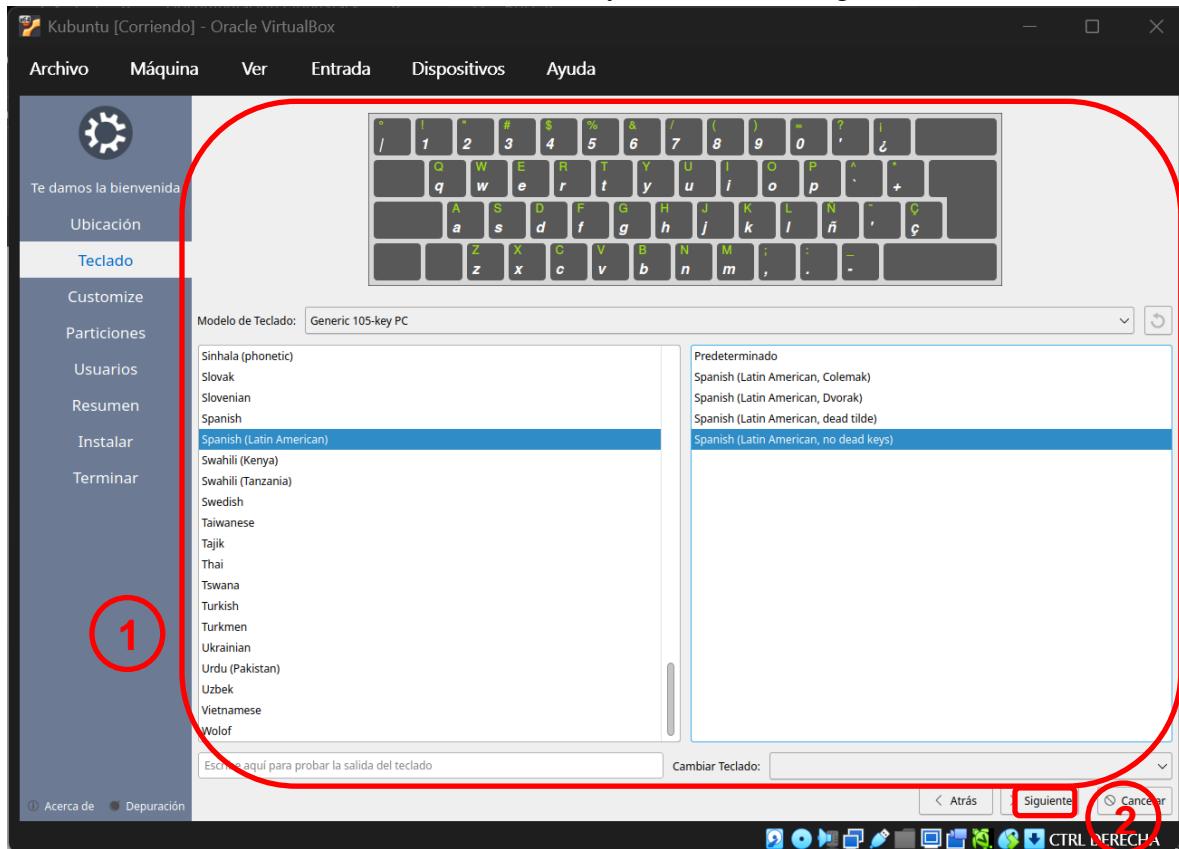


Figura 20: Selección de distribución de teclado.

Clic en siguiente [21]

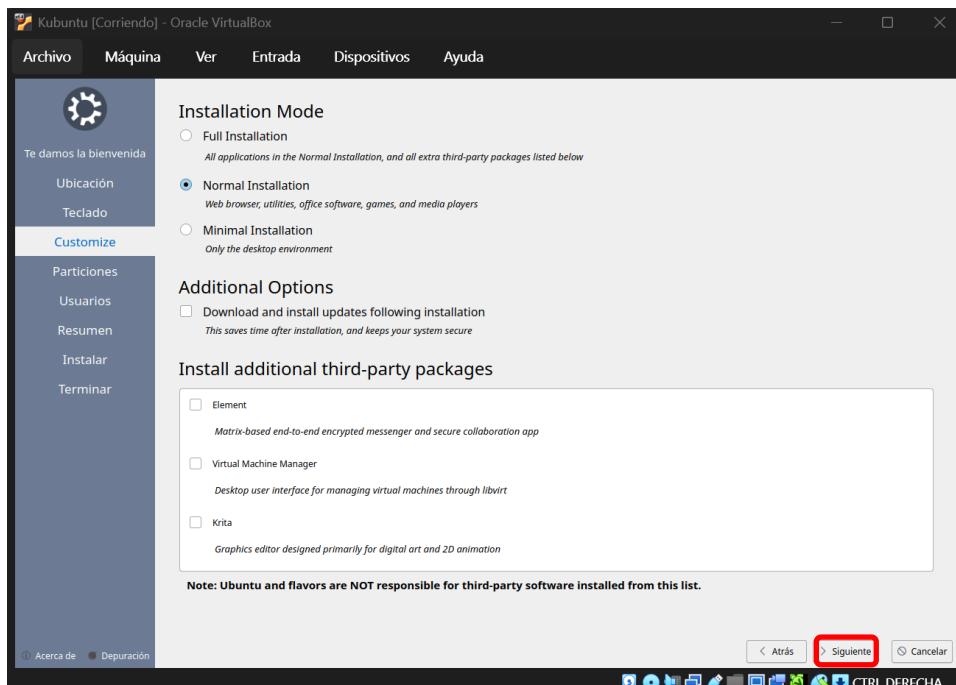


Figura 21: Modo de instalación.

Damos clic en borrar disco y luego en siguiente [22]

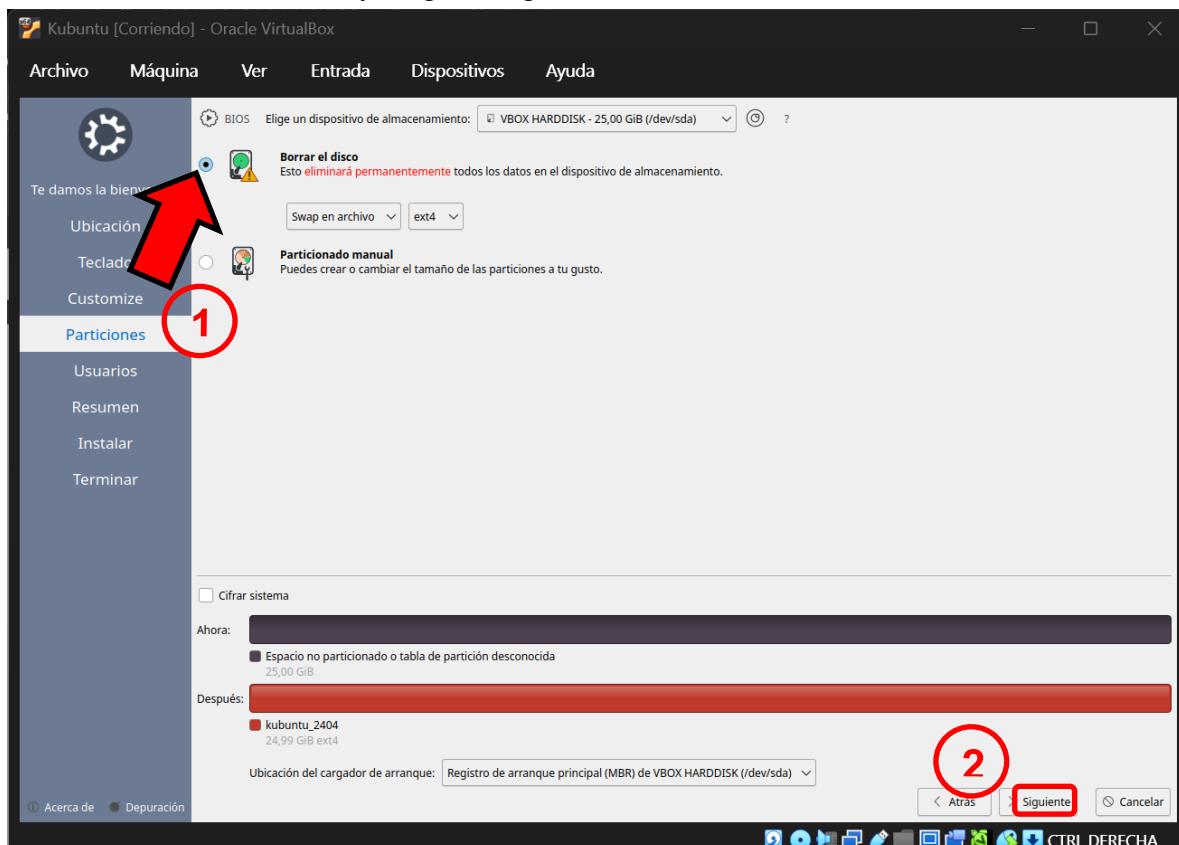


Figura 22: Seleccionar disco.

Configuramos el usuario y la contraseña y damos clic en siguiente [23]

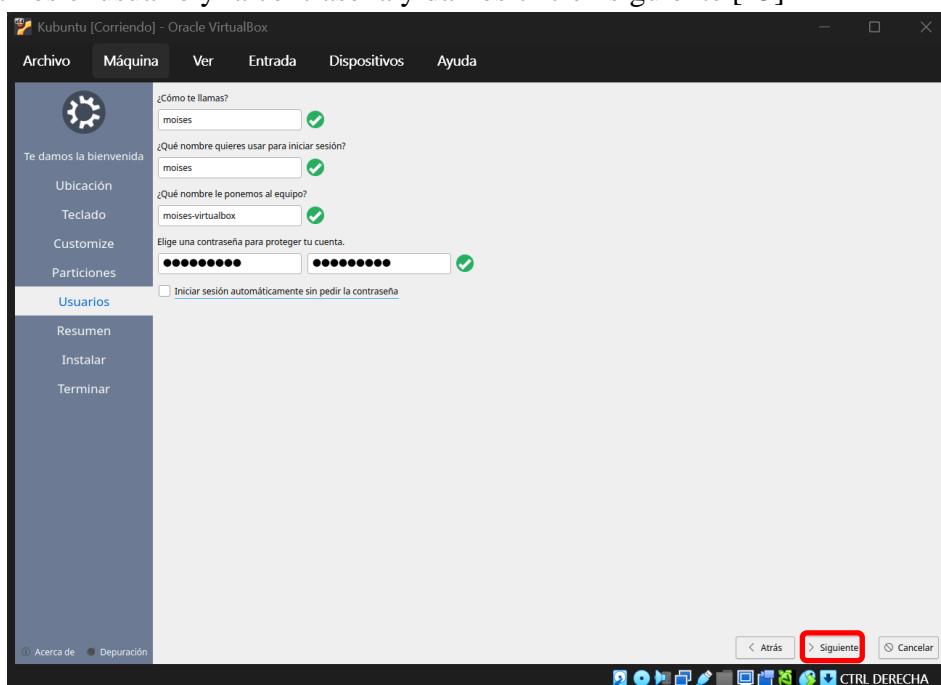


Figura 23: Creación de usuario.

Clic en instalar y luego en Instalar ahora [24]

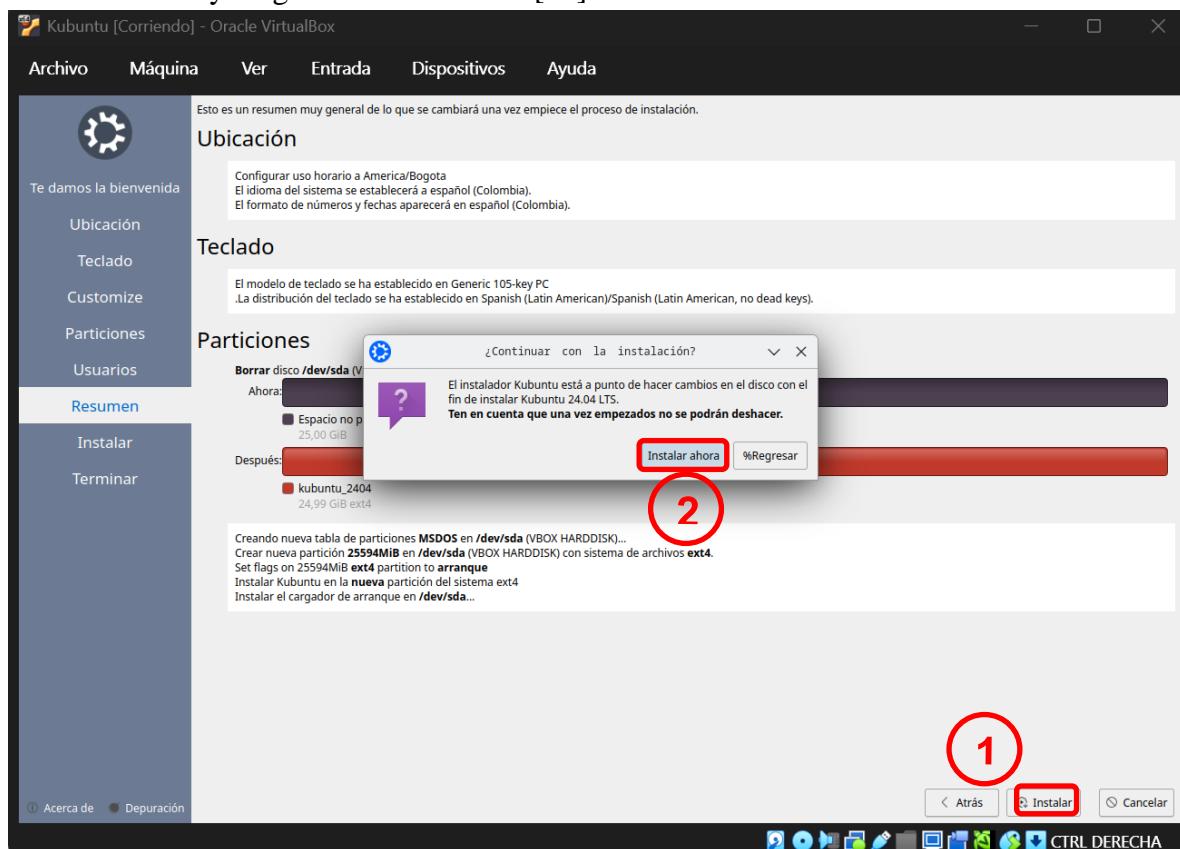


Figura 24: Confirmar Instalación.

Esperamos a que finalice la instalación [25].

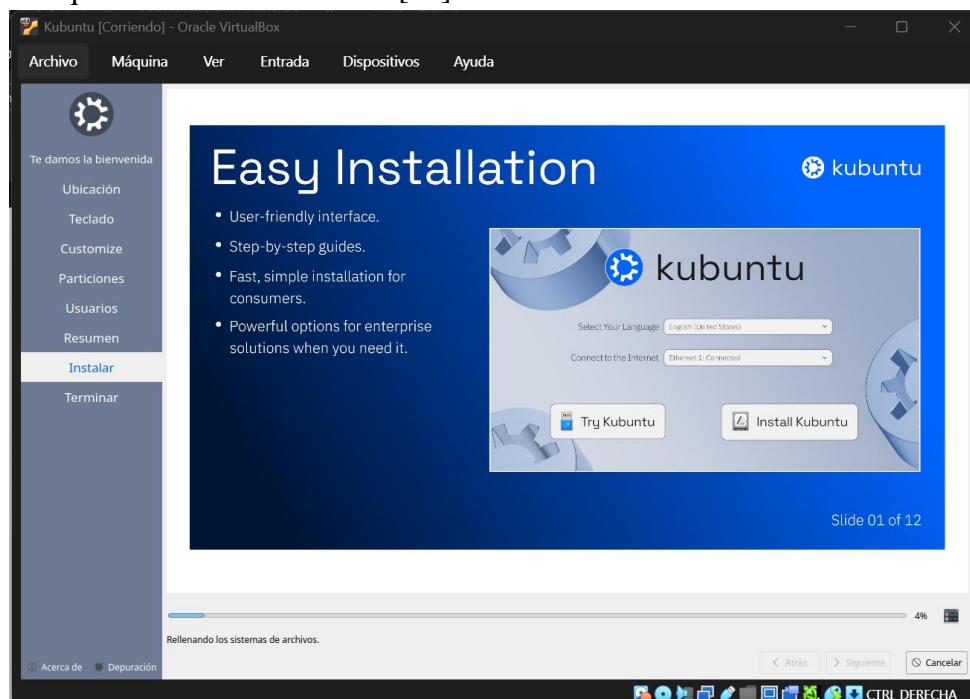


Figura 25: Progreso de instalación.

Una vez finalizada la instalación damos clic en Hecho [26].

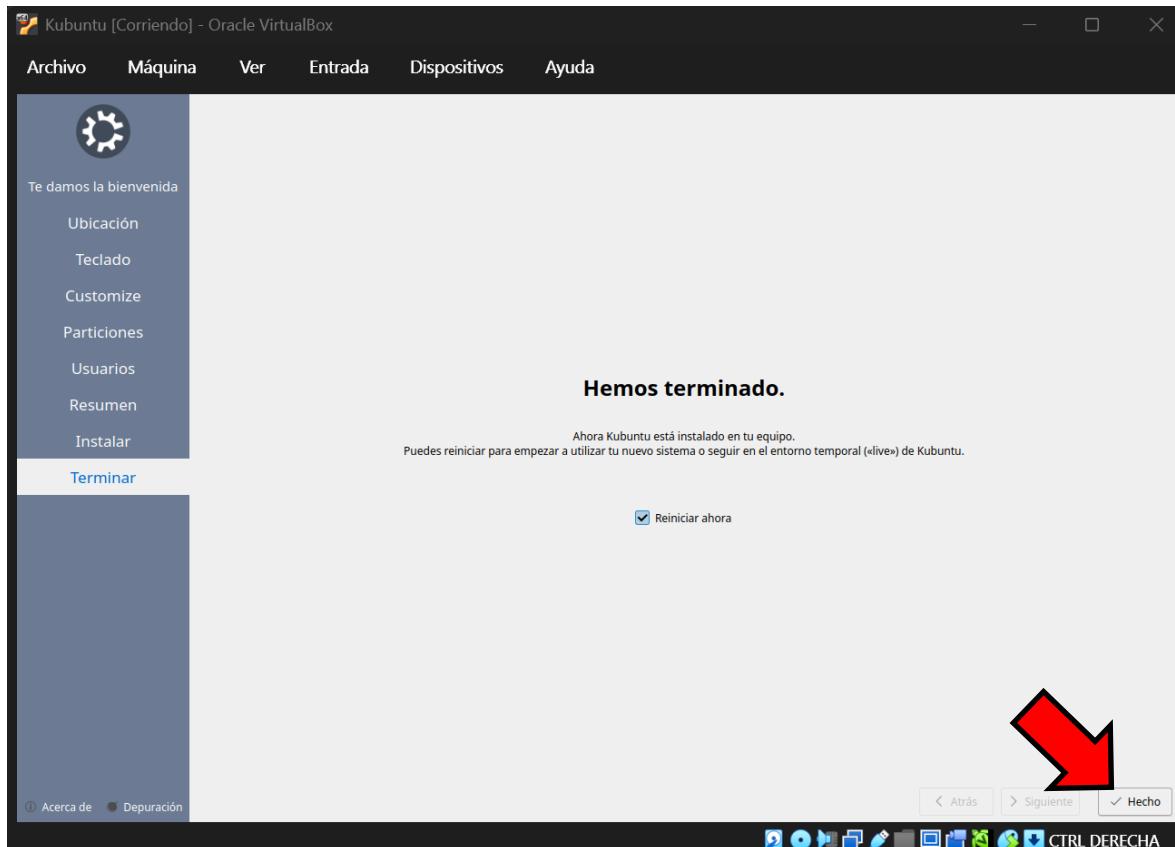


Figura 26: Instalación terminada.

Presionamos la tecla Enter y esperamos a que inicie Kubuntu[27]

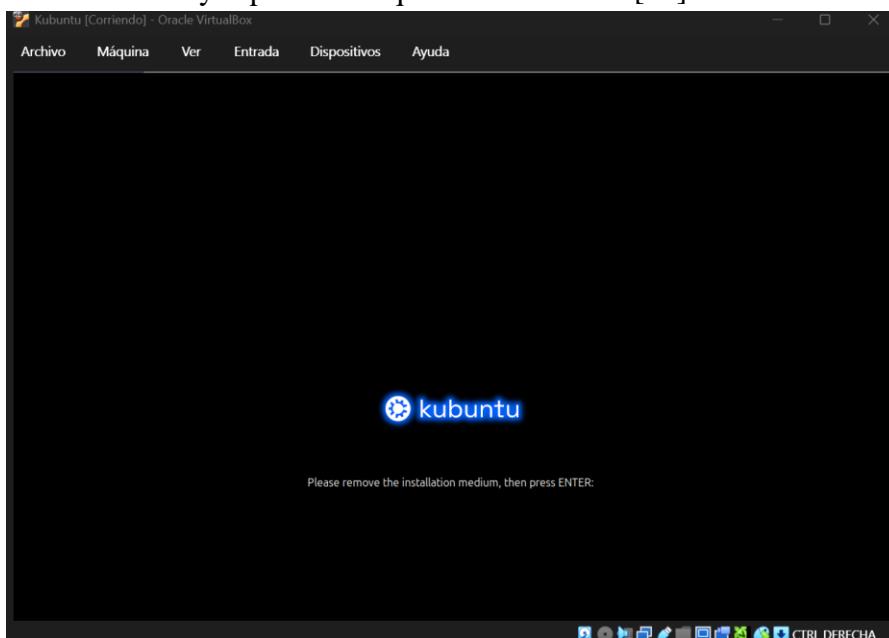


Figura 27: Primer inicio.

Aquí termina la instalación de la máquina virtual, procedamos con OpenStack [28]

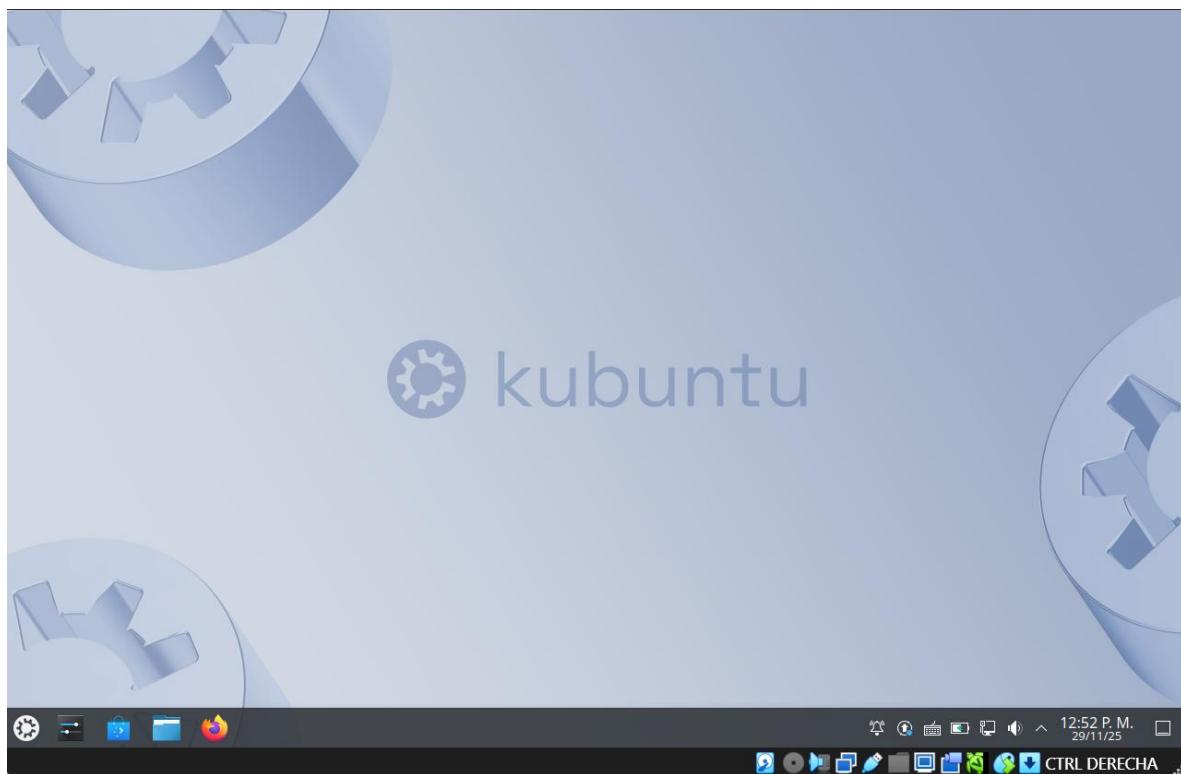


Figura 28: Maquina Funcional.

Capítulo 12. Instalación de OpenStack.

Abrimos la consola [29]

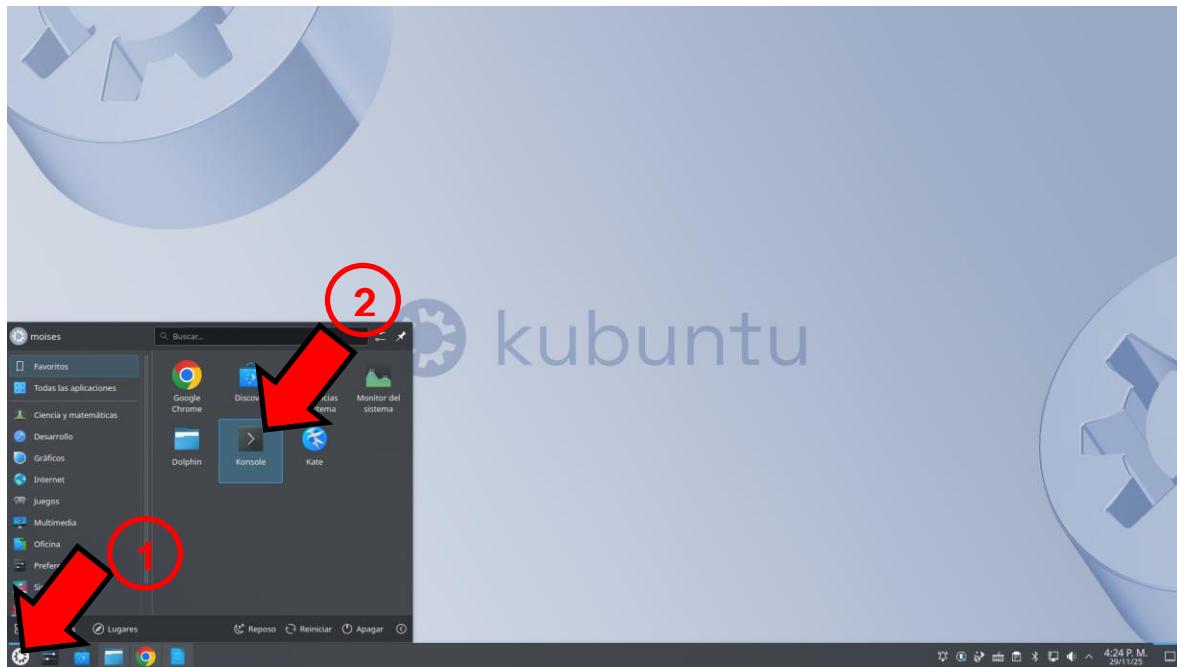
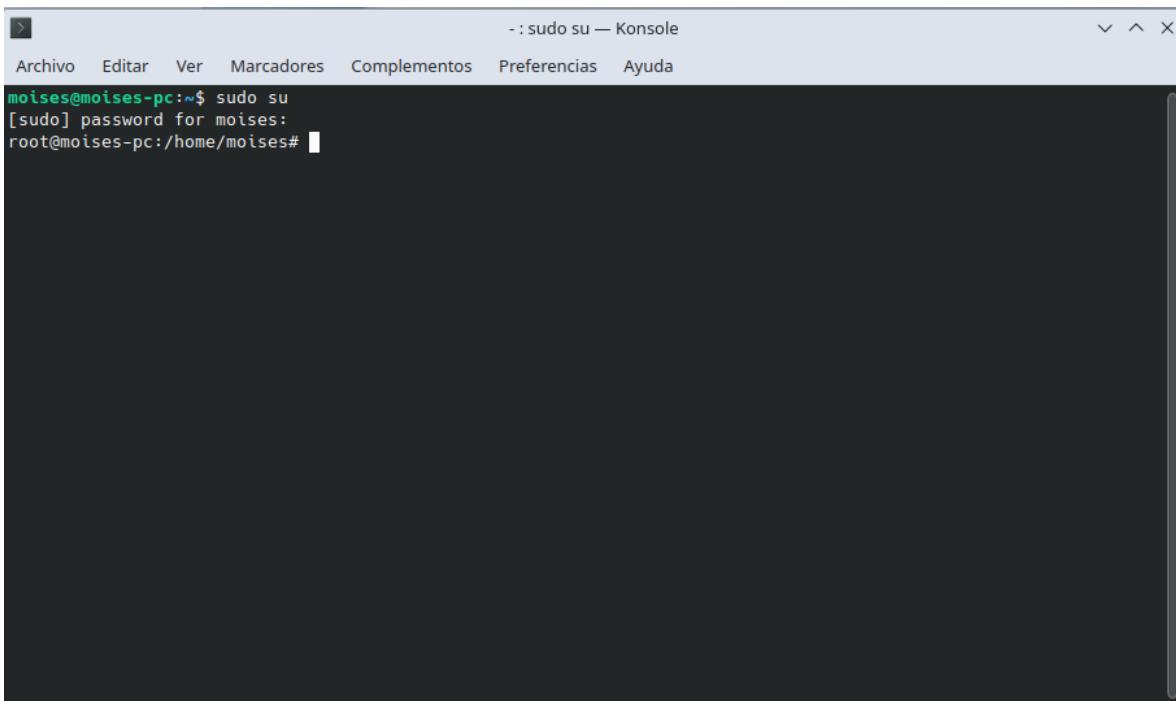


Figura 29: Abrir Consola.

Ejecutamos el comando `sudo su` y ponemos nuestra contraseña para entrar con privilegios de administrador y evitar que algunos comandos no corran por falta de permisos [30].



```
moises@moises-pc:~$ sudo su
[sudo] password for moises:
root@moises-pc:/home/moises#
```

Figura 30: Dar privilegios de administrador.

Realizamos una actualización general del sistema con el comando `apt update && apt upgrade` y presionamos enter [31].

```
- : sudo su — Konsole
Archivo Editar Ver Marcadores Complementos Preferencias Ayuda
moises@moises-pc:~$ sudo su
[sudo] password for moises:
root@moises-pc:/home/moises# apt update && apt upgrade
Obj:1 https://dl.google.com/linux/chrome/deb stable InRelease
Obj:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Obj:3 http://archive.ubuntu.com/ubuntu noble InRelease
Obj:4 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Obj:5 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Todos los paquetes están actualizados.
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Calculando la actualización... Hecho
Get more security updates through Ubuntu Pro with 'esm-apps' enabled:
 libvlc5 freerdp2-x11 libzvbi-common vlc-data libvlccore9 libjs-katex
 libwinpr2-2t64 libcurl4 libavdevice60 ffmpeg libpostproc57 libavcodec60
 libzvbi0t64 libavutil58 libswscale7 libfreerdp-client2-2t64 fonts-katex
 libswresample4 vlc-plugin-video-output 7zip libavformat60 libfreerdp2-2t64
 libvlc-bin vlc-plugin-base libavfilter9
Learn more about Ubuntu Pro at https://ubuntu.com/pro
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
root@moises-pc:/home/moises#
```

Figura 31: Actualización general.

Entramos a la página oficial de devstack y seguimos los pasos.

<https://docs.openstack.org/devstack/latest/>

En la consola ejecutamos `sudo useradd -s /bin/bash -d /opt/stack -m stack` [32]

```
root@moises-pc:/home/moises# sudo useradd -s /bin/bash -d /opt/stack -m stack
```

Figura 32: Creación de usuario stack.

Ejecutamos `sudo chmod +x /opt/stack` para darle permisos de ejecución al usuario [33]

```
root@moises-pc:/home/moises# sudo chmod +x /opt/stack
```

Figura 33: Permisos del usuario stack.

Le damos privilegios de administrador al usuario con el comando `echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack` [34]

```
root@moises-pc:/home/moises# echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
```

Figura 34: Dar privilegios al usuario stack.

Entramos al usuario con `sudo -u stack -i` [35]

```
root@moises-pc:/home/moises# sudo -u stack -i
stack@moises-pc:~$
```

Figura 35: Entrar al usuario stack.

Descargamos devstack usando el siguiente comando

`git clone https://opendev.org/openstack/devstack` [36]

```
stack@moises-pc:~$ git clone https://opendev.org/openstack/devstack
```

Figura 36: Descargar DevStack.

Una vez descargado con el comando `cd devstack` entramos a la carpeta descargada [37]

```
stack@moises-pc:~$ cd devstack
stack@moises-pc:~/devstack$
```

Figura 37: Ingreso a la carpeta devstack.

Con el comando `ip a s` puedes consultarnos nuestra ip y la anotamos [38]

```
stack@moises-pc:~/devstack$ ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp5s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 22:32:4d:06:35:d8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.100/24 brd 192.168.0.255 scope global dynamic noprefixroute enp5s0
        valid_lft 83461sec preferred_lft 83461sec
    inet6 fe80::2961:39a7:1faf:df8a/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether ea:90:f7:cc:19:fe brd ff:ff:ff:ff:ff:ff
4: br-ex: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether b7:43:86:a7:4b brd ff:ff:ff:ff:ff:ff
5: br-int: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 96:cb:07:d5:4f:c8 brd ff:ff:ff:ff:ff:ff
6: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:75:7f:3f brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
        valid_lft forever preferred_lft forever
```

Figura 38: Ver IP.

Con `sudo nano local.conf` creamos el archivo de configuración de openstack [39]

```
stack@moises-pc:~/devstack$ sudo nano local.conf
```

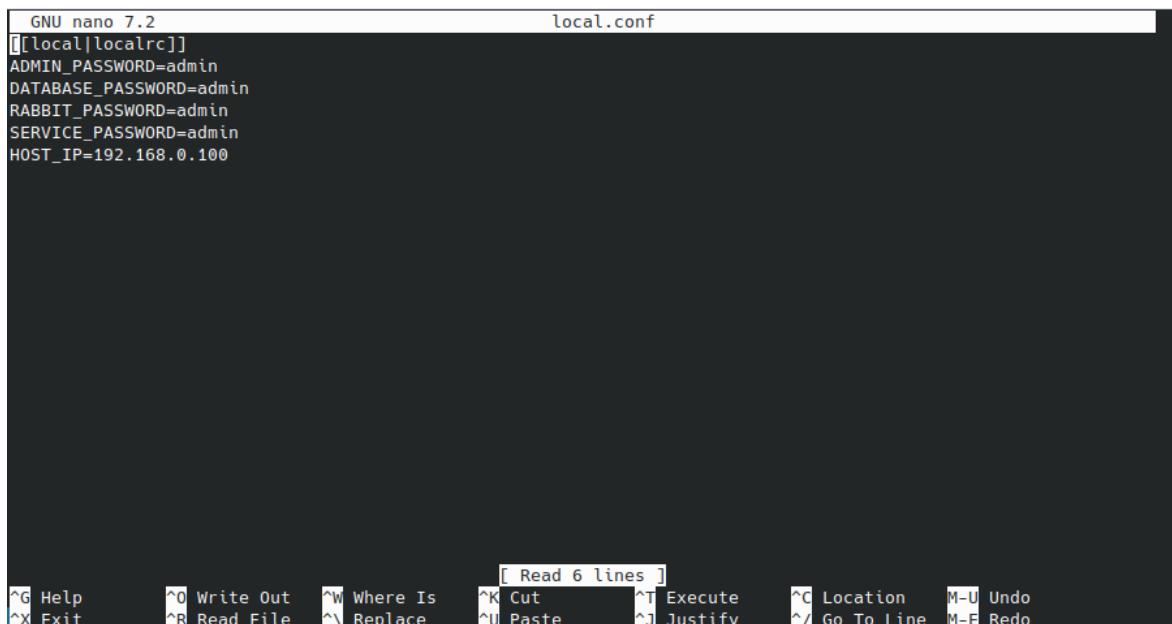
Figura 39: Crear archivo de configuración.

Se nos abrirá una ventana en la cual colocaremos el siguiente código

```
[[local|localrc]]  
ADMIN_PASSWORD=secret  
DATABASE_PASSWORD=$ADMIN_PASSWORD  
RABBIT_PASSWORD=$ADMIN_PASSWORD  
SERVICE_PASSWORD=$ADMIN_PASSWORD  
HOST_IP=
```

Todo lo que está en rojo usted debe borrarlo y colocar una contraseña, y en HOST_IP debe colocar la dirección ip que anotó, para guardar presione Ctrl+o, luego Enter para confirmar y por último Ctrl+x para salir.

Algo así debería quedar, esta es la configuración mínima requerida para comenzar a utilizar DevStack [40].



```
GNU nano 7.2                               local.conf  
[[local|localrc]]  
ADMIN_PASSWORD=admin  
DATABASE_PASSWORD=admin  
RABBIT_PASSWORD=admin  
SERVICE_PASSWORD=admin  
HOST_IP=192.168.0.100
```

Figura 40: Contenido del archivo de configuración.

Ejecute `./stack.sh` para iniciar la instalación, este proceso puede demorar dependiendo de las características de su pc [41].

```

Archivo Editar Ver Marcadores Complementos Preferencias Ayuda
+functions-common:apt_get_update:1180      time_start apt-get-update
+functions-common:time_start:2409      local name=apt-get-update
+functions-common:time_start:2410      local start_time=
+functions-common:time_start:2411      [[ -n '' ]]
+functions-common:time_start:2414      date +%%3N
+functions-common:time_start:2414      _TIME_START[$name]=`date +%%3N`
+functions-common:apt_get_update:1182      local 'proxies=http_proxy= https_proxy= no_proxy='
+functions-common:apt_get_update:1183      local 'update_cmd=sudo http_proxy= https_proxy= no_proxy= apt-get
update'
+functions-common:apt_get_update:1184      timeout 300 sh -c 'while ! sudo http_proxy= https_proxy= no_proxy=
apt-get update; do sleep 30; done'
Hit:1 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:4 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:5 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
+functions-common:apt_get_update:1188      REPOS_UPDATED=True
+functions-common:apt_get_update:1190      time_stop apt-get-update
+functions-common:time_stop:2423      local name=
+functions-common:time_stop:2424      local end_time
+functions-common:time_stop:2425      local elapsed_time
+functions-common:time_stop:2426      local total
+functions-common:time_stop:2427      local start_time
+functions-common:time_stop:2429      name=apt-get-update
+functions-common:time_stop:2430      start_time=`date +%%3N`
+functions-common:time_stop:2432      [[ -z $start_time ]]

```

Figura 41: Inicialización de OpenStack.

Una vez terminada la ejecución nos dejará las credenciales de acceso, ingresaremos la ip en el navegador y hacemos login [42].

	cinder	DELETE	1
nova_cell1 INSERT 5			
nova_cell1 UPDATE 22			
+-----+-----+-----+			

```

This is your host IP address: 192.168.0.100
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.0.100/dashboard
Keystone is serving at http://192.168.0.100/identity/
The default users are: admin and demo
The password: admin

Services are running under systemd unit files.
For more information see:
https://docs.openstack.org/devstack/latest/systemd.html

DevStack Version: 2026.1
Change: 4220539d470dddbdf2f75b75263a2fd4cfb0964f Merge "Re-add the ironic job in gate" 2025-11-26 20:40:34 +00
00
OS Version: Ubuntu 24.04 noble

2025-11-29 22:23:27.847 | stack.sh completed in 499 seconds.

```

Figura 42: OpenStack iniciado.

UNIVERSIDAD DE CÓRDOBA



Nos logueamos poniendo el usuario y la contraseña que asignamos, presionamos Sign in[43]

← → ⌂ △ No seguro 192.168.0.100/dashboard/auth/login/?next=/dashboard/

◎ ◻ ☆ ⓘ ⋮



openstack

Log in

User Name

Password

Figura 43: Login.

Capítulo 13. Configuración de OpenStack para la implementación de un servidor web.

Luego de haber entrado al dashboard en la consola corremos el siguiente comando `wget https://cloud-images.ubuntu.com/xenial/current/xenial-server-cloudimg-amd64-disk1.img` con este descargaremos la imagen que usaremos en la instancia para correr la pagina web [44].

```
stack@moises-pc:~/devstack$ wget https://cloud-images.ubuntu.com/xenial/current/xenial-server-cloudimg-amd64-disk1.img
```

Figura 44: Selección de idioma.

En la esquina superior izquierda cambiamos al proyecto demo [45]

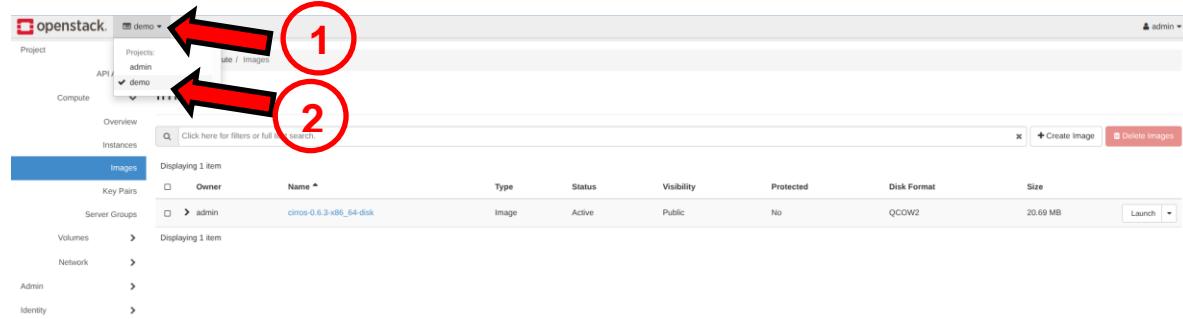


Figura 45: Selección de proyecto demo.

Nos vamos a admin, y damos click en OpenStack RC File, se nos descargará un archivo el cual debemos abrir y copiar todo su contenido [46].

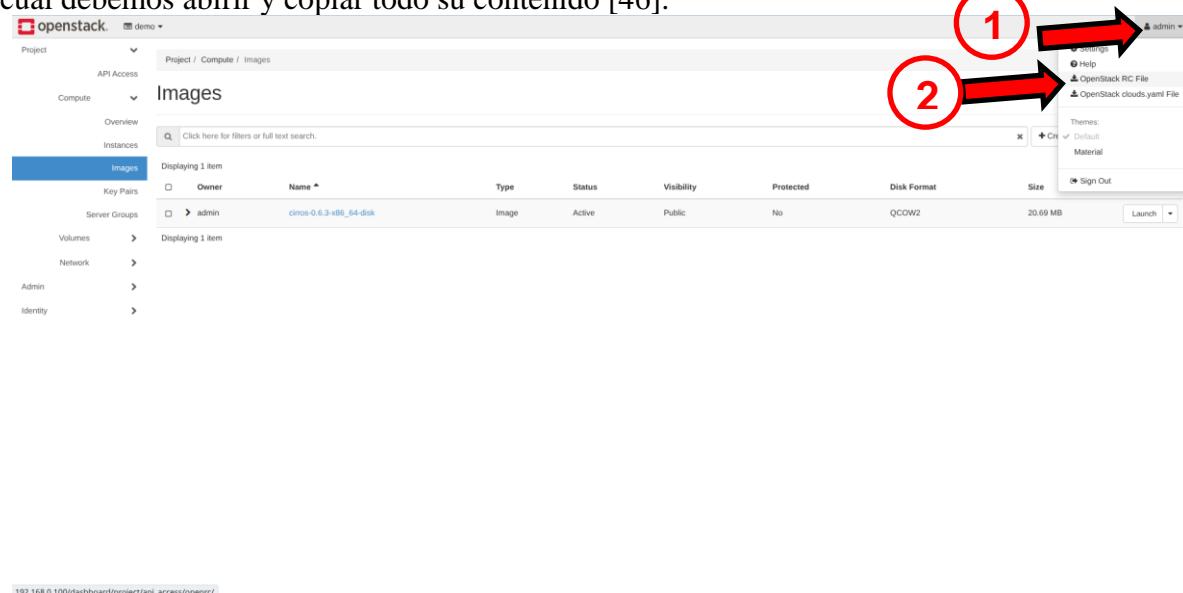
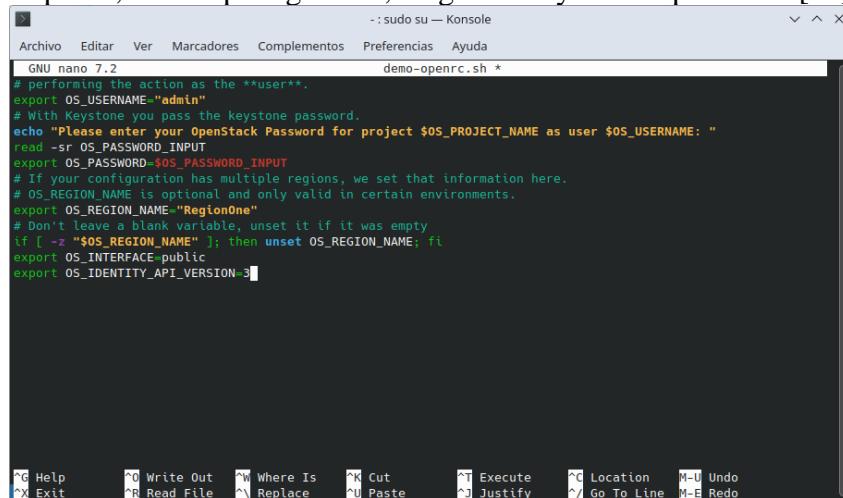


Figura 46: Descarga de demo-openrc.sh

En la consola corremos el comando `sudo nano demo-openrc.sh` y pegamos todo el contenido que habíamos copiado, Ctrl+o para guardar, luego Enter y Ctrl+x para salir [47].



```

GNU nano 7.2                               - : sudo su — Konsole
# performing the action as the **user**.
export OS_USERNAME="admin"
# With Keystone you pass the keystone password.
echo "Please enter your OpenStack Password for project $OS_PROJECT_NAME as user $OS_USERNAME: "
read -sr OS_PASSWORD_INPUT
# If your configuration has multiple regions, we set that information here.
# OS_REGION_NAME is optional and only valid in certain environments.
export OS_REGION_NAME="RegionOne"
# Don't leave a blank variable, unset it if it was empty
if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi
export OS_INTERFACE=public
export OS_IDENTITY_API_VERSION=3

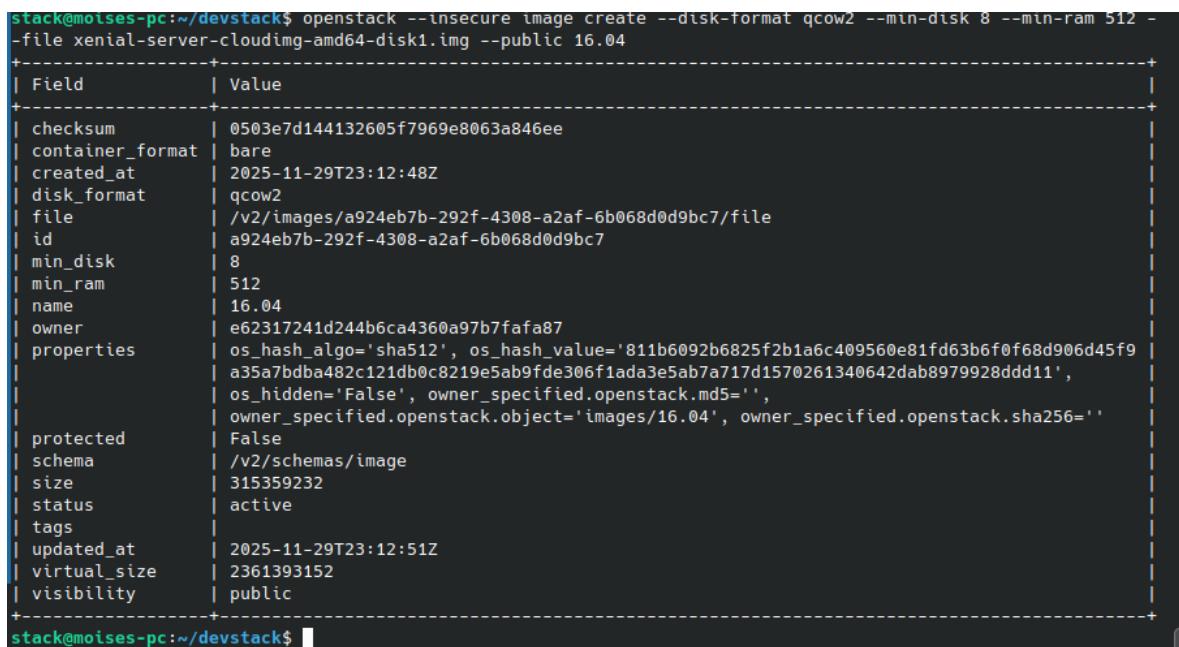
```

Figura 47: Crear y guardar demo-openrc.sh

Corremos el comando `./demo-openrc.sh` y escribimos la contraseña con la que nos logueamos

Ahora en la consola corremos el siguiente comando `openstack --insecure image create --disk-format qcow2 --min-disk 8 --min-ram 512 --file xenial-server-cloudimg-amd64-disk1.img --public 16.04` [48]

Esto que hicimos nos permite por medio de la consola crear la imagen que luego vamos a utilizar en la instancia.



```

stack@moises-pc:~/devstack$ openstack --insecure image create --disk-format qcow2 --min-disk 8 --min-ram 512 --file xenial-server-cloudimg-amd64-disk1.img --public 16.04
+-----+
| Field      | Value
+-----+
| checksum   | 0503e7d144132605f7969e8063a846ee
| container_format | bare
| created_at | 2025-11-29T23:12:48Z
| disk_format | qcow2
| file       | /v2/images/a924eb7b-292f-4308-a2af-6b068d0d9bc7/file
| id         | a924eb7b-292f-4308-a2af-6b068d0d9bc7
| min_disk   | 8
| min_ram   | 512
| name       | 16.04
| owner      | e62317241d244b6ca4360a97b7fafa87
| properties | os_hash_algo='sha512', os_hash_value='811b6092b6825f2b1a6c409560e81fd63b6f0f68d906d45f9
|           | a35a7dbba482c121db0c8219e5ab9fde306f1ada3e5ab7a717d1570261340642dab8979928ddd11',
|           | os_hidden='False', owner_specified.openstack.md5='',
|           | owner_specified.openstack.object='images/16.04', owner_specified.openstack.sha256=''
| protected  | False
| schema     | /v2/schemas/image
| size       | 315359232
| status     | active
| tags       |
| updated_at | 2025-11-29T23:12:51Z
| virtual_size | 2361393152
| visibility | public
+-----+
stack@moises-pc:~/devstack$ 

```

Figura 48: Crear imagen por consola.

Ahora vamos project, compute, images y verificamos que se haya creado la imagen [49]

Name	Type	Status	Visibility	Protected	Disk Format	Size
centos-6.5-x86_64-disk	Image	Active	Public	No	QCOW2	300.75 MB
cirros-0.6.3-x86_64-disk	Image	Active	Public	No	QCOW2	20.69 MB

Figura 49: Imagen creada.

Damos click en instances y luego en launch instance para crear una instancia [50].

Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
No items to display.									

Figura 50: Crear instancia.

Ponemos el nombre de nuestra instancia [51]

Launch Instance

Details

Source *	Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.
Flavor *	demo
Networks *	myWebsite
Network Ports	
Security Groups	
Key Pair	
Configuration	nova
Server Groups	
Scheduler Hints	
Metadata	

Total Instances (10 Max)
10%
0 Current Usage
1 Added
9 Remaining

< Back | Next > | Launch Instance

Figura 51: Asignar nombre a instancia.

En source, create new volume lo ponemos en no y seleccionamos la imagen que creamos dando clic en la flecha que apunta hacia arriba [52].

Launch Instance

Source * **1**

Select Boot Source: Image **2**

Create New Volume: Yes **3**

Name	Updated	Size	Format	Visibility
16.04	11/29/25 11:12 PM	300.75 MB	QCOW2	Public
cirros-0.6.3-x86_64-disk	11/29/25 11:06 PM	20.69 MB	QCOW2	Public

< Back | Next > | Launch Instance

Figura 52: Selección de imagen para la instancia.

Vamos a flavor y seleccionamos ds1G [53]

Launch Instance

Flavors manage the sizing for the compute, memory and storage capacity of the instance.

Allocated

Displaying 0 items

Name	vCPUs	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
Select a flavor from the available flavors below.						

Networks *

Displaying 0 items

Security Groups

Available 12

Select one

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Name	vCPUs	RAM	Total Disk	Root Disk	Ephemeral Disk	Public	
m1.nano	1	▲ 192 MB	1 GB	▲ 1 GB	0 GB	Yes	▲
m1.micro	1	▲ 256 MB	1 GB	▲ 1 GB	0 GB	Yes	▲
cirros256	1	▲ 256 MB	1 GB	▲ 1 GB	0 GB	Yes	▲
m1.tiny	1	512 MB	1 GB	▲ 1 GB	0 GB	Yes	▲
ds512M	1	512 MB	5 GB	▲ 5 GB	0 GB	Yes	▲
ds1G	1	1 GB	10 GB	10 GB	0 GB	Yes	▲
m1.small	1	2 GB	20 GB	20 GB	0 GB	Yes	▲

Figura 53: Asignar recursos a la instancia.

En network seleccionamos private [54]

Launch Instance

Details

Source

Flavor

Networks *

Networks provide the communication channels for instances in the cloud. You can select ports instead of networks or a mix of both.

Allocated

Displaying 0 items

Network	Subnets Associated	Shared	Admin State	Status
Select one or more networks from the available networks below.				

Network Ports

Displaying 0 items

Security Groups

Available 2

Select one or more

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Network	Subnets Associated	Shared	Admin State	Status	
shared	shared-subnet	No	Up	Active	▲
private	ipv6-private-subnet private-subnet	No	Up	Active	▲

Displaying 2 items

< Back **Next >** **Launch Instance**

Figura 54: Selección de red.

En Key Pair damos clic en Create Key Pair [55]

Launch Instance

The screenshot shows the 'Create Key Pair' step in the AWS Lambda console. On the left, a sidebar lists various resources: Details, Source, Flavor, Networks, Network Ports, Security Groups, Key Pair (highlighted with a red circle), Configuration, Server Groups, Scheduler Hints, and Metadata. The main area is titled 'Allocated' and shows a table for selected key pairs. Below it, the 'Available' section is shown with a table header 'Name', 'Type', and 'Fingerprint'. A search bar and a 'Select one' button are also present. At the bottom, there are 'Cancel', 'Back', 'Next', and 'Launch Instance' buttons.

Figura 55: Creación de llave.

Ponemos el nombre, en tipo seleccionamos SSH Key y damos click en Create Keypair [56]

Create Key Pair

Key Pairs are how you login to your instance after it is launched. Choose a key pair name you will recognize. Names may only include alphanumeric characters, spaces, or dashes.

Key Pair Name *

ssh-key

Key Type *

SSH Key

3

Create Keypair

Copy Private Key to Clipboard

Done

Figura 56: Configurar llave.

Se nos desplegará un texto, lo copiamos completo dando clic en el botón Copy Private Key to Clipboard [57]

Create Key Pair

Key Pairs are how you login to your instance after it is launched. Choose a key pair name you will recognize. Names may only include alphanumeric characters, spaces, or dashes.

Key Pair Name *

ssh-key

Key Type *

SSH Key

Private Key

```
----BEGIN RSA PRIVATE KEY----
MIIEogIBAAKCAQEAzkm/aXNCVPB9nwB74qVGTVoU2BLciuwS68Qbt8s95Jicmte
l/zLGa/6OcCEwvfy1yU03el9bOG8my+lWPe0T/TK2Hte3P6Cshl+i7enfj4H1em
/s/D38Khz0Fzp6SYjY57Gm1yTQnrR18gdsAip3WcGPDoyT8z1WzZdwFb5kuvOeL
7CXrTGOBCmRZ22AJU2cymPLZhSTDY413tCbUWeYAD7l0K3yDMz+cGSm5xT2Ni1N
NsbEU29RzVAU1So9V+XhtLj64wmij8dV9G4HUtIYIVcaKzrml5b5D+MzHF7C5TV+
XLijN9YamD8CnHsR06tDd0TZM8iKSnqcL9F8HQIDAQABaoIBAAMse90K38x07VbN
MXpttz1cHoapxH9NDLkdSDNsW1Sze7Am1p2f1b/fZQneXCxjg/RMyD04aeUNjdXM
tmG114eAY88XAoATmJTx4B+pteLYIXjZ8EebCng2liaJ6y146p/CwUH2zPtCW8wN
OoKZEe9hb7Uyf2IZBsMRmYarLdY5jdS+olaq74N+NPdEnE/3L36iinbbetbpaDhE
qTkblrPTNJ3mzIN0RZ7N6Nf7lVrqQGSvACHpdnbjDXcKSm47i07N/Bv+ekfZAZ
hFkWg1o/4xnLecBygKZyZq2XqFcR7b5XPtSV5qRA+s8zhzV+Lv2khyQccKxrXdbJ
5OKdvkECgYEAIzDevLSIQb7snflfcfru1tvIAEfUHVOS2YA/fhIHKDvQoL
eidyh8nPvIuol1xhpFW6E158x+gkNxZGw4bcX/5+/25IB/F/XzJgJkDfG3dp9z7
Re4SSBRcf9iK0alt+WLLRF//Hx8P3MVQto61J4lcey178ReK+VYrxtcCgYEAOYMs
```



Create Keypair

Copy Private Key to Clipboard

Done

Figura 57: Copiar contenido de llave.

En la consola escribimos `nano ssh-key.pem` y pegamos lo copiado Ctrl+o para guardar, presionamos Enter y luego Ctrl+x [58]

```
GNU nano 7.2                                         ssh-key.pem *
Re44SBRCf9IK0alt+WLLRF//Hx8P3MVQto61J4lcey178ReK+VYrxtcCgYEAOYMs
C5CVrk5jeMnxZvXGp6WNDjChqjz/7B3vvfYo4F1Y1etGR/qYBktiuJZrKsaN3g
9o3Bil8CkF4IiQcoZ3hbToJNTcfAgrTvgkvlsk8/rTYCbc5Q03SoLru2cjWyIe3
++BYIimWNFD4scwKvdppqvEZLIHkW0Ypecvl82isCgYBK/VTERHuRjwwhbySbb4Ph
Has6NEbDEIGRvCTB+fgdg6CZgVQoEAxJPfnxvDzEij0e8F/PsqNIDk@q9K5n6K
EXQQGe0cEqaza4/TiN3VpaZnwTV/y7xgQKFVaj9TUZcfwv8FGood/TFKNPq8bTD
10YB8FTVPf3zAvG0nxZ+GQKBgFh7i3h+4ZgvTI9N5gmMig9/sJgd8CTuR0yqjPo7
uQ41aPJE0EBjSTBuyFnXTIQN5w6SK4YBCsKIS0TaYtlHzaacVoZnl465bKf6WF
qwOFjcsZpuzAxBxV1GSpTgHgjYU3PE6V2uLiQvSc72U0xmT6wwEtuTV/vsT8vuf
vkjtAoGAIzlpI0zokzDyw5Gg9D6rdR4N2TrjIKIMITYVFYyht82IT3yXTzhwYiP
xWJ5hNHT8UzWrnB5sSx95q0a+y0KqaA0017ChTtWwPuVrURinWfeJ7WhTJ7Qqcr6
eY8Zft857FaClwbWg9DQ/o+OWI8SNniU27Q7JVQgNzGTk5VIQs=
-----END RSA PRIVATE KEY-----
```

Figura 58: Guardar llave.

Ponemos el comando `chmod 400 ssh-key.pem` para darle permisos de lectura [59]

```
stack@moises-pc:~/devstack$ chmod 400 ssh-key.pem
stack@moises-pc:~/devstack$
```

Figura 59: Dar permisos al archivo de la llave.

Volvemos al navegador y damos click en Done [60]

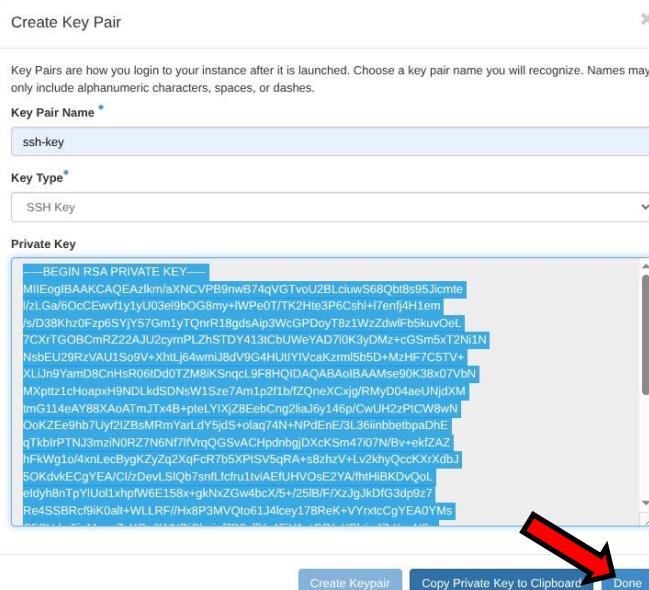


Figura 60: Confirmar creación de llave.

Luego vamos a configuration y pegamos el script que crea la página web con su respectivo contenido, por último, damos click en launch instance [61]

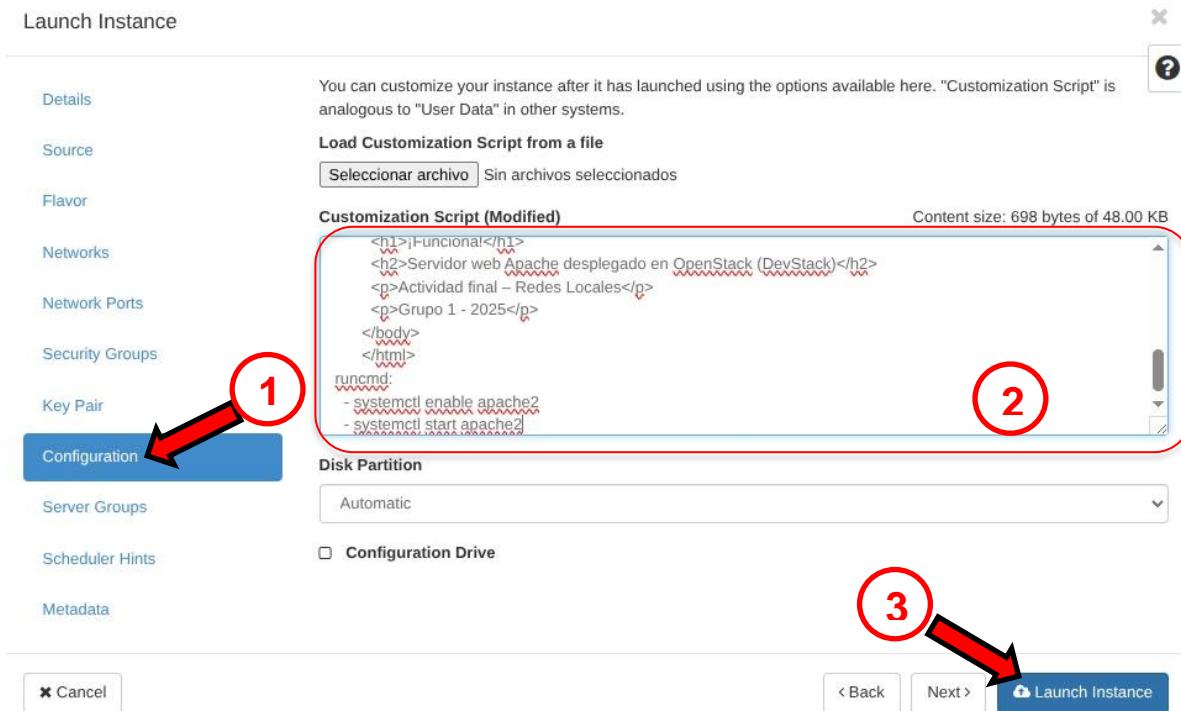


Figura 61: Script de la página.

En el apartado Actions de la instancia desplegamos el menú y seleccionamos la opción Associate Floating IP [62]

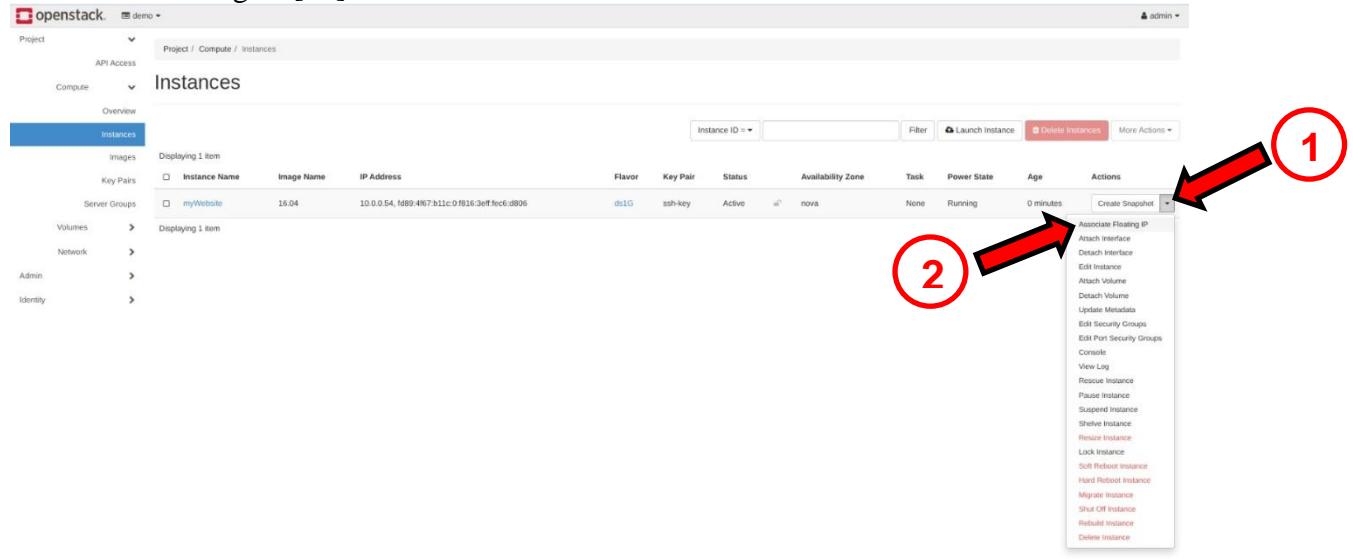


Figura 62: Asociar IP flotante a la instancia.

Damos click en el signo mas (+) [63]

Manage Floating IP Associations

IP Address *

No floating IP addresses allocated +

Select the IP address you wish to associate with the selected instance or port.

Port to be associated *

myWebsite: 10.0.0.54

Cancel Associate

Figura 63: Nueva IP flotante.

Damos click en Allocate IP [64]

Allocate Floating IP

Pool *

public

Floating IP Address (optional) ⓘ

|

Description

DNS Domain

DNS Name

Description:

Allocate a floating IP from a given floating IP pool.

Project Quotas

Floating IP 0 of 50 Used

Cancel Allocate IP

Figura 64: Crear IP flotante.

Clic en Associate [65]

Manage Floating IP Associations

IP Address *

Port to be associated *

Select the IP address you wish to associate with the selected instance or port.

Cancel Associate

Figura 65: Confirmar asociación.

Ahora vamos a Project, Network, Security Groups y damos clic en Manage Rules [66]

Project > Compute > Security Groups

Network > Security Groups

Security Groups

Manage Rules

Figura 66: Configurar reglas del grupo de seguridad.

Damos clic en Add Rule [67]

Manage Security Group Rules: default (3b657b14-4fc0-409d-a8f9-708846b433d8)

+ Add Rule Delete Rules

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
Egress	IPv4	Any	Any	0.0.0.0/0	-	-	Delete Rule
Egress	IPv6	Any	Any	::/0	-	-	Delete Rule
Ingress	IPv4	Any	Any	-	default	-	Delete Rule
Ingress	IPv6	Any	Any	-	default	-	Delete Rule

Figura 67: Agregar reglas.

En Rule desplegamos y seleccionamos All ICMP, y damos click en add [68]

Add Rule

Rule *

Description ?

Direction

Remote * ?

CIDR * ?

Description:

Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.

Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Cancel **Add**

Figura 68: Añadir regla ICMP.

Luego hacemos lo mismo, Add Rule, en Rule desplegamos y seleccionamos SSH, y damos click en add [69]

Add Rule

Rule *

Description ?

Remote * ?

CIDR * ?

Description:

Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.

Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Cancel **Add**

Figura 69: Añadir regla SSH.

Luego hacemos lo mismo, Add Rule, en Rule desplegamos y seleccionamos HTTP, y damos click en add [70]

Add Rule

Rule *

HTTP

Description ?

Remote * ?

CIDR

CIDR* ?

0.0.0.0/0

Description:

Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.

Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Cancel
Add

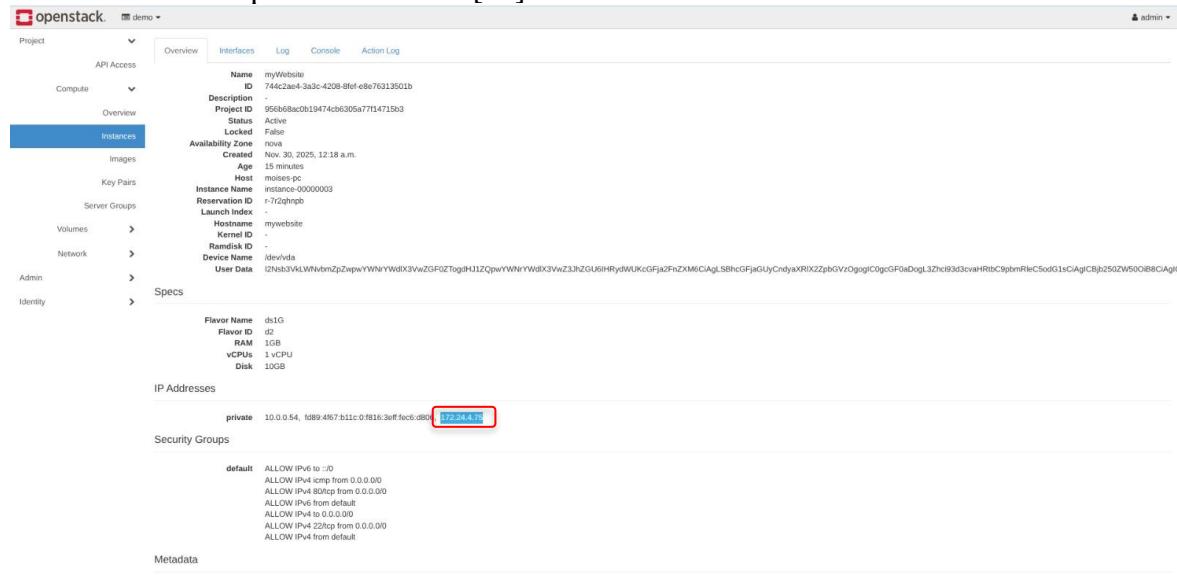
Figura 70: Añadir regla HTTP.

Nos Vamos a instances y damos click en el nombre de la instancia [71]

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
<input type="checkbox"/> myWebsite	16.04	10.0.0.54, 172.24.4.75, f689:4f67:b11c:0f816:3eff:fc6:d806	ds1G	ssh-key	Active	nova	None	Running	14 minutes	<button style="border: none; background-color: #0070C0; color: white; border-radius: 5px; padding: 2px 10px; font-size: small;">Create Snapshot</button>

Figura 71: Entrar a la instancia.

En IP addresses copiamos la última [72]



The screenshot shows the OpenStack dashboard with the 'myWebsite' instance selected. The 'IP Addresses' section displays the private IP address 10.0.0.54, followed by the public IP address 172.24.4.175, which is highlighted with a red box.

Figura 72: Copiar IP de la instancia.

Nos vamos a la consola y le hacemos ping con [ping 172.24.4.175](#) para verificar que tenemos comunicación con la instancia [73]

```
stack@moises-pc:~/devstack$ ping 172.24.4.75
PING 172.24.4.75 (172.24.4.75) 56(84) bytes of data.
64 bytes from 172.24.4.75: icmp_seq=1 ttl=63 time=6.33 ms
64 bytes from 172.24.4.75: icmp_seq=2 ttl=63 time=1.53 ms
64 bytes from 172.24.4.75: icmp_seq=3 ttl=63 time=0.311 ms
64 bytes from 172.24.4.75: icmp_seq=4 ttl=63 time=0.256 ms
^C
--- 172.24.4.75 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3035ms
rtt min/avg/max/mdev = 0.256/2.104/6.325/2.489 ms
stack@moises-pc:~/devstack$
```

Figura 73: Ping a la instancia.

Ahora en la consola ejecutamos `ssh -i ssh-key.pem ubuntu@172.24.4.175` damos enter y escribimos yes [74]

```
stack@moises-pc:~/devstack$ ssh -i ssh-key.pem ubuntu@172.24.4.75
The authenticity of host '172.24.4.75 (172.24.4.75)' can't be established.
ED25519 key fingerprint is SHA256:vzz1oafqNWy/a7CdJY5dats0Ii423cNXR3C7eFdK8M.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.24.4.75' (ED25519) to the list of known hosts.
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.4.0-210-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

UA Infra: Extended Security Maintenance (ESM) is not enabled.

0 updates can be applied immediately.

45 additional security updates can be applied with UA Infra: ESM
Learn more about enabling UA Infra: ESM service for Ubuntu 16.04 at
https://ubuntu.com/16-04

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@mywebsite:~$
```

Figura 74: Creación de usuario Ubuntu dentro de la máquina que está corriendo dentro de la instancia.

Agregamos el nombre del host (mywebsite) al archivo host del usuario en el que estamos con el comando `sudo nano /etc/hosts` para que pueda tener acceso a internet, debería quedar así [] y presionamos Ctrl+o, Enter y Ctrl+x [75]

```
- : sudo -u — Konsole
Archivo Editar Ver Marcadores Complementos Preferencias Ayuda
GNU nano 2.5.3 File: /etc/hosts
127.0.0.1 localhost
127.0.1.1 mywebsite
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Figura 75: Agregar host al archivo de hosts de la maquina.

Reiniciamos el servicio con `sudo hostnamectl set-hostname mywebsite` [76]

```
ubuntu@mywebsite:~$ sudo hostnamectl set-hostname mywebsite
ubuntu@mywebsite:~$
```

Figura 76: Reiniciar servicio de hosts.

Ahora en la consola editamos el archivo dns para poder acceder a internet con `sudo nano /etc/resolv.conf` el archivo debe quedar de la siguiente manera [77] y presionamos Ctrl+o, Enter y Ctrl+x

```
-:sudo -u — Konsole
Archivo Editar Ver Marcadores Complementos Preferencias Ayuda
GNU nano 2.5.3 File: /etc/resolv.conf Modified
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
# DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 8.8.8.8
nameserver 1.1.1.1

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos ^Y Prev Page
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line ^V Next Page
```

Figura 77: Agregar las DNS.

Instalamos apache con `sudo apt install apache2 -y` [78]

```
ubuntu@mywebsite:~$ sudo apt install apache2 -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
  liblua5.1-0 ssl-cert
Suggested packages:
  www-browser apache2-doc apache2-suexec-pristine | apache2-suexec-custom openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap liblua5.1-0 ssl-cert
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 1559 kB of archives.
After this operation, 6448 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu xenial/main amd64 libapr1 amd64 1.5.2-3 [86.0 kB]
Get:2 http://archive.ubuntu.com/ubuntu xenial/main amd64 libaprutil1 amd64 1.5.4-1build1 [77.1 kB]
Get:3 http://archive.ubuntu.com/ubuntu xenial/main amd64 libaprutil1-dbd-sqlite3 amd64 1.5.4-1build1 [10.6 kB]
Get:4 http://archive.ubuntu.com/ubuntu xenial/main amd64 libaprutil1-ldap amd64 1.5.4-1build1 [8720 B]
Get:5 http://archive.ubuntu.com/ubuntu xenial/main amd64 liblua5.1-0 amd64 5.1.5-8ubuntu1 [102 kB]
Get:6 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 apache2-bin amd64 2.4.18-2ubuntu3.17 [927 kB]
Get:7 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 apache2-utils amd64 2.4.18-2ubuntu3.17 [81.9 kB]
Get:8 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 apache2-data all 2.4.18-2ubuntu3.17 [162 kB]
Get:9 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 apache2 amd64 2.4.18-2ubuntu3.17 [86.8 kB]
Get:10 http://archive.ubuntu.com/ubuntu xenial/main amd64 ssl-cert all 1.0.37 [16.9 kB]
Fetched 1559 kB in 1s (895 kB/s)
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
  LANGUAGE = (unset),
  LC_ALL = (unset),
  LC_TIME = "es_CO.UTF-8",
  LC_MONETARY = "es_CO.UTF-8",
  LC_ADDRESS = "es_CO.UTF-8",
  LC_TELEPHONE = "es_CO.UTF-8",
```

Figura 78: Instalar apache2.

Por último, ponemos la ip de la instancia en el navegador, presionamos Enter y listo, tenemos nuestra página web desplegada dentro de OpenStack usando apache2 [79]

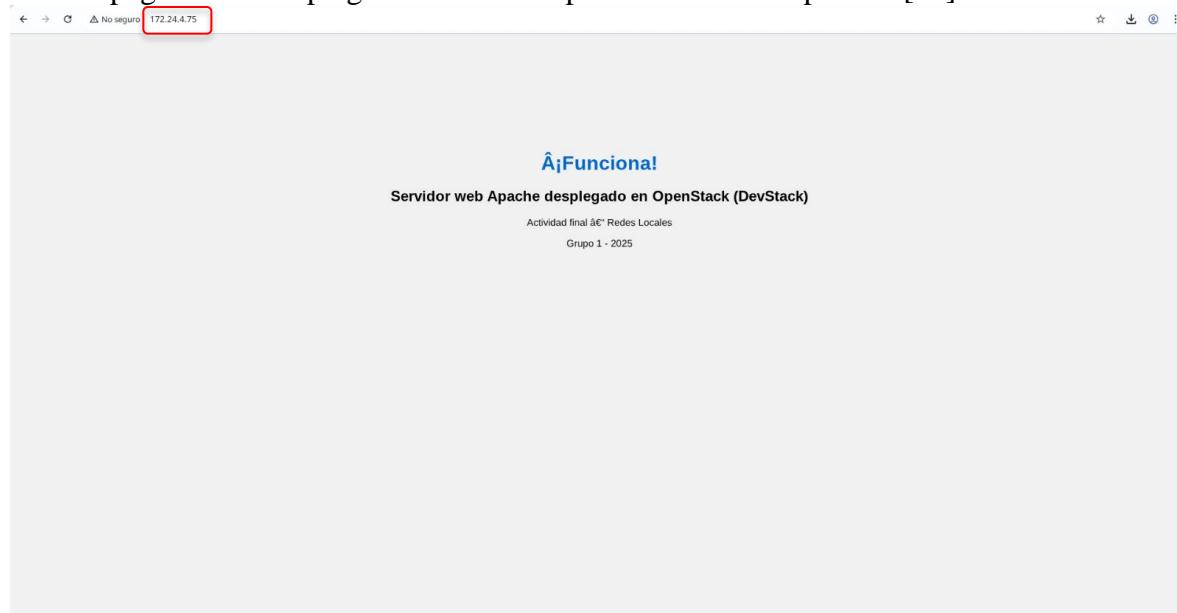


Figura 79: Pagina Web desplegada.