

Despliegue de un Datacenter con Openstack

Matías Capucho, Santiago Elizondo
Universidad de la República, Montevideo, Uruguay.
Email: matias.capucho@fing.edu.uy, selizondo@fing.edu.uy

4 de Agosto de 2019

Resumen

El presente documento estudia los aspectos fundamentales de la plataforma Openstack. A su vez, relata el proceso de diseño, instalación, y configuración básica de un Datacenter mediante la utilización de dicha plataforma. Se especifican los pasos necesarios a seguir y se detallan puntos sumamente relevantes a tener en cuenta.

Índice

1. Introducción	3
2. Marco teórico	4
2.1. Cloud computing	4
2.2. Virtualización	5
2.3. Contenerización	6
2.4. Datacenters	6
2.5. Redes	7
3. Openstack	9
3.1. Origen y definición	9
3.2. Módulos Core	9
3.2.1. Keystone	10
3.2.2. Nova	11
3.2.3. Neutron	13
3.2.4. Glance	15
3.2.5. Cinder	17
3.2.6. Swift	19
3.3. Tipos de nodos	20
3.4. Servicios de infraestructura	21
3.5. Métodos de instalación	22
3.5.1. Ansible	23
3.6. Arquitectura	23
3.6.1. Arquitectura de red	24
3.7. Configuración OSA	28
3.7.1. Convenciones	28
3.7.2. Inventario	29
3.7.3. openstack_user_config.yml	29
4. Instalación	30
4.1. Diseño de arquitectura	30
4.2. Ambiente de trabajo	32
4.2.1. Hardware utilizado	32
4.2.2. Conexión remota hacia el servidor renata	33
4.2.3. Virtualización con KVM	33
4.2.4. Especificaciones servidor renata	40
4.2.5. Acceso al exterior desde nodos	41
4.3. Preparación de nodos	42
4.4. Configuración	48
4.4.1. Configuración claves SSH	48
4.4.2. Archivos de configuración OSA	49

4.4.3. Generación de claves	52
4.4.4. Correcciones	52
4.5. Ejecución de playbooks	53
4.6. Verificación	54
5. Interacción	55
5.1. Configuraciones de administrador	56
5.2. Interacción de un usuario	61
5.3. Acceso a una instancia	69
5.3.1. Por SPICE	69
5.3.2. Por SSH	70
6. Inconvenientes	73
7. Trabajo a futuro	75
8. Conclusiones	76

1

Introducción

En la actualidad hay un fuerte potencial de crecimiento en los datacenters, principalmente dado por la demanda de sus servicios provenientes de nuevas tendencias como la digitalización de la información, el crecimiento del comercio electrónico y la adopción del cloud computing como una alternativa real. En función de esto resulta sumamente interesante investigar alternativas y procedimientos que deben ser llevados a cabo para desplegar y operar un datacenter.

Una tendencia que fue en aumento en los últimos años es la utilización de Openstack como plataforma de despliegue y gestión. Por lo tanto la finalidad de este informe será relatar una primera experiencia con dicha plataforma en una ambiente de prueba en el Instituto de Computación de la Facultad de Ingeniería.

2

Marco teórico

A continuación se presentan conceptos introductorios relevantes para comprender el estudio que se desarrollará más adelante en el informe.

2.1. Cloud computing

Cloud computing resulta ser un modelo que involucra tanto a los servicios computacionales provistos a los clientes a través de Internet, como a la implementación de hardware y software que logra proveerlos. Esta implementación se aloja en los denominados datacenters, que integran adecuadamente los diversos recursos tales como redes, servidores, almacenamiento y software necesarios para ofrecer los mencionados servicios en función de la demanda. Este modelo computacional, impulsado durante el siglo 21, ha evolucionado velozmente y ha migrado el mundo de TI de las antiguas PCs locales de usuarios y de los cuartos de servidores empresariales a la llamada “nube” alojada en lejanos datacenters. Según la definición brindada por The National Institute of Standards and Technology (NIST) [15], la computación en la nube debe cumplir con 5 características esenciales:

- **On-demand self-service:** los servicios alojados deben poder obtener capacidad de cómputo a demanda, consumiendo los recursos requeridos y sin que exista interacción humana con proveedores.
- **Broad network access:** los servicios deben encontrarse disponibles a través de la red y accesibles mediante mecanismos estándares.
- **Resource pooling:** los recursos computacionales se mantienen en grupos pudiendo ser asignados a múltiples consumidores en forma dinámica según la demanda necesaria.
- **Rapid elasticity:** los recursos deben poder escalar en forma sencilla e incluso automática en función de la demanda. De esta forma, los consumidores de servicios tendrán una visión de la nube como una fuente de recursos ilimitada.
- **Measured service:** los recursos consumidos deben ser monitorizados y medidos con un determinado nivel de abstracción en función del servicio provisto. Deben existir reportes de consumo que brinden absoluta transparencia tanto al proveedor como al consumidor.

Estas propiedades ambiciosas demuestran que brindar un servicio de cloud no es para nada trivial y requiere grandes conocimientos de TI a la hora de diseñar y poner en marcha un datacenter.

En cuanto a los modelos de cloud computing, existen tres clasificaciones que se distinguen según el tipo de servicio provisto [66]:

- **Software as a Service (SaaS):** refiere al servicio que provee aplicaciones a través de internet que se encuentran corriendo en la infraestructura de la nube (datacenter). Los consumidores del servicio no manipulan la infraestructura subyacente ni las configuraciones de aplicación.

- **Platform as a Service (Paas):** consiste en proveer recursos a nivel de plataforma, como por ejemplo soporte de sistemas operativos, que permiten al cliente desplegar sus propias aplicaciones. El consumidor no puede manipular la infraestructura pero sí es capaz de manipular lo que se despliega sobre ella.
- **Infrastructure as a Service (IaaS):** se asocia al suministro de recursos de infraestructura a demanda. Se suele proveer al cliente de capacidad de procesamiento, almacenamiento y conectividad de red. Típicamente se ofrecen máquinas virtuales con la posibilidad de que el cliente manipule los sistemas operativos y las aplicaciones. Nuevamente el consumidor no es capaz de influir en la infraestructura que implementa la nube.

2.2. Virtualización

Uno de los principales conceptos que se encuentra involucrado en la implementación de cloud computing es el de virtualización [61][65]. Esta tecnología permite simular diversos ambientes con recursos dedicados a partir de un solo sistema físico. Es posible crear aplicaciones, servidores, almacenamiento y redes, utilizando al máximo todos los recursos del sistema subyacente aumentando el rendimiento general. Los usuarios finales interactúan directamente con las virtualizaciones, llamadas máquinas virtuales. Una máquina virtual puede ser transferida de un sistema host a otro y funcionar de igual forma.

La implementación de esta tecnología se da mediante los denominados **hipervisores** [61][56], los cuales se encargan de conectar los recursos físicos de la máquina host con las máquinas virtuales. Estos trabajan sobre el sistema operativo o directamente en el hardware, creando una plataforma virtual, que divide los recursos físicos, sobre la cual se ejecutan las diferentes virtualizaciones. Por otro lado también son responsables de crear ambientes aislados que brinden seguridad entre las máquinas virtuales.

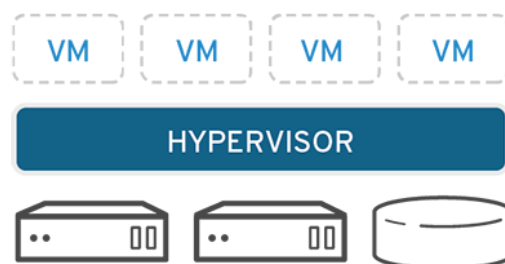


Figura 2.1: Hipervisores. Extraída de [61].

Los hipervisores se suelen clasificar en dos tipos:

- **Nativos o bare metal:** se trata de software que corre directamente sobre el hardware del sistema host, controlando el hardware y monitoreando el sistema operativo invitado. Algunos ejemplos son Oracle VM, Microsoft Hyper-V, VMware ESXi y Xen.
- **Alojados o hosted:** consisten en hipervisores que corren dentro de un sistema operativo tradicional. Se encuentran en una capa de aplicación por encima del sistema operativo del host pero por debajo del sistema operativo guest. Algunos ejemplos son Oracle VM VirtualBox, VMware Server y Workstation, Microsoft Virtual PC, KVM, QEMU y Parallels.

KVM [60], siglas de Kernel-based Virtual Machine, se trata de una tecnología de virtualización open source sobre Linux. Se encuentra formada por un módulo del kernel denominado `kvm.ko` y permite transformar un sistema operativo Linux en un hipervisor. Como módulo del kernel se encuentra incluido en Linux a partir de la versión 2.6.20. Como hipervisor se clasifica como de tipo alojado, o hosted, utilizando los componentes del sistema Linux para implementar cada máquina virtual como un proceso de Linux.

2.3. Contenerización

Un contenedor [10] es una unidad de software liviana diseñada para ejecutar una aplicación. Está formado únicamente por el código de la aplicación y las dependencias necesarias para que esta ejecute, creando un paquete portable e independiente. De esta forma múltiples contenedores pueden correr como procesos diferentes en una misma máquina host compartiendo el kernel del sistema operativo. Si bien tanto contenedores como máquinas virtuales tienen como fin aislar y compartir recursos de la máquina host, lo hacen en diferentes niveles. Los primeros virtualizan únicamente el sistema operativo guest, utilizando el kernel del host subyacente, mientras que las VMs virtualizan a nivel de hardware.

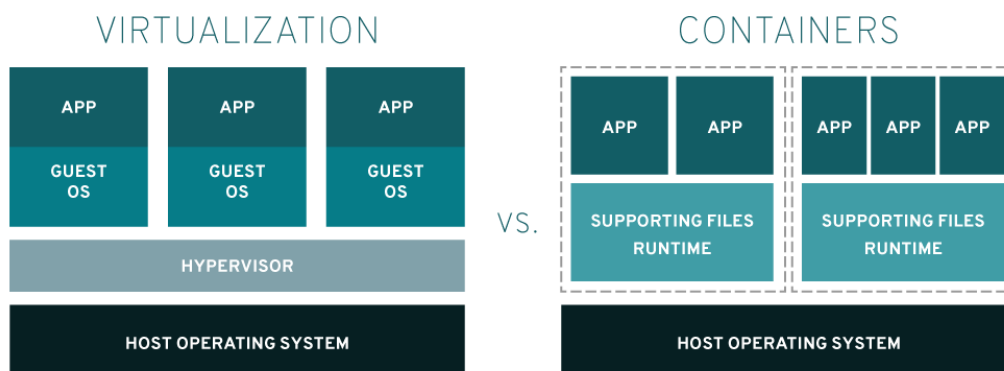


Figura 2.2: Virtualización vs Contenerización. Extraída de [62].

La combinación de estas tecnologías provee gran flexibilidad al realizar el despliegue de aplicaciones sobre varios servidores, complementando lo ligero de los contenedores con el aislamiento de recursos físicos y la seguridad obtenidos mediante VMs.

Citando la ‘Application Container Security Guide’ (SP 800-190 de NIST [64]): “A pesar de que a veces se considera que los contenedores son la siguiente fase de virtualización, sobrepasando la virtualización de hardware, la realidad de la mayoría de las organizaciones apunta menos a la revolución que a la evolución. Los contenedores y la virtualización de hardware no solo pueden, sino que frecuentemente lo hacen, coexistir y heredar las capacidades del otro. Las VMs brindan muchos beneficios, tales como fuerte aislamiento, automatización de SO, y un amplio y profundo ecosistema de soluciones. Las organizaciones no necesitan tomar una decisión entre contenedores y máquinas virtuales. Por el contrario pueden continuar utilizando VMs para desplegar, particionar y administrar su hardware, mientras que utilizan contenedores para empaquetar sus aplicaciones, aprovechando cada VM más eficientemente.” Es por esto que ambas tecnologías serán utilizadas más adelante en la puesta en marcha de un data-center de prueba mediante Openstack.

LXC [4][62] Un “Linux container” es un contenedor formado por un conjunto de procesos que se encuentran aislados del resto del sistema host. Como tal, brinda un entorno virtual con su propia CPU, memoria, red, etc, implementado mediante el uso de los namespaces y cgroups del kernel linux corriendo en la máquina host.

Este tipo de contenedores es utilizado por Openstack durante su despliegue para ejecutar los servicios en cuya configuración se haya indicado que utilicen contenedores en lugar de correr directo sobre el servidor.

2.4. Datacenters

La infraestructura que se encuentra por debajo de la mencionada nube se conoce con el nombre de Datacenter. Un Datacenter [5] es un espacio físico que aloja múltiples componentes de hardware interconectados tales como servidores, racks, switches, routers, sistemas de almacenamiento, etc. Estos

últimos proveen una red de recursos de red, cómputo y almacenamiento, necesaria para alojar diversas aplicaciones o grandes cantidades de datos. A su vez, los nuevos Datacenters escalan implementando infraestructuras virtualizadas, utilizando los mecanismos mencionados anteriormente, por encima de la física ya existente llegando a interconectar múltiples espacios físicos ubicados en diversas partes del mundo.

Debido al gran potencial computacional existente en este tipo de infraestructuras, su mayor explotación se encuentra en la ejecución de tecnologías tales como big data, inteligencia artificial, aprendizaje automático, entre otras.

Por su gran importancia en la tecnología de la información actual, los Datacenters no solo requieren un diseño de infraestructura de recursos computacionales sino también un diseño de componentes físicos externos que garanticen la seguridad física y una tasa de resistencia a fallas prácticamente perfecta. En función de estos aspectos es que existe un estándar internacional especificado por la ANSI que califica y certifica el diseño de un datacenter. Existen entonces cuatro categorías bajo el estándar ANSI/TIA-942 que se resumen a continuación:

- **TIER 1:** especificado para pequeñas empresas y organizaciones con una infraestructura básica. Ofrece una escasa protección ante riesgos físicos externos, sin la implementación de ninguna redundancia.
- **TIER 2:** se corresponde a Datacenters que posean un mínimo nivel de redundancia a nivel de componentes pero no de distribución eléctrica. Además aumentan su protección en cuanto a eventos físicos, en comparación con el nivel anterior.
- **TIER 3:** suele indicarse para organizaciones que requieren un servicio con disponibilidad 24/7. Mantiene redundancia tanto a nivel de componentes como de distribución eléctrica. Tolera prácticamente cualquier tipo fallas físicas. Para mantener el sistema (ej: recambio de componentes) no es necesario paralizarlo.
- **TIER 4:** enfocado a grandes organizaciones mundiales. Mantiene el mayor nivel de tolerancia a fallas junto con redundancia de componentes y distribuciones eléctricas. Es posible que ocurra un mantenimiento del sistema junto con una falla inesperada sin que el servicio se vea afectado.

2.5. Redes

En la etapa de configuración e instalación y posteriormente en la utilización de Openstack, se refieren diversos conceptos de red, por ejemplo, protocolos, tipos de redes y componentes virtualizados. A continuación se introducirán brevemente algunos de estos conceptos.

Flat Una red Flat hace referencia a una red en la cual no se utiliza ningún tipo de tag. Las interfaces físicas o virtuales se asocian directamente al switch o bridge por lo tanto solamente una red flat puede existir por cada interfaz física.

VLAN Una Virtual Local Area Network (VLAN) permite segmentar de manera virtual un dominio de difusión de capa 2 en múltiples dominios de difusión. Esto permite utilizar un switch como si fuera múltiples switches. A modo de ejemplo, dos host conectados a un mismo switch pero en distintas VLANs no podrán ver el tráfico generado por el otro. Para identificar a qué VLAN corresponde una trama se utiliza un nuevo campo en el cual se introduce el ID de la VLAN, estos cambios que se realizan a la trama Ethernet se establecen en el estándar IEEE 802.1Q [1].

Openstack utiliza las VLANs con el fin de aislar el tráfico de datos de diferentes clientes, sin importar en qué servidor físico (nodo de cómputo) estén corriendo las máquinas de los mismos [19].

VXLAN El protocolo Virtual extensible local area network (VXLAN) se ubica dentro de los protocolos de superposición (overlay protocols) que utilizan el mecanismo de tunelización para el transporte de datos. VXLAN encapsula una trama Ethernet dentro de paquetes UDP los cuales pueden ser ruteados. Esto permite extender una red local sobre múltiples redes de capa 3 en forma transparente para los host finales. El funcionamiento del protocolo se encuentra en el RFC 7348 [12]. Para diferenciar las distintas redes virtuales en lugar de utilizar un VLAN ID se utiliza un VXLAN Network Identifier (VNI), el cual puede tomar aproximadamente 16 millones de valores siendo una de las principales diferencias con las VLANs que pueden tomar 4096 valores únicos. Esta diferencia para datacenters de gran porte es vital para poder aislar el tráfico de los clientes del mismo. Un componente necesario para encapsular y desencapsular son los VXLAN Tunnel Endpoint (VTEP) los cuales residen en los nodos físicos. Un listado con pros y contras de las redes de capa 2 y capa 3 se presenta en [43].

3

Openstack

3.1. Origen y definición

Openstack fue creado en los primeros meses del 2010. En ese momento Rackspace quería reescribir el código de infraestructura que corría en sus servidores cloud y además estaban considerando hacer open source el código cloud existente. En ese entonces, Anso Labs, quienes trabajaban para la NASA, publicaron el código beta para Nova, un proyecto basado en Python descrito como “cloud computing fabric controller”. Dadas las semejanzas en las necesidades de ambas empresas, decidieron unir fuerzas dando como resultado una base de Openstack.

El primer Summit de diseño tuvo lugar en Austin, TX del 13 al 14 de julio de 2010, en donde participaron aproximadamente 25 compañías: AMD, Autonomic Resources, Citrix, Cloud.com, Cloudkick, Cloudscaling, CloudSwitch, Dell, enStratus, FathomDB, Intel, iomart Group, Limelight, Nicira, NTT DATA, Opscode, PEER 1, Puppet Labs, RightScale, Riptano, Scalr, SoftLayer, Sonian, Spiceworks, Zenoss y Zuora. El proyecto fue oficialmente anunciado el 21 de julio de ese mismo año.

Originalmente la misión de Openstack era “to produce the ubiquitous Open Source Cloud Computing platform that will meet the needs of public and private clouds regardless of size, by being simple to implement and massively scalable”. Luego en 2016 se actualizó la misma incluyendo interoperabilidad y mejor servicio a los usuarios finales. En septiembre de 2012, fue lanzada la fundación de Openstack como una institución independiente proporcionando recursos para proteger, potenciar y promover el software Openstack y la comunidad que lo rodea [32]. Como es definido en el sitio de Openstack, *“es un sistema operativo en la nube que controla grandes grupos de recursos de computación, almacenamiento y redes a través de un centro de datos, administrados y provisionados a través de APIs”*[50].

Openstack es un software open source gobernado por una fundación. Ser parte de la misma es gratis y está abierta a todo el mundo. El proyecto tiene una arquitectura modular, en donde cada instalación de Openstack tendrá instalados y configurados los módulos que se ajusten a las necesidades del caso. Dichos módulos están implementados en Python. Seis de estos proyectos se denominan como módulos core dado que se encargan de las funciones principales del cloud como son las conexiones de red, el almacenamiento, el servicio de identidad, servicios de imágenes y de cómputo. Permite construir nubes públicas y privadas ofreciendo principalmente servicios de infraestructura (IaaS) y en un grado menor, servicios de plataforma (PaaS).

3.2. Módulos Core

En esta sección se describirán los módulos Nova, Neutron, Glance, Cinder, Keystone y Swift, llamados core, profundizando en los aspectos más importantes de cada uno. Conceptualmente los módulos se relacionan de la siguiente forma:

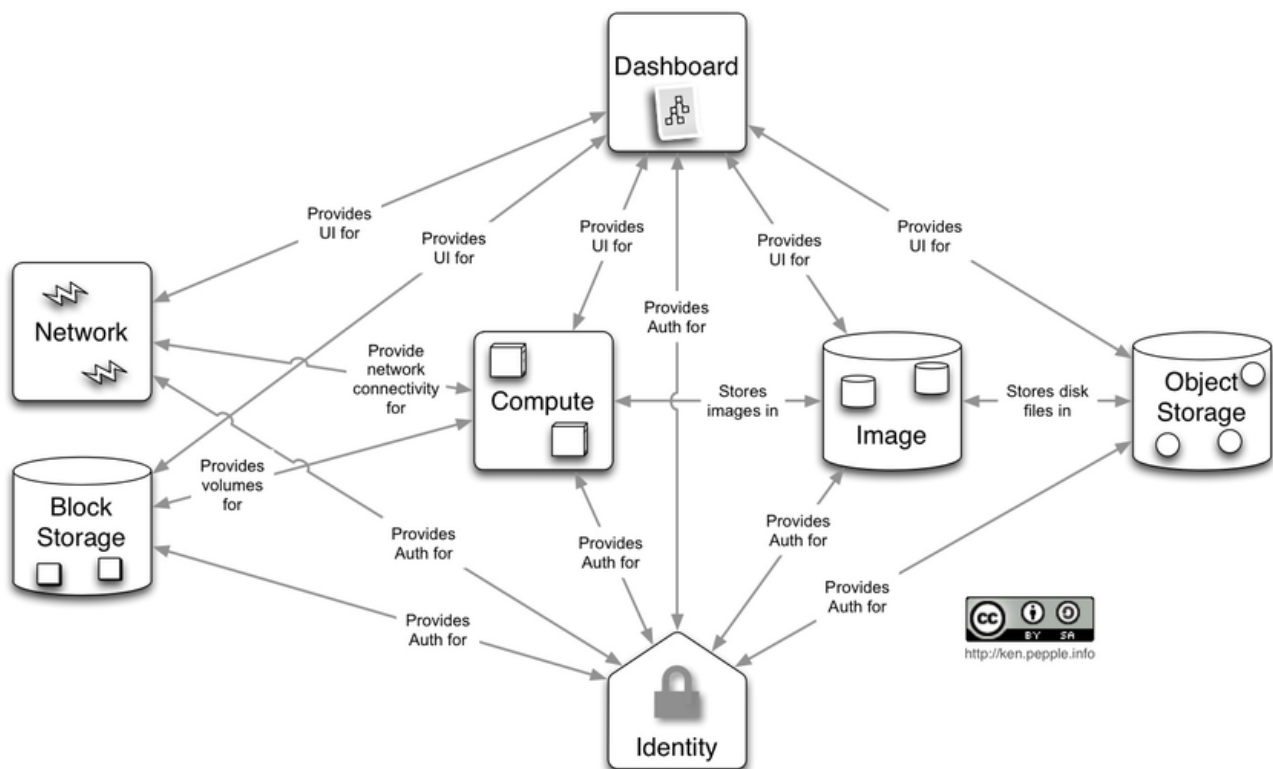


Figura 3.1: Relacionamiento entre módulos core

En [28] se muestra un esquema de una arquitectura lógica estándar de Openstack, en donde se puede apreciar las interacciones internas entre los componentes de un proyecto, las interacciones entre proyectos y las interacciones entre agentes externos y openstack.

3.2.1. Keystone

Keystone es el servicio de identidad que utiliza Openstack para la autenticación y autorización. El mismo se organiza como un conjunto de servicios internos, que se exponen en uno o varios endpoints, listados a continuación:

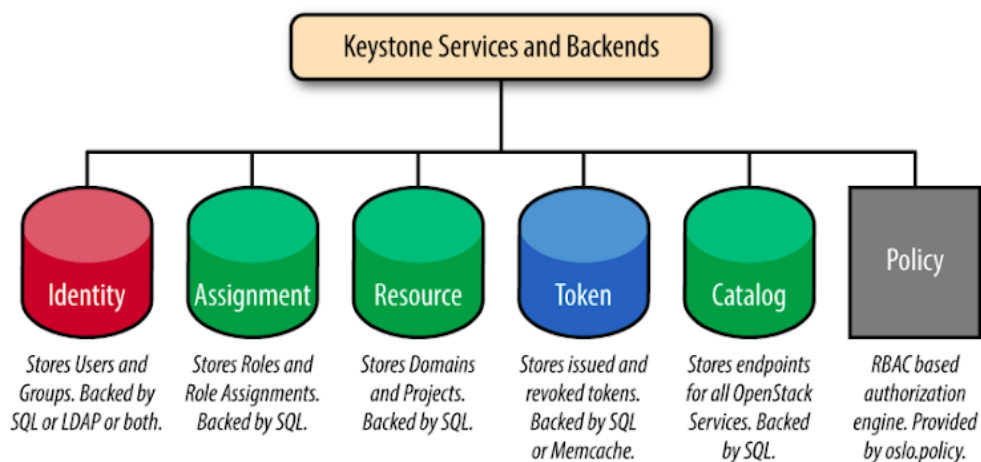


Figura 3.2: Servicios y backends soportados por Keystone. Extraída de [13].

- El **servicio de identidad** autenticación e información de usuarios y grupos. En una instalación estándar este servicio se encarga de todas las operaciones referentes a estos datos, sin embargo

en instalaciones más complejas se pueden configurar distintos backends para llevar a cabo estas tareas. Un ejemplo de esto puede ser LDAP.

- Un usuario representa a un consumidor individual de la API el cual necesariamente debe pertenecer a un dominio y es único dentro del mismo.
- Un grupo representa un conjunto de usuarios, los cuales al igual que los usuarios, deben pertenecer a un único dominio.
- El **servicio de recursos** provee información sobre proyectos y dominios.
 - Los proyectos representan la unidad base de propiedad en Openstack, debido a que todos los recursos deben pertenecer a un proyecto necesariamente. Los proyectos son únicos dentro de cada dominio.
 - Los dominios son grandes contenedores para los proyectos, grupos y usuarios. Por defecto Openstack crea el dominio “Default”. Dada la naturaleza de los dominios los mismos pueden ser utilizados para delegar la administración de los recursos.
- El **servicio de asignación** provee información sobre roles y asignaciones.
 - Un rol indica el nivel de autorización que un usuario final puede obtener. Un rol se puede otorgar a nivel de dominio o proyecto. Por otro lado un rol puede ser asignado a nivel de usuario o grupo.
 - Las asignaciones de roles son tripletas que contienen un rol, un recurso y una identidad.
- El **servicio de Tokens** válida y administra los tokens utilizados para las solicitudes de autenticaciones enviadas luego que las credenciales del usuario fueron validadas.
- El **servicio de catálogo** utilizado para el descubrimiento de endpoints.

Las definiciones mencionadas fueron extraídas de [35] y [14].

3.2.2. Nova

Nova es el proyecto que se encarga de proveer una forma para provisionar instancias o servidores virtuales. El proyecto soporta la creación de imágenes, servidores baremetal con la ayuda del proyecto ironic y un soporte limitado para el manejo de containers.

Nova se compone por un conjunto de demonios que permiten ofrecer los servicios mencionados. Adicionalmente para poder desarrollar las funciones básicas, este módulo requiere de los siguientes servicios core de openstack: keystone, glance, neutron y placement.

En la figura 3.3 se puede apreciar un esquema conceptual de los principales componentes de una instalación de nova estándar. Un dato relevante sobre los componentes de nova es que pueden ser escalados horizontalmente, siendo independiente el grado de escalado para cada servicio.

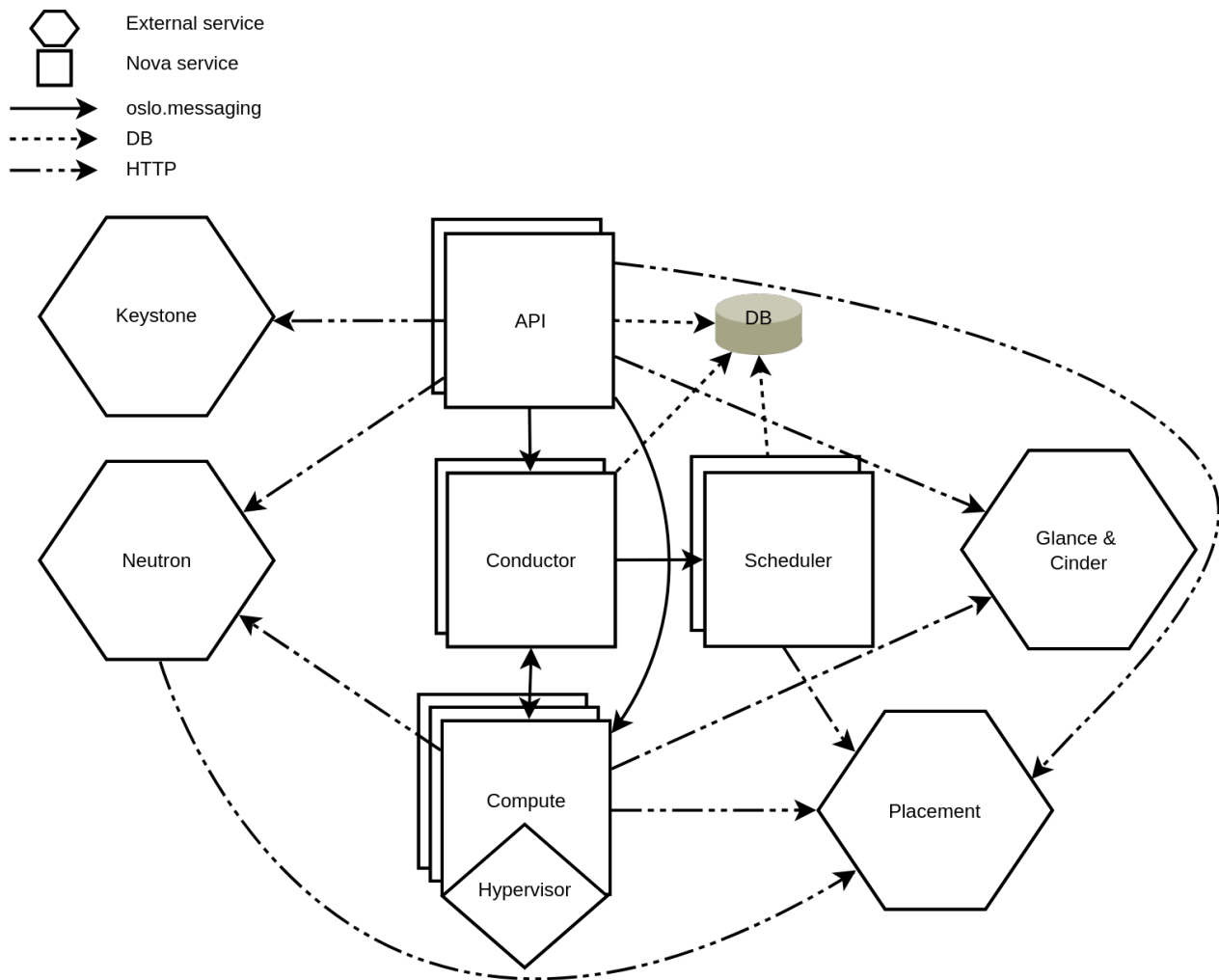


Figura 3.3: Principales componentes de Nova. Extraída de [44].

Internamente los componentes de nova se comunican mediante RPC mientras que la interfaz de cara al usuario es una API REST. Los principales componentes de nova son:

API Se encarga de recibir y responder las solicitudes HTTP y comunicarse con los otros componentes de nova mediante la cola de mensajes.

Scheduler Se encarga de decidir en qué host de cómputo se aloja cada instancia. Para tomar estas decisiones existen diversos algoritmos que pueden ser configurados, siendo algunos ejemplos el algoritmo simple donde selecciona el host con menor carga actual, chance en donde se elige un nodo de forma randómica, entre otros. Esta tarea puede ser muy sencilla como es el caso del algoritmo random o muy complicada para los casos donde se requiera realizar un uso eficiente de los recursos [57].

Compute El módulo nova-compute es principalmente un demonio que se encarga de administrar la comunicación con el hypervisor y con las máquinas virtuales. Esto lo lleva a cabo mediante las APIs del hipervisor. Algunos ejemplos de APIs son: libvirt para KVM o QEMU, XenAPI para XenServer y VMwareAPI para VMware.

Haciendo a un lado la complejidad del módulo, básicamente el demonio acepta acciones de la cola de mensajes para luego realizar una serie de comandos contra la API del hipervisor. Además se encarga de actualizar el estado de la base de datos [22].

Conductor En las primeras versiones de Openstack, todos los servicios del componente de cómputo tenían acceso directo a la base de datos hosteada en el nodo controlador. Esto presentaba dos grandes problemas: seguridad y performance. En lo que respecta a la seguridad, en el caso que un nodo de cómputo sea vea comprometido, el atacante tendrá acceso a la base de datos de Openstack. Por el lado de la performance, las llamadas realizadas desde el nodo de cómputo soportan un único hilo y son bloqueantes. Esto generaba un cuello de botella debido a no poder paralelizar las invocaciones.

Para mejorar estos dos aspectos se introdujo el servicio nova-conductor el cual se presenta como una capa por encima del servicio de cómputo. Nova-compute en lugar de acceder directamente a la BD, delega la responsabilidad al servicio nova-conductor. En otras palabras este servicio actúa como un proxy para el servicio nova-compute.

El problema de seguridad se resuelve dado que los nodos de cómputo dejan de acceder directamente a la BD y el de performance se resuelve gracias a que el servicio de nova-conductor es no bloqueante, permitiendo realizar múltiples invocaciones en paralelo [26]. Por los motivos de su existencia, este servicio no se debe instalar en los nodos de cómputo. [58].

Además puede servir como un lugar donde centralizar y permitir administrar las operaciones que involucran al scheduler y el servicio de computo cómo construir, cambiar el tamaño o migrar instancias. Esto se realiza con el fin de separar las responsabilidades entre los servicios de nova. En [58] se muestra un ejemplo de los beneficios que este cambio presenta.

Placement Este servicio se presenta como una API REST que se encarga de realizar un seguimiento de los proveedores de recursos. Los recursos pueden ser de distintas clases. A modo de ejemplo un proveedor puede ser un nodo de cómputo o un pool de storage.

3.2.3. Neutron

Neutron es el proyecto encargado de proveer y administrar recursos de red en una nube creada con Openstack. Es un sistema escalable horizontalmente y diseñado para añadirle diversos plugins con el fin de proporcionar nuevas funcionalidades. Como otros servicios de Openstack, Neutron requiere una base de datos para persistir las configuraciones de los elementos de red. El servicio de red de Openstack permite crear desde redes y subredes hasta topologías de red avanzadas las cuales pueden contener firewalls, balanceadores, VPNs entre otros.

Las instalaciones del módulo de red deben tener al menos un red externa, la cual no es una SDN definida dentro de Openstack sino que es una red accesible por fuera de Openstack. Estas redes permiten que las redes virtuales construidas con neutron tengan conectividad con el mundo exterior. Por otro lado, neutron permite crear redes internas a las cuales se conectarán directamente las VMs. Para lograr que dichas VMs se conecten con las redes externas son necesarios routers que relacionen ambos tipos de redes.

Una arquitectura simplificada de Neutron se puede ver en la figura 3.4.

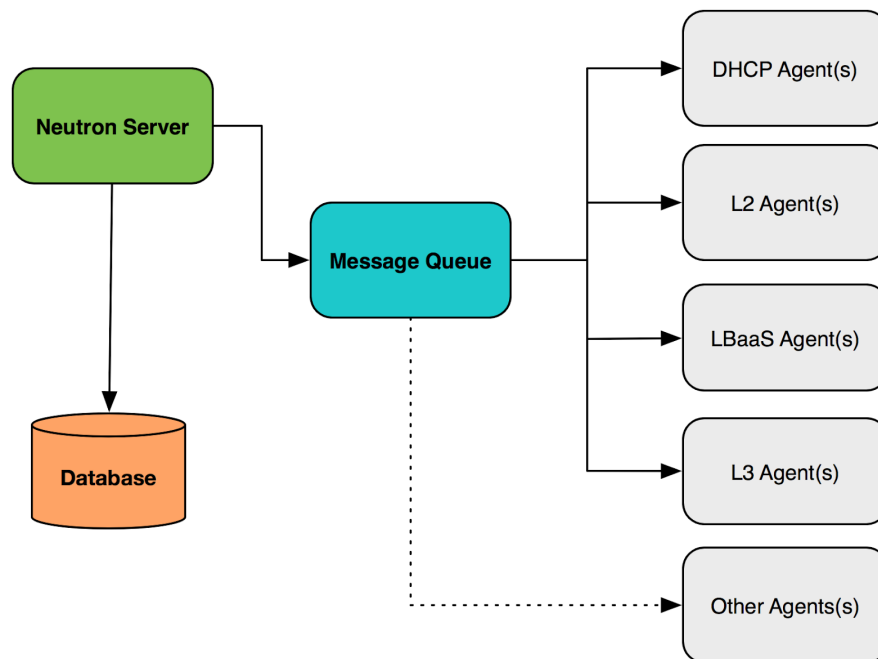


Figura 3.4: Extraída de [6]

Neutron-server El módulo Neutron server es en donde reside la API recibiendo y respondiendo las solicitudes de recursos de red. El servidor se comunica con la base de datos para almacenar las configuraciones existentes como se mencionó previamente.

Plugins y agentes Por otro lado se encuentran una serie de agentes que se distribuyen típicamente en los nodos de red y cómputo. Estos agentes se agregan a demanda según las funcionalidades y rendimiento requeridos. Estos agentes o plugins serán los encargados de crear los recursos virtuales de red del cloud.

Cola de mensajes En general se utiliza en las instalaciones del módulo de red para llevar a cabo la comunicación entre el neutron-server y los distintos agentes.

En la figura 3.4 se muestra como el servidor de Neutron se conecta con la base de datos que es donde se persisten los recursos de red creados. Por otro lado el servidor acepta solicitudes en la API que expone desde los usuarios y servicios.

Tipos de redes en Openstack

En Openstack los usuarios tienen la posibilidad de crear su propio esquema de red, decidiendo libremente el direccionamiento IP a utilizar. Los recursos de red creados dentro de un proyecto son privados al mismo. En Openstack se pueden crear dos tipos de redes:

- **Project/tenant network:** es una red virtual creada por un proyecto o por un administrador en nombre del proyecto. Este tipo de red se encarga de proveer conectividad a los recursos de un proyecto. Los usuarios pueden crear, modificar y eliminar redes tenants. En general este tipo de redes está aislado del resto de las redes de proyecto por VLANs o algún otro mecanismo de segmentación como VXLAN.
- **Provider network:** es una red virtual creada para mapearse a una red física de los servidores que alojan Openstack. Este tipo de redes son creadas para habilitar el acceso a recursos de red externos a la nube de Openstack. Las mismas son creadas y administradas por los administradores.

Cuando se crea una red de proyecto los aspectos físicos de como es implementada son transparentes al usuario, por lo contrario en la red de proveedor se puede especificar el tipo de red, la interfaz física y el identificador de segmentación utilizado.

Tipo de tráfico

El tráfico en Openstack se puede dividir en dos categorías: plano de control y plano de datos. El plano de control está relacionado con el tráfico utilizado para administración de los nodos físicos o contenedores, para las API de los servicios de Openstack y todo el tráfico que no esté relacionado con las instancias virtuales de Openstack. El plano de datos está relacionado con el tráfico generado o dirigido hacia instancias virtuales.

En ambientes de producción es recomendable utilizar interfaces físicas diferentes para los distintos tipos de tráfico [7]. La decisión de colapsar todo el tráfico en una sola interfaz y segmentarlo con VLANs o utilizar múltiples interfaces depende de las necesidades de cada caso. Al utilizar una sola interfaz física las probabilidades de fallas de red aumentan.

3.2.4. Glance

Glance es el nombre del proyecto utilizado por Openstack para la administración de imágenes. Este servicio permite a los usuarios, a través de una API RESTful, gestionar tanto las imágenes de las máquinas virtuales como la metadata asociada a estas. Típicamente esta gestión involucra el descubrimiento, registro y obtención de los recursos mencionados.

La implementación del servicio Glance se basa en una arquitectura cliente-servidor, comprendida por los siguientes componentes principales:

- **Cliente:** es quien realiza pedidos a través de la API REST provista.
- **API REST:** permite exponer todos los servicios ofrecidos por Glance.
- **Database Abstraction Layer:** es una API interna encargada de comunicar el servicio glance con la base de datos.
- **Glance Domain Controller:** se trata de un middleware que implementa las principales funcionalidades (autorización, notificaciones, políticas, conexiones a la base de datos).
- **Glance Store:** formado por un conjunto de drivers que permiten comunicar Glance con los diferentes backends de almacenamiento posibles.

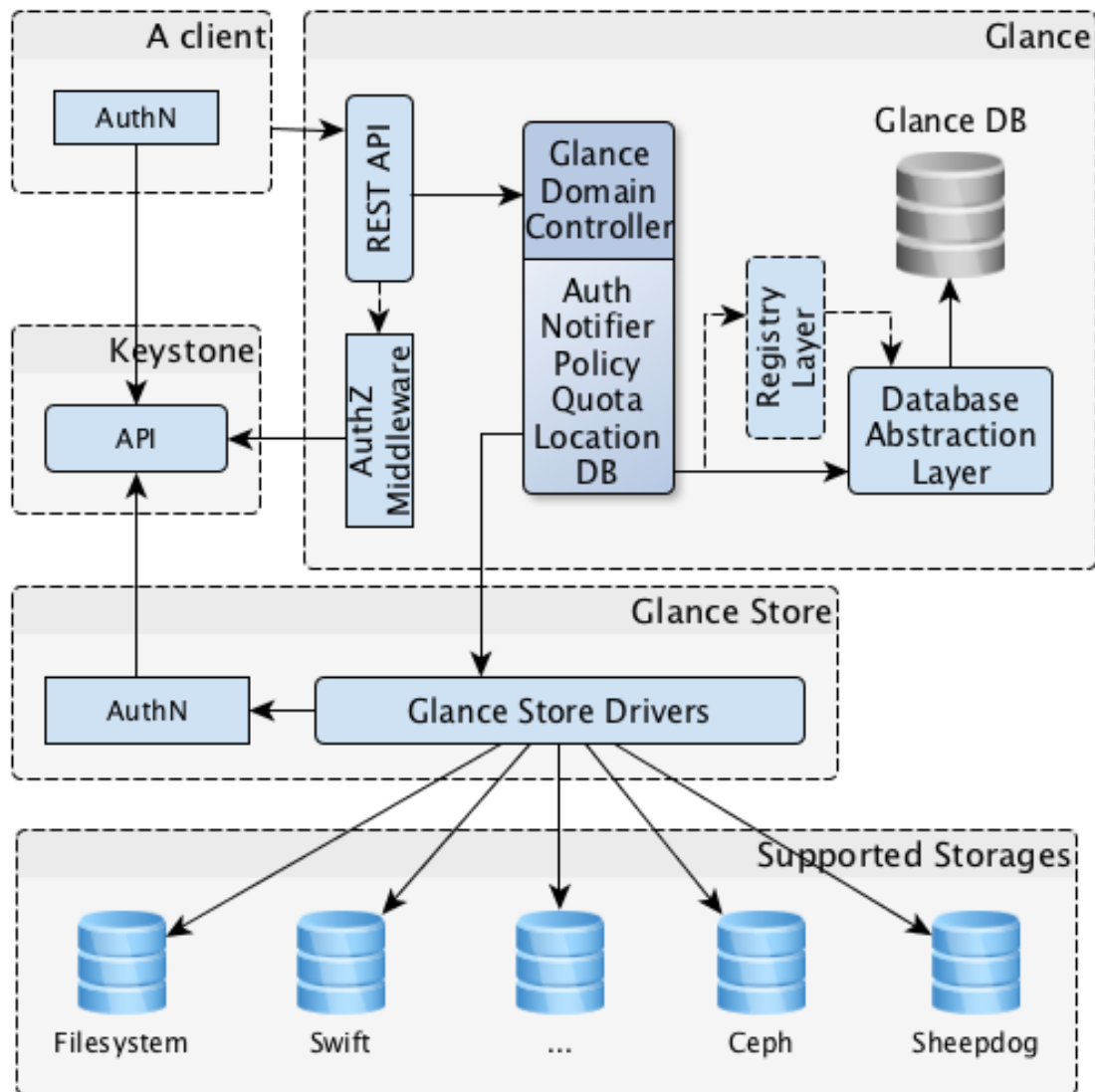


Figura 3.5: Componentes del módulo Glance. Extraída de [18].

Las imágenes son un concepto fundamental en Openstack debido a que son necesarias para crear instancias. Una instancia es una determinada máquina virtual que corre físicamente en un nodo de cómputo específico, siendo creada a partir de una imagen. Además de la imagen es necesario seleccionar un sabor (flavor) que determina las especificaciones de los recursos virtuales asignados a la VM, como pueden ser cantidad de CPUs, MB de memoria RAM y espacio en disco.

Creación de una VM Cuando se lanza una VM, en general y a grandes rasgos se llevan a cabo los siguientes pasos:

1. Se selecciona una imagen administrada por Glance y un flavor que indica tanto los recursos de cómputo necesarios a ser instanciados por Nova como de almacenamiento gestionados por Cinder.
2. El nodo de cómputo copia la imagen de base copiada desde Glance hacia el disco principal de la VM conectándose en forma remota con el servicio de Cinder.
3. El nodo de cómputo provee los recursos virtuales como vCPUs y memoria.
4. La instancia levanta y comienza a ejecutar. Cualquier modificación almacenada en el disco de la

instancia no altera la imagen de base original que seguirá estando disponible para lanzar nuevas instancias.



Figura 3.6: Extraída de [31].

Cuando la instancia deja de existir, se liberan todos los recursos con excepción del almacenamiento persistente en Cinder.

Cada imagen tiene asociada su metadata, que incluye conjunto mínimo de propiedades [20]:

- **architecture**: indica la arquitectura de CPU que debe ser soportada por el hipervisor.
- **instance_uuid**: en caso de tratarse de una snapshot de una instancia, es utilizada para determinar con cual de ellas se encuentra asociada.
- **kernel_id**: el identificador de la imagen que puede ser utilizada como kernel cuando se inicia una imagen AMI (Amazon Machine Image).
- **ramdisk_id**: el identificador de la imagen que puede ser utilizada como ramdisk cuando se inicia una imagen AMI.
- **os_distro**: el nombre común de la distribución del sistema operativo (en minúsculas).
- **os_version**: la versión del sistema operativo, especificada por el distribuidor.

La lista completa de propiedades se encuentra en [53]. Una de las características principales a especificar es el formato de disco o contenedor. El formato de disco de imagen de VM no es más que el formato de la imagen de disco subyacente, este puede ser alguno de los siguientes: raw, vhd, vhdx, vmdk, vdi, iso, ploop, qcow2, aki, ari, ami. Por su parte, el formato de contenedor refiere a aquellos formatos de imágenes que incluyen metadata en sí mismos, pueden ser: bare, ovf, aki, ari, ami, ova, docker.

3.2.5. Cinder

El servicio de almacenamiento de bloques proporciona administración de almacenamiento de bloques persistente para discos duros virtuales. Las operaciones básicas que permite realizar son:

- Crear, listar y eliminar volúmenes.
- Crear, listar y eliminar snapshots.
- Atachear y desatachar volúmenes a máquinas virtuales.

Los volúmenes son utilizados como una solución para persistir los datos de una instancia incluso luego de destruir la misma. Los volúmenes pueden ser asignados a una instancia a la vez.

A continuación se muestra la arquitectura de alto nivel de los componentes de Cinder y cómo interactúan:

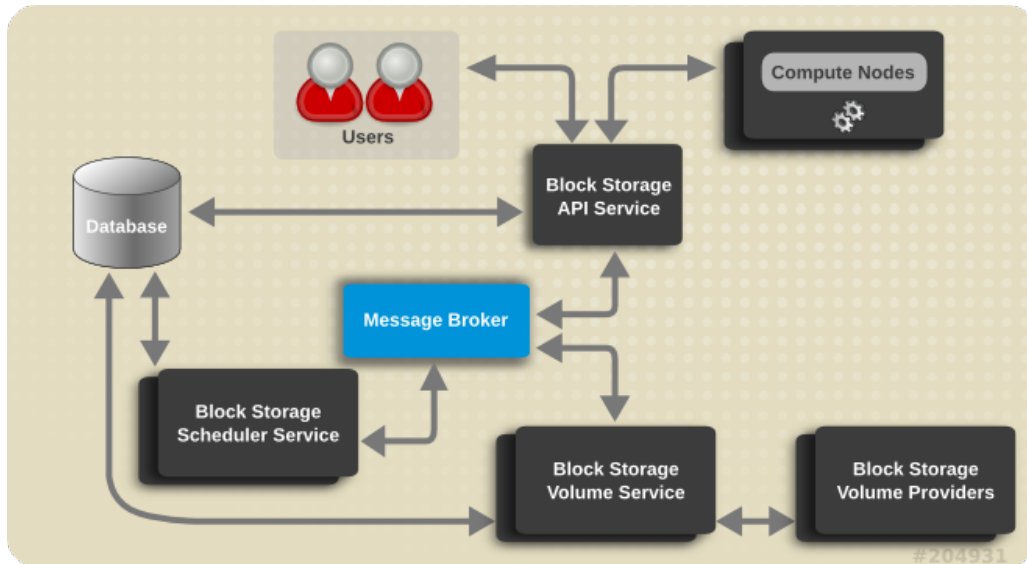


Figura 3.7: Principales componentes de Cinder. Extraído de [59].

El servicio **cinder-volume** administra la interacción con los dispositivos de almacenamiento de bloque. Al recibir una nueva solicitud del scheduler, el servicio crea, modifica o elimina volúmenes a demanda.

Este servicio incluye en su repositorio un conjunto de drivers para soportar diversos dispositivos de almacenamiento para proveer volúmenes. La instalación por defecto utiliza volúmenes locales administrados por Logical Volume Manager (LVM). Además los drivers pueden soportar diversos protocolos de transporte, en el caso de LVM son iSCSI y iSER [37]. En [54] muestran un listado de los drivers disponibles.

El servicio **cinder-api** recibe y responde las solicitudes del módulo. Este servicio al recibir un nuevo mensaje se encarga en primer lugar de verificar que se cumplan los requerimientos de identidad (tokens), luego en base al mensaje construye uno nuevo indicando las acciones a realizar sobre los volúmenes. El mensaje se envía al broker de mensajes para ser procesado por otros servicios del bloque de almacenamiento.

El servicio **cinder-backup** permite crear backups de los volúmenes a repositorios de almacenamiento externos al proyecto, como por ejemplo realizar los respaldos en Swift.

El servicio **cinder-scheduler** planifica y realiza el ruteo de las solicitudes a los servicios cinder-volume apropiados bajo un criterio definido. Dicho criterio puede ser sencillo como realizar un round robin entre los distintos servicios de volúmenes o ser más sofisticado utilizando filtros de planificación. Estos filtros pueden ser fijados sobre la capacidad, el tipo de volumen, las capacidades funcionales, entre otros.

3.2.6. Swift

Se trata del componente de almacenamiento de objetos de Openstack, implementado mediante un sistema distribuido de alta disponibilidad y accesible a través de una API REST. Gestiona el almacenamiento a largo plazo de grandes cantidades de información utilizando redundancia de datos en clusters bajo una arquitectura que evita tener un único punto de control, permitiendo escalar fácilmente.

Swift maneja una jerarquía en tres niveles para organizar el almacenamiento de objetos [45]:

- **Account:** la cuenta es el nivel mas alto en la jerarquía, creando un namespace en el cual residen las instancias del siguiente nivel. A nivel de Openstack una cuenta está dada por un proyecto o usuario tenant.
- **Container:** no es mas que un contenedor de objetos, pero permite gestionar un control de acceso a estos mediante ACLs (no es posible tener ACLs directamente sobre objetos). Como contenedor, define un namespace para los objetos que almacena.
- **Objeto:** es el contenido a almacenar propiamente dicho, como documentos o imágenes, incluyendo la metadata asociada a estos. Todos los objetos tienen una URL que los identifica para poder ser accedidos.

La siguiente imagen ilustra la arquitectura del servicio Swift, en la cual se pueden identificar componentes clave que serán definidos a continuación [21][51].

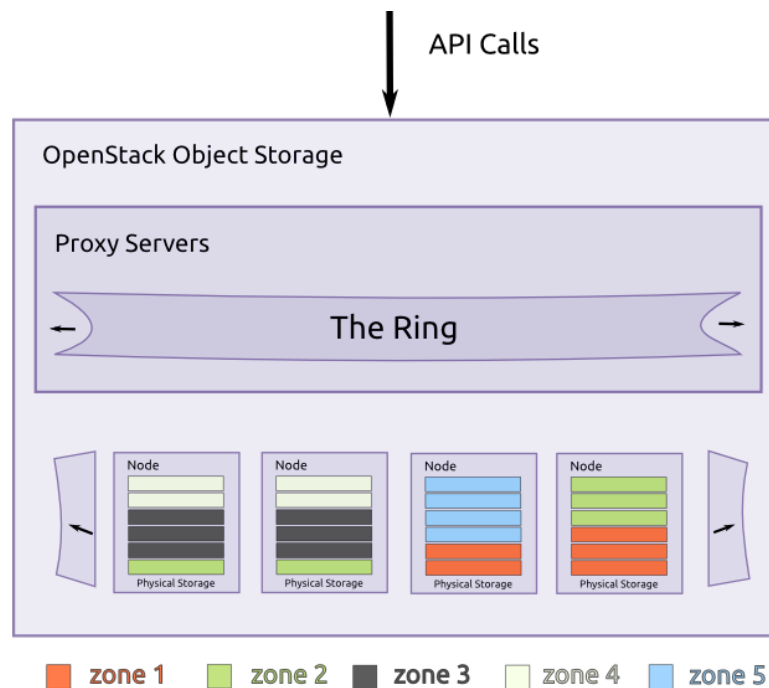


Figura 3.8: Arquitectura del módulo Swift. Extraída de [21].

Principales componentes

- **Proxy servers:** se encargan de comunicar los diferentes componentes del servicio Swift y brindan un acceso público a los usuarios, manejando todos los pedidos que llegan a la API. Para cada uno de los pedidos, busca la ubicación del elemento (cuenta, contenedor u objeto) dentro del anillo para luego acceder al nodo de almacenamiento adecuado.

- **Rings:** un anillo simboliza el mapeo entre los nombres de los elementos almacenados en el cluster y sus ubicaciones físicas, existiendo anillos separados para cada uno de los tipos de elementos. Cuando el sistema debe realizar una operación sobre un elemento debe interactuar con el anillo correspondiente para poder accederlo. A su vez, los anillos gestionan:
 - zonas: para generar aislamiento de información.
 - dispositivos: determina qué dispositivos se encuentra en uso y cuáles utilizar en caso de falla.
 - particiones: distribuidas en forma balanceada en todos los dispositivos.
 - réplicas: cada partición es replicada al menos 3 veces, para no tener un único punto de falla.
- **Zones:** las zonas son utilizadas para aislar la información almacenada y evitar una pérdida total en caso de fallas. Cada réplica de las mencionadas, intenta ser almacenada en una zona diferente. Cada zona puede ser representada desde un solo disco físico separado, hasta racks servidores completos.
- **Accounts and containers:** anteriormente se definieron en forma conceptual, pero como componentes cada cuenta o contenedor es implementado mediante una base de datos SQLite distribuida a lo largo del cluster.
- **Partitions:** cada partición puede estar compuesta por un conjunto de bases de datos de cuentas, bases de datos contenedores y objetos propiamente dichos. Se trata de un agrupamiento de elementos para poder ser trasladados dentro del cluster, facilitando operaciones de replicación.

3.3. Tipos de nodos

En general los sistemas de Openstack están contruidos con servidores o nodos físicos, aunque también es posible utilizarlos virtualizados como es el caso de este trabajo, en donde se pueden agrupar en 4 categorías [8]:

Nodo de control Estos nodos en general corren las APIs de todos los servicios de los componentes de Openstack. Además estos nodos alojan la base de datos de los módulos, los servidores de mensajes y los componentes de caché en memoria como Memcache. Para escalar horizontalmente las APIs pueden ser instaladas en varios nodos de control pudiendo adicionalmente balancear la carga.

Nodo de red Estos nodos en general corren los servicios de metadatos y DHCP. Además permiten crear routers virtuales cuando el agente de Neutron L3 es instalado. Al igual que los nodos de control, para mejorar el rendimiento y escalar horizontalmente se pueden separar los servicios de red entre los distintos nodos de red. El uso de nodos de red dedicados mejora la seguridad y resistencia, dado que los nodos de control tendrán menor riesgo de que se sature la red y sus recursos.

Un ejemplo de como quedan organizados los componentes de Neutron al utilizar nodos de control, red y computo se muestra a continuación:

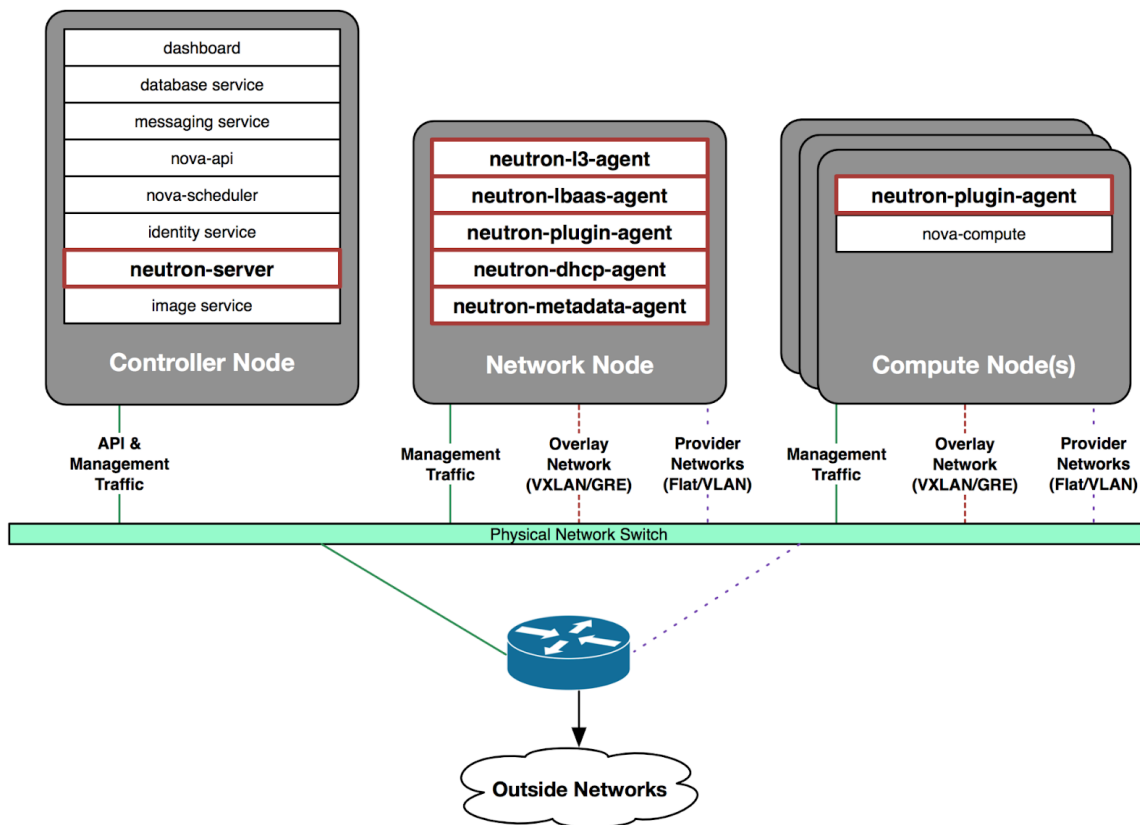


Figura 3.9: Arquitectura de Neutron. Extraída de [8].

En esta arquitectura, la API de Neutron es instalada en el nodo de control, los agentes encargados de realizar tareas específicas son instalados en un nodo dedicado de red y por último, cada nodo de cómputo tiene un agente de red encargado de brindar conectividad a las instancias que aloja.

Nodo de cómputo Estos nodos en general corren un hipervisor como puede ser KVM, Hyper-V o Xen; o por el lado de los contenedores LXC o Docker.

Nodo de almacenamiento Estos nodos en general se limitan a correr software que esté relacionado con los proyectos como Cinder, Ceph o Swift. En estos nodos no es común alojar servicios de otro tipo como de red o cómputo.

Nodo de balanceamiento de carga Estos nodos son fundamentales para el funcionamiento óptimo de una instalación de Openstack, dado que se espera que los servicios que se ofrecen tengan una alta disponibilidad. Como la forma de mejorar el rendimiento es escalar horizontalmente, resulta vital tener nodos que se encarguen de distribuir la carga entre los mismos.

3.4. Servicios de infraestructura

Estos servicios son transversales a todos los módulos del sistema y son mandatorios para el funcionamiento de Openstack. En general estos servicios son instalados en los nodos de control como fue mencionado en la sección anterior.

Galera - MariaDB La base de datos MariaDB se utiliza para almacenar el estado de todos los servicios de openstack, utilizando usualmente una base de datos MySQL. El servidor de base de datos en general se despliega con una configuración activo/pasivo en donde solamente el servidor principal

(activo) podrá ser utilizado por los servicios. Para poder utilizar un esquema activo/activo se puede utilizar Galera, el cual se define como un software de clusterización para MySQL, el cual utiliza un mecanismo sincrónico de replicación para lograr una alta disponibilidad [63].

Message queue Openstack utiliza una cola de mensajes para llevar a cabo la comunicación entre procesos. Los servicios de cola de mensaje que Openstack soporta son: RabbitMQ y Qpid. Ambos servicios son Advanced Message Queuing Protocol (AMQP) frameworks, los cuales proveen colas de mensajes para comunicaciones punto a punto. En general las colas de mensajes son desplegadas como pools de servidores centralizados o descentralizados. En la instalación realizada se utilizó RabbitMQ [41].

Memcached Es un sistema de caché de objetos en memoria distribuido, apuntado a mejorar el rendimiento de los sistemas mediante la reducción de carga a la base de datos. En Openstack este software es utilizado por el mecanismo de autenticación de keystone para cachear los tokens del sistema [40].

3.5. Métodos de instalación

Realizar la instalación básica de Openstack (módulos core) es una tarea sumamente compleja. Esto se debe a la gran cantidad de configuraciones y diversos tópicos en los cuales hay que tener un grado de entendimiento no menor, como en bases de datos, linux, redes y backends de almacenamiento. Las guías de instalación que se pueden encontrar en el sitio oficial de openstack consisten de cientos de configuración y comandos a ejecutar en donde es muy probable equivocarse y en consecuencia instalar incorrectamente los módulos de openstack.

Debido a la relevancia que ha tomado en los últimos años openstack, la amplia comunidad formada por decenas de compañías y personas buscaron caminos alternativos a realizar la instalación “manualmente”. Estas formas se basan en la automatización de las tareas. Para esto existen varias tecnologías como puppet [55] o Ansible [47]. Además existen diversas distribuciones de Openstack, como DevStack que permite armar un ambiente rápidamente para realizar pruebas, Packstack-RDO o TripleO. Finalmente existe una gran oferta de distribuciones comerciales donde podemos encontrar grandes compañías como IBM, Debian, DELL, Red Hat, VMware, Huawei, etc. Un listado más extenso se puede ver en [52].

Otra de las razones principales para utilizar herramientas de automatización además de facilitar la instalación es para poder mantener el sistema luego de su instalación. En el caso de realizar las tareas manualmente no se podrá escalar la nube a cientos o miles de servidores dado que sería prácticamente imposible de mantener o actualizar.

En el presente trabajo se emplea la herramienta de automatización Ansible dentro del proyecto Openstack-Ansible (OSA). Dicho proyecto se encarga de proveer playbooks y roles para deployar y configurar un ambiente de Openstack. OSA no es un proyecto que funcione simplemente con los archivos y configuraciones por defecto sino que modificaciones por parte del administrador del cloud serán necesarias. El resultado final que se obtiene con OSA es un cloud de Openstack probada para ambientes de cualquier tamaño, desde datacenters de testing hasta producción.

Antes de continuar profundizando en las particularidades de OSA se introducirá brevemente Ansible.

3.5.1. Ansible

Ansible es una herramienta para la automatización de tareas. Permite configurar sistemas, aplicaciones o dispositivos, desplegar o mantener software, entre otras tareas de IT. Algunos puntos a destacar son: su diseño simple orientado a que sea fácil de utilizar, el uso de OpenSSH como transporte y el diseño del lenguaje el cual es legible por humano y no requiere de conocimientos de programación. Estas características junto a que el software no tiene muchas dependencias son lo que potencian a utilizar OSA. A continuación se describirán los principales conceptos, necesarios para la utilización de Ansible:

Nodo de control El nodo de control será quien ejecute comandos o playbooks de Ansible. Los requisitos serán tener Python instalado y Ansible. En la arquitectura utilizada, como se menciona luego en el informe, el nodo de control es el de deploy, el cual contiene los diversos scripts de Ansible y es comunica con el resto de los nodos para instalar y configurar los módulos y componentes de Openstack. Este nodo puede ser uno de los utilizados como parte del datacenter o de uso exclusivo para la instalación.

Inventario Mantiene una lista de los nodos administrados. El inventario es un archivo en el cual se puede especificar la IP de cada nodo administrado, se pueden organizar los nodos en grupos para una mejor escalabilidad. En la siguiente sección se profundiza en este punto.

Módulos Todas las acciones que se pueden realizar con Ansible se llevan a cabo con un módulo. Estos son las unidades de código que se ejecutan. En cada tarea se puede ejecutar uno o más módulos. Una lista completa de los módulos se encuentra en [3].

Tarea Es la unidad de acción en Ansible.

Playbook Contienen una lista ordenada de tareas. Las playbooks se escriben en YAML lo cual beneficia la lectura, escritura y la comprensión de las mismas. Son una forma de organizar las tareas bajo un criterio determinado. En el caso de OSA utiliza varias playbooks para setear el ambiente inicial, instalar los componentes de infraestructura, entre otros que se especificarán en las próximas secciones.

3.6. Arquitectura

El método de instalación OSA emplea LXC para desplegar los servicios de Openstack. Además utiliza Linux bridges entre los contenedores y las interfaces físicas o lógicas del host con el fin de proveer conectividad directa a nivel de capa 2. El aislamiento de cada contenedor se logra mediante la utilización de namespaces, esto genera que se deban utilizar pares de interfaces virtuales (veth pairs) para tener conectividad entre los mismos. Esta implementación se puede ver en la imagen 3.10.

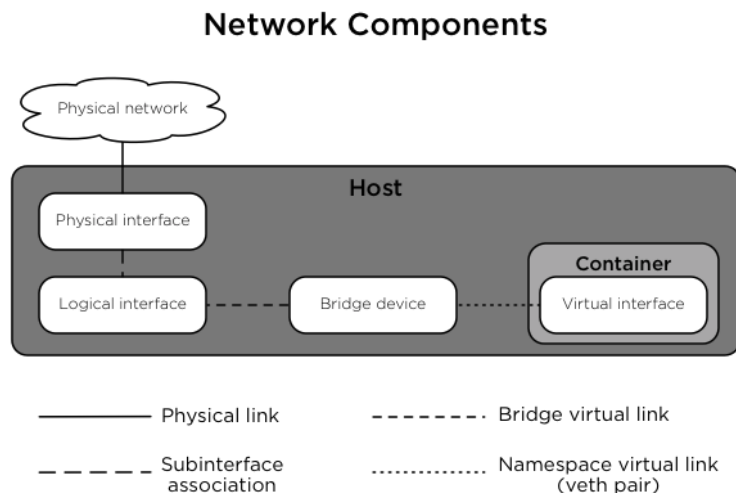


Figura 3.10: Componentes de red en Openstack. [25].

3.6.1. Arquitectura de red

OSA soporta múltiples arquitecturas de red, las cuales se diferencian según sea para un ambiente de producción o testing, cantidad de interfaces físicas de los servidores o módulos de openstack utilizados. Para los ambientes de producción es común utilizar interfaces bondeadas mejorando la disponibilidad de los servicios. En general para segmentar el tráfico, tanto en los casos donde se realiza bonding o en donde hay interfaces simples, se utilizan VLANs asignando un ID para cada subred utilizada dentro de Openstack. A continuación se describen las subredes empleadas por OSA para su funcionamiento:

Management Network La red de administración o también container network se encarga de proveer la administración y comunicación entre la infraestructura y los servicios de Openstack en containers o en servidores físicos. Para que todas las instancias tengan acceso a esta red, es necesario que todos los nodos host del datacenter cuenten con el bridge `br-mgmt`. A este último se le asocian las interfaces virtuales de cada contenedor y la lógica o física del host en cuestión, asignadas a dicha red. Esta interfaz suele ser la primaria del nodo mediante la cual se accede por SSH.

Overlay Network La red de superposición o también tunnel network, provee conectividad entre los hosts virtualizados dentro de Openstack encapsulando el tráfico con algún protocolo de tunelización como son VXLAN o GENEVE. La VLAN o interfaz utilizada para esta subred se asocia al bridge `br-vxlan`. Este bridge debe instanciarse en todos los nodos que manejen agentes del módulo Neutron, típicamente involucra los nodos de cómputo y/o red.

Storage Network La red de almacenamiento provee acceso entre los backends de almacenamiento, tales como Block storage, y los servicios de Openstack, como Cinder o Glance. En este caso las interfaces o VLANs se asocian al bridge `br-storage`, que debe instanciarse en todos los nodos que alojan servicios de cómputo o almacenamiento.

Interfaces de red

Como se mencionó en OSA se pueden tener diversas configuraciones dependiendo de la cantidad de interfaces del host físico, algunas de ellas se muestran a continuación.

Network Interface Layout - Multiple Interfaces

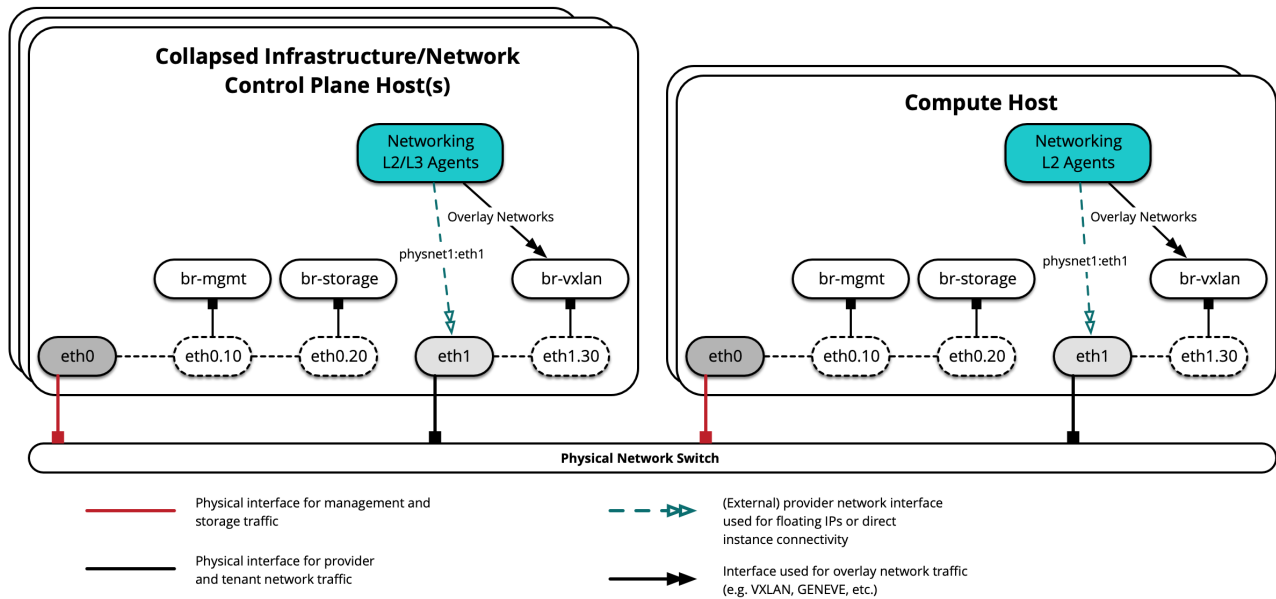


Figura 3.11: Diagrama de múltiples interfaces de red. Extraída de [42].

En la figura 3.11 se utilizan dos interfaces simples, subdivididas mediante la implementación de VLANs que finalmente se asocian a los bridges mencionados anteriormente.

Network Interface Layout - Multiple Bonded Interfaces

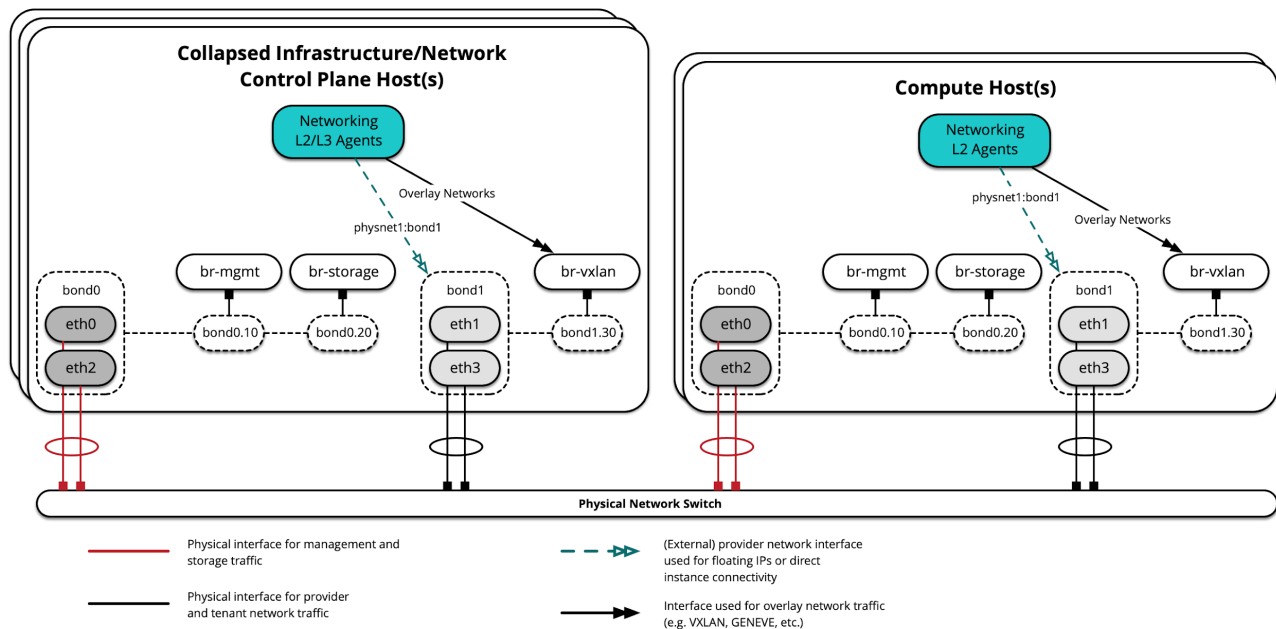


Figura 3.12: Diagrama de bonds de múltiples interfaces de red. Extraída de [42].

En el segundo, se cuenta con cuatro interfaces (provenientes de dos tarjetas físicas) que son bondeadas en forma cruzada para mejorar la disponibilidad y subdivididas mediante la implementación de VLANs para finalmente asociarlas a los bridges.

En ambos casos, se resalta la separación de las redes tenant con las redes internas para el funcionamiento de OSA y la ausencia de veth pairs asociadas a los bridges debido a que no se visualizan servicios desplegados en contenedores.

A continuación se presenta un diseño que ilustra servicios desplegados en contenedores y cómo varían las interconexiones de red para proveer conectividad.

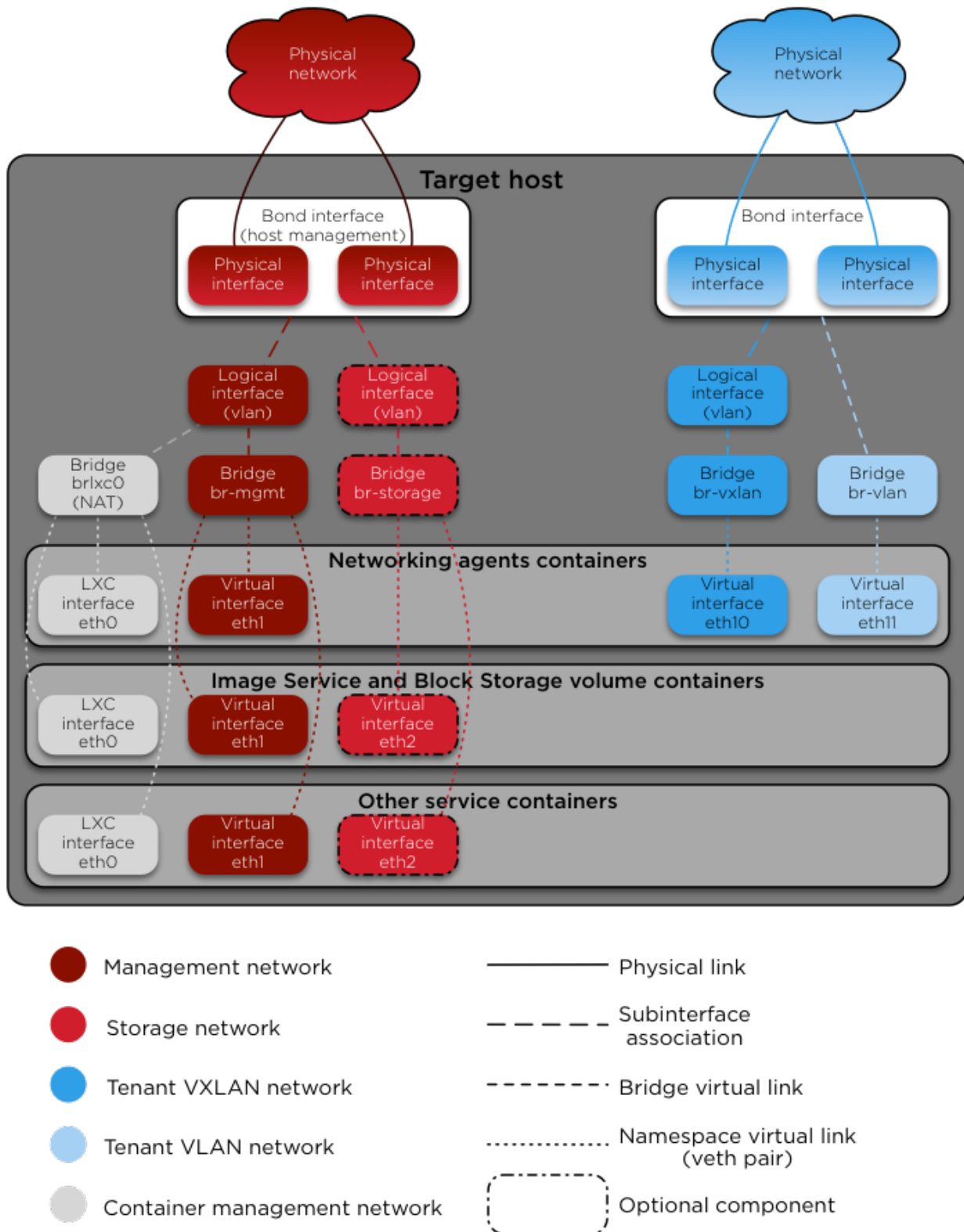


Figura 3.13: Despliegue de servicios Openstack en contenedores. Extraída de [17].

Como se menciona anteriormente los contenedores que corren servicios de Openstack requieren de interfaces virtuales para conectarse con los bridges del nodo físico. Además se puede observar cómo varían las conexiones necesarias a los distintos tipos de red, en función del tipo de agente que corre en el contenedor. Al utilizar contenedores, OSA crea automáticamente una nueva red que se utilizará para

la administración de los mismos (por ejemplo para descargar paquetes). Un punto a tener en cuenta es que Ansible para crear esta subred utiliza el rango 10.0.3.0/24.

A continuación se presenta el diagrama de una arquitectura modelo de Openstack Ansible en un ambiente de producción que involucra los conceptos mencionados en puntos anteriores.

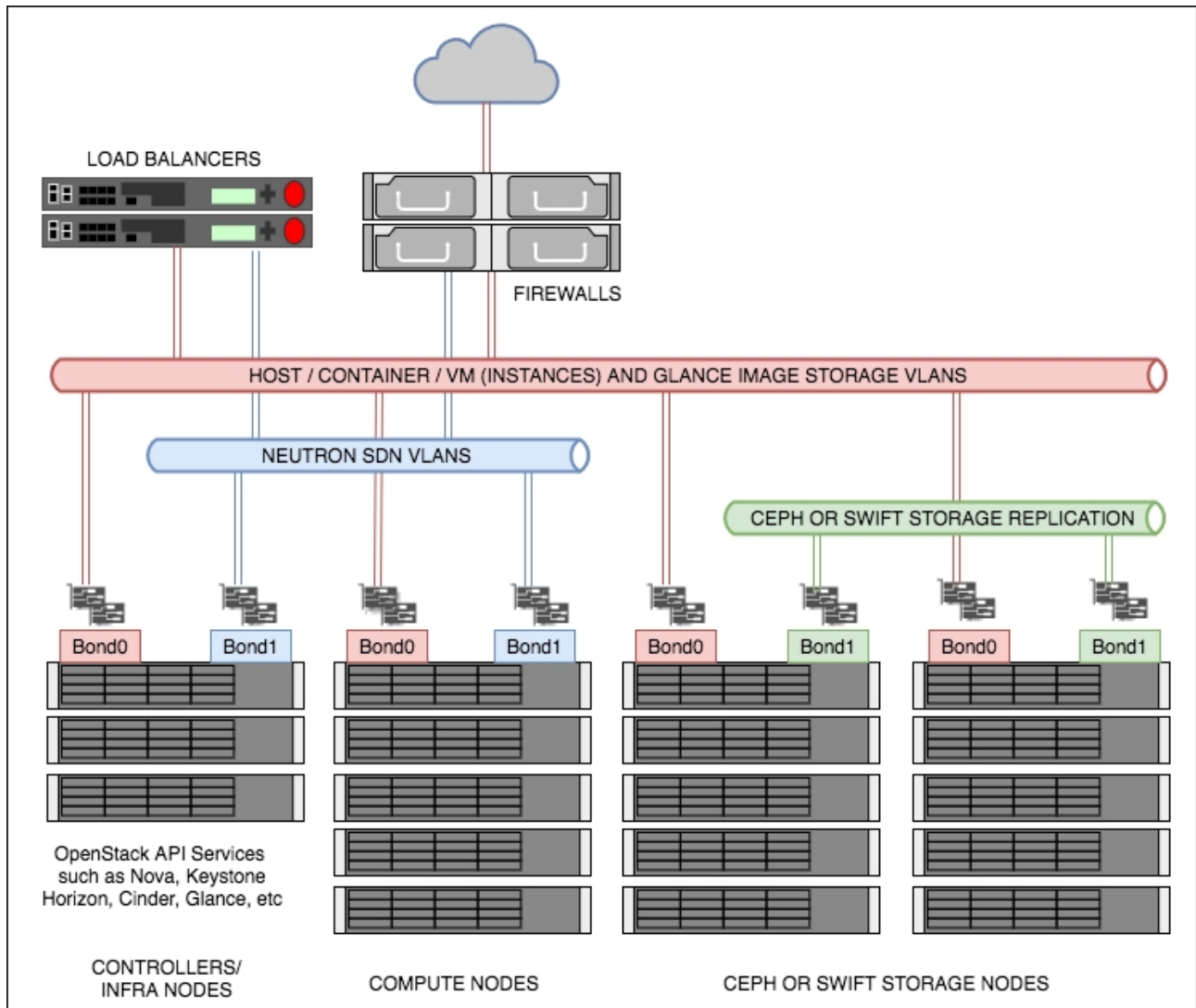


Figura 3.14: Extraída de [11]

3.7. Configuración OSA

En esta sección se presentan las configuraciones y conceptos más relevantes del nodo Deploy a tener en cuenta durante un despliegue de Openstack utilizando Ansible. Estas incluyen las convenciones de directorios empleadas, la configuración estándar y el significado del contenido de los archivos que deben ser modificados.

3.7.1. Convenciones

- El repositorio de OSA se clona generalmente en el directorio `/opt/openstack-ansible`.
- Los roles de Ansible utilizados por defecto se encuentran en el directorio `/etc/ansible/roles` los cuales son generados a partir del archivo `/opt/openstack-ansible/ansible-role-requirements`.

yml mediante la ejecución del `scriptbootstrap-ansible.sh`.

- Las configuraciones realizadas por el administrador son indicadas en el directorio `/etc/openstack_deploy`.

3.7.2. Inventario

Define las especificaciones de los hosts y contenedores dentro del ambiente actual de Openstack. Esta información se encuentra en el archivo `/etc/openstack-ansible/openstack_inventory.json`, generado a partir de los `host groups`, `containers groups` y `components` indicados en:

- La estructura por defecto almacenada en `/opt/openstack-ansible/inventory/env.d`
- Lo configurado por el administrador dentro de `/etc/openstack_deploy/` en:
 - El archivo `openstack_user_config.yml`
 - El directorio `conf.d/`
 - El directorio `env.d/`

Este archivo es considerado como referencia en cualquiera de los comandos asociados al despliegue de OSA por lo tanto nunca debe ser eliminado o modificado en un ambiente de producción.

Los `components` hacen referencia a los diferentes servicios que serán instalados durante el despliegue de Openstack, tanto en contenedores virtuales como directamente en los `target host`. Los `containers groups` agrupan estos `components`, determinando los potenciales contenedores a ser creados junto con sus especificaciones. En las configuraciones realizadas en ambos directorios `env.d/` se asocian los `containers groups` anteriores con los `hosts groups`, los cuales agrupan diversos `target hosts`. De esta forma se determina qué servicio debe ser instalado en qué `target host`.

3.7.3. `openstack_user_config.yml`

Es el principal archivo de configuración, creado por el operador de Openstack. Las especificaciones de cada sección se detallan en [48].

4

Instalación

4.1. Diseño de arquitectura

A continuación se presenta un diagrama 4.1 de la arquitectura diseñada para la instalación de Openstack Ansible. Al tratarse de un ambiente meramente de prueba, sólo se cuenta con un nodo de cada tipo definido previamente. En el mismo se puede apreciar que no se han utilizado VLANs ni bonds en las interfaces de los servidores, sino que se han agregado tantas interfaces físicas como fueran necesarias. Se detallan también las numeraciones IP a ser utilizadas en cada una de las redes necesarias y las asignaciones a los diferentes bridges de cada uno de los nodos. Cabe mencionar que la IP pública del balanceador de carga pertenece a la subred en la que se encuentra el servidor físico sobre el cual se virtualizarán todos los nodos, con el fin de tener acceso externo a la plataforma.

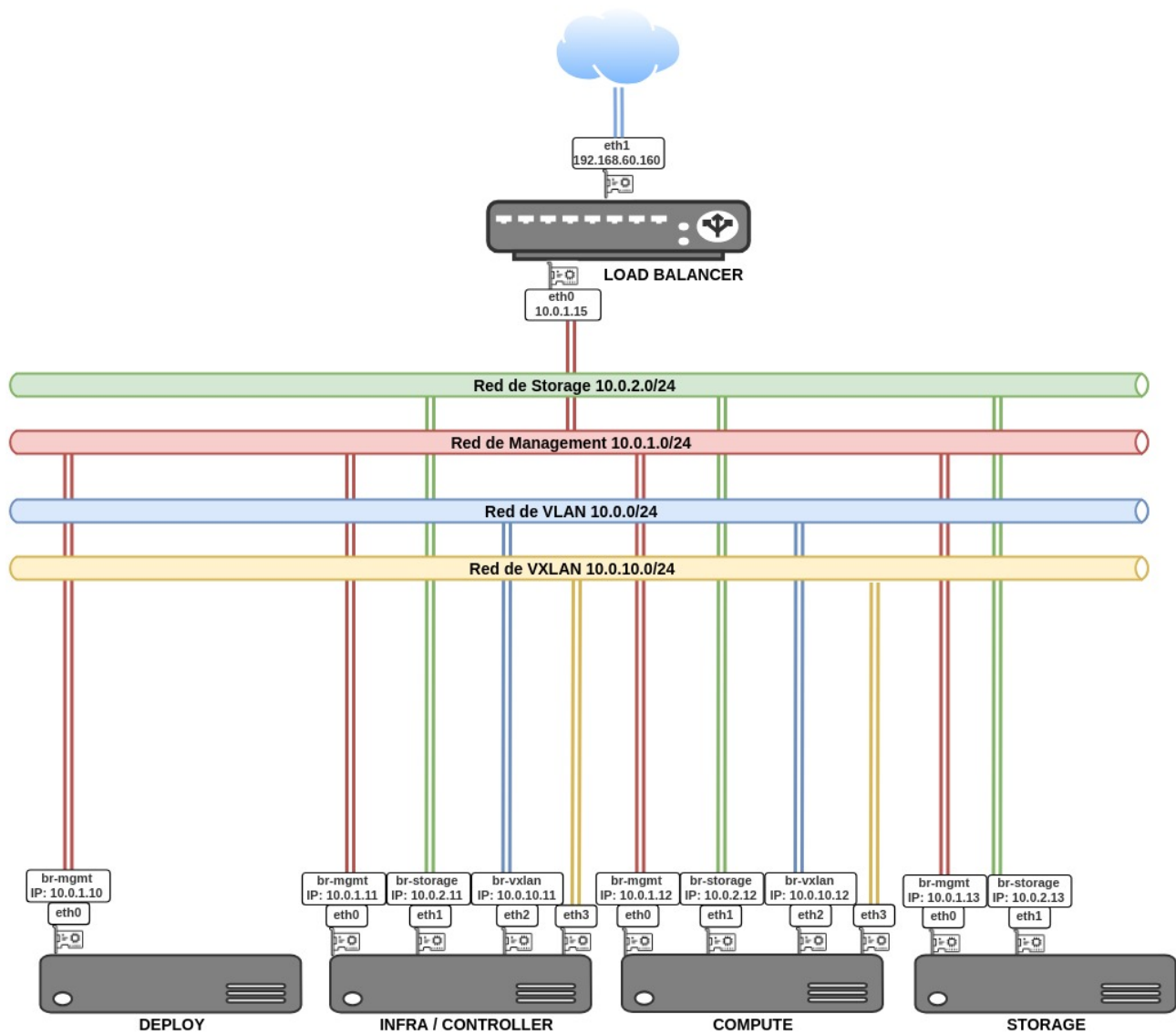


Figura 4.1: Arquitectura diseñada.

El siguiente diagrama muestra la disposición de los componentes de red en la arquitectura utilizada en donde el nodo de control y red se colapsaron a uno solo:

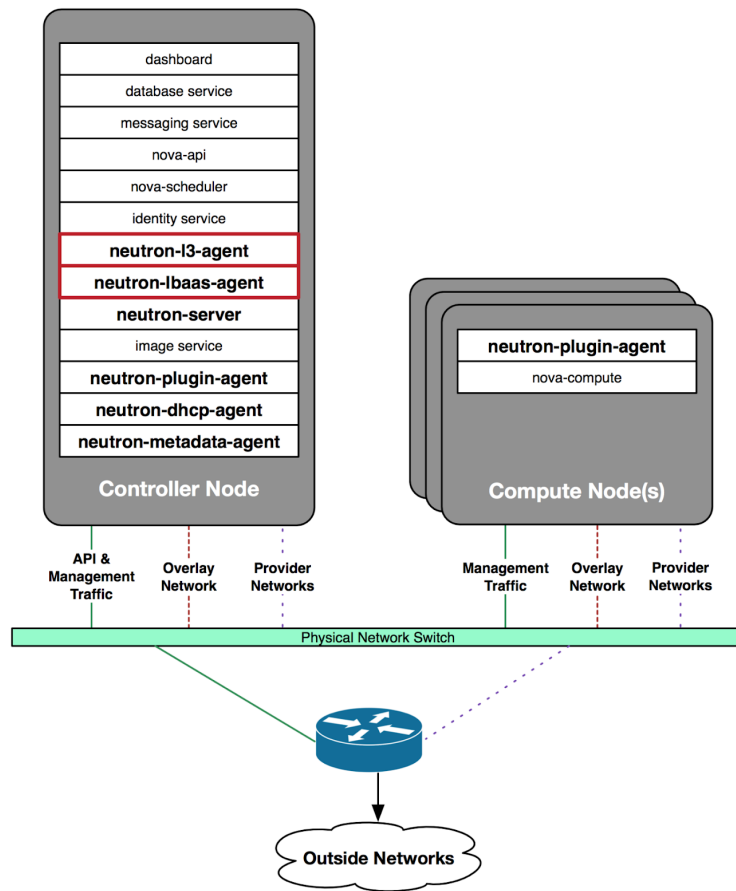


Figura 4.2: Disposición de componentes en Neutron. Extraída de [9].

4.2. Ambiente de trabajo

4.2.1. Hardware utilizado

Para realizar la instalación de Openstack se utilizó un servidor físico (denominado renata) alojado en el Instituto de Computación de la Facultad de Ingeniería (InCo). El mismo cuenta con una amplia cantidad de recursos destacando sus 40 procesadores virtuales, 128 GB de RAM y 40 TB de disco duro. Se aloja en una red privada del InCo en donde para salir a Internet se debe pasar por un proxy, provocando algunas limitaciones que luego se mencionan.

En el servidor fueron necesarias las siguientes configuraciones de red:

1. Creación de un bridge el cual será utilizado por la interfaz física del servidor y por los distintos NAT que se deben crear con KVM para la arquitectura que se virtualiza. Para esto se creó la interfaz br-mgmt con la siguiente configuración:

```
DEVICE="br-mgmt"
BOOTPROTO="none"
IPADDR="192.168.60.242"
PREFIX="24"
GATEWAY="192.168.60.1"
DNS1="192.168.60.230"
ONBOOT="yes"
TYPE="Bridge"
NM_CONTROLLED="no"
```

2. En la interfaz eno2 la cual tenía configurada la IP del bridge quedó de la siguiente forma:

```
BOOTPROTO=none
NAME=en02
UUID=824cd835-662a-4d47-a148-512aec3dd237
DEVICE=en02
ONBOOT=yes
BRIDGE="br-mgmt"
NM_CONTROLLED="no"
```

4.2.2. Conexión remota hacia el servidor renata

Dado que el servidor se encuentra en una red privada del InCo, para conectarse al mismo de forma remota se deben establecer algunas conexiones SSH. A continuación se detallan las conexiones y comandos utilizados.

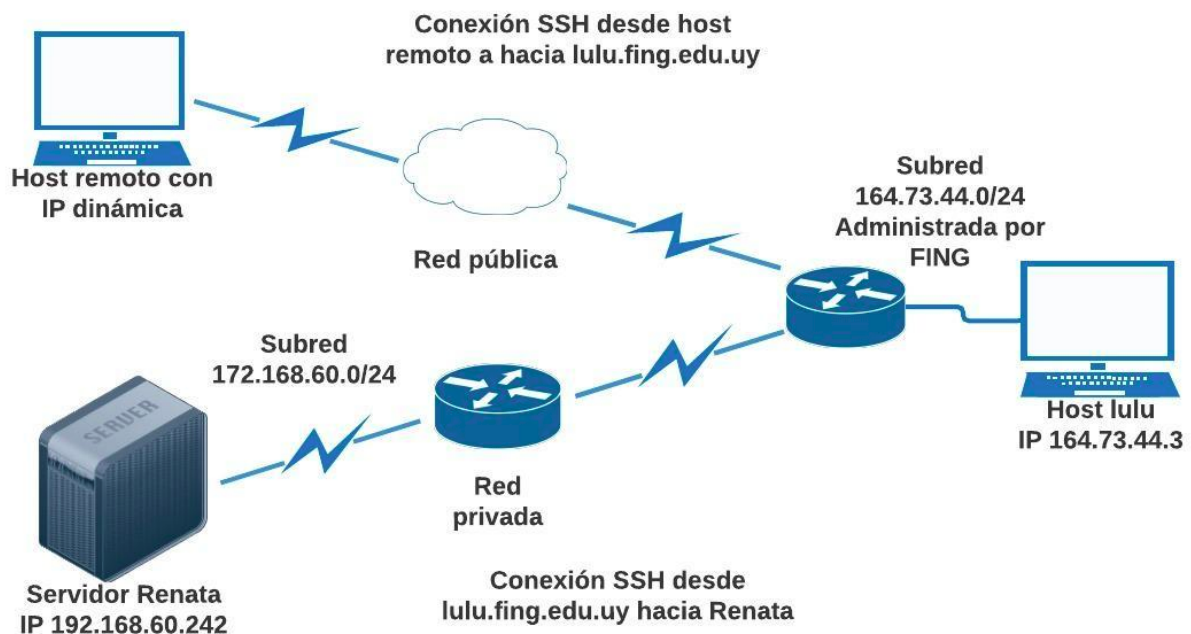


Figura 4.3: Acceso remoto al servidor renata.

1. Como el servidor se encuentra en una red privada del InCo solo se puede acceder desde un host que se encuentre en una red interna de la FIng, por ej: lulu.fing.edu.uy. Para esto ejecutar:

```
$ ssh usuario_fing@lulu.fing.edu.uy
```

2. Desde el host lulu para conectarse al servidor renata se debe ejecutar:

```
$ ssh openstack@192.168.60.242
```

4.2.3. Virtualización con KVM

Para crear la arquitectura de nodos se utilizó el virtualizador KVM, debido a que es la tecnología de virtualización utilizada normalmente en los servidores del InCo. Con el fin de facilitar la interacción con KVM a través de una interfaz gráfica, se utilizó el programa virt-manager.

Utilización virt-manager

Dentro del virt-manager lo primero a realizar es configurar una nueva conexión desde el menú Archivo -> Añadir conexión... de la siguiente forma:

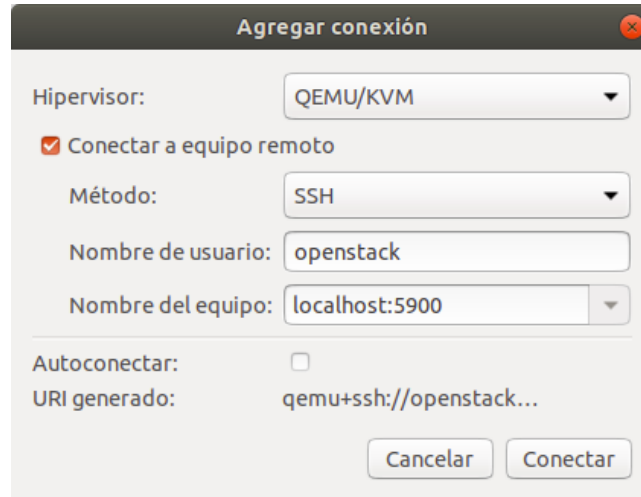


Figura 4.4: Nueva conexión en virt-manager.

Como un host remoto está a dos conexiones SSH del servidor Renata, la configuración que se muestra en la figura X no será suficiente. Para el correcto funcionamiento se debe crear una regla de forwarding que envíe todas las acciones realizadas por el virt-manager hacia el host `lulu.fing.edu.uy` el cual tiene acceso al servidor. Para lograr esto se debe ejecutar:

```
ssh -L 5900:192.168.60.242:22 <usuario_fing>@lulu.fing.edu.uy
```

El número de puerto utilizado puede ser cualquiera que no esté siendo utilizado y no sea privilegiado.

El orden adecuado para conectarse al servidor mediante virt-manager es:

1. Crear la conexión ssh indicada.
2. Iniciar virt-manager.
3. Inicializar la conexión. En este paso se puede llegar a requerir la contraseña del usuario del servidor renata desde la consola que esté ejecutando el manager.

Lo siguiente a realizar es crear con KVM las redes virtuales que se natean al bridge `br-mgmt` del servidor físico. Es necesario crear 4 redes, donde cada una se corresponde con las redes necesarias para el funcionamiento de Openstack. Estas redes son:

- NAT-Open (Red management Subred 10.0.1.0/24).
- NAT-Open-Storage (Red storage Subred 10.0.10.0/24).
- NAT-Open-Vxlan (Red vxlan Subred 10.0.2.0/24).
- NAT-Open-Vlan (Red vlan Subred 10.0.4.0/24).

Para crear estas redes se debe ir al menú Editar -> Detalles de la conexión. Luego como se muestra en la imagen en la pestaña de redes virtuales seleccionar el icono (+).

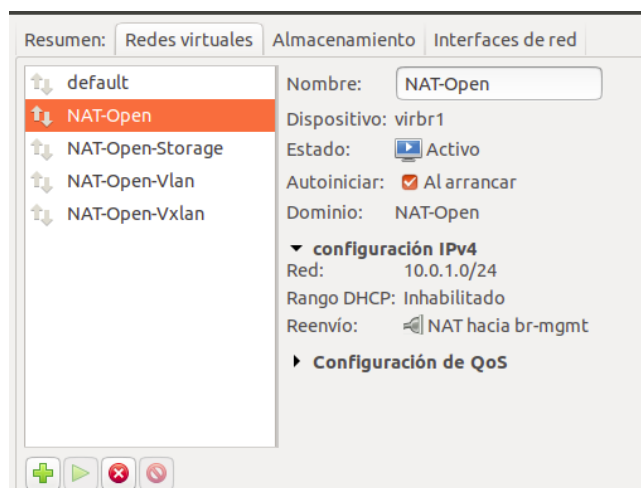
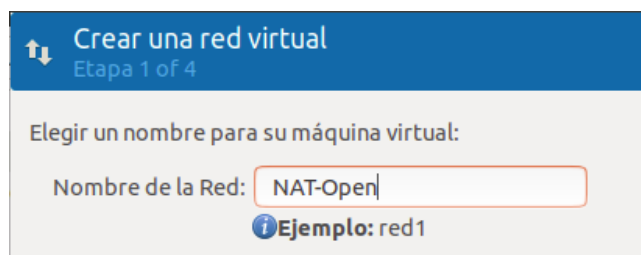


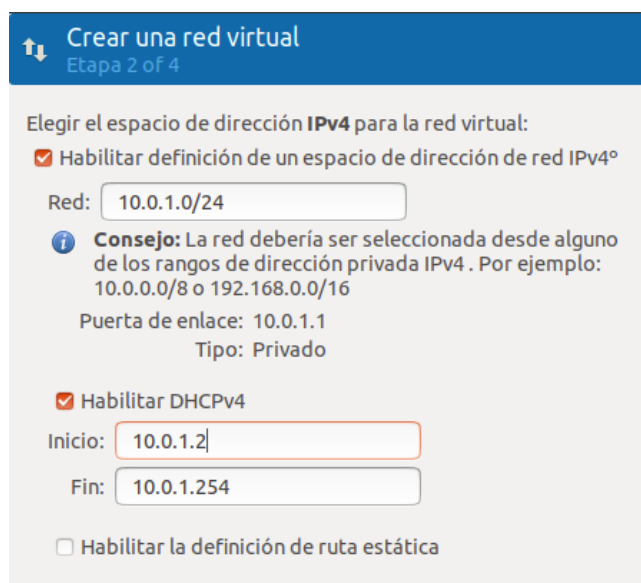
Figura 4.5: Configuración de redes virtuales en virt-manager.

Al presionar el botón para agregar una nueva red, desplegará en pantalla un wizard. A continuación se muestra el paso a paso de la creación de la red NAT-Open a modo de ejemplo:

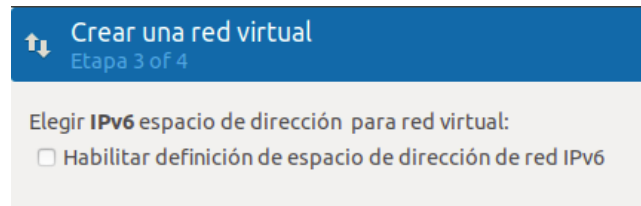
1. Seleccionar el nombre de la red



2. Seleccionar la subred y el rango para el DHCP



3. No habilitar direcciones IPv6

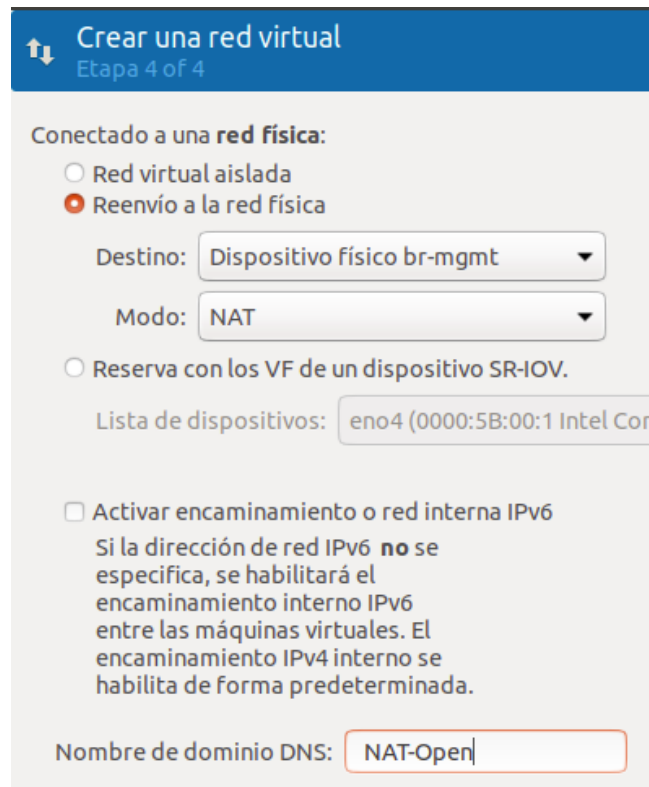


Crear una red virtual
Etapa 3 of 4

Elegir **IPv6** espacio de dirección para red virtual:

☐ Habilitar definición de espacio de dirección de red IPv6

4. Seleccionar el tipo de red virtual



Crear una red virtual
Etapa 4 of 4

Conectado a una **red física**:

☐ Red virtual aislada

☒ Reenvío a la red física

Destino:

Modo:

☐ Reserva con los VF de un dispositivo SR-IOV.

Lista de dispositivos:

☐ Activar encaminamiento o red interna IPv6

Si la dirección de red IPv6 **no** se especifica, se habilitará el encaminamiento interno IPv6 entre las máquinas virtuales. El encaminamiento IPv4 interno se habilita de forma predeterminada.

Nombre de dominio DNS:

El mismo procedimiento se deberá repetir con el resto de las redes listadas.

Lo último que es necesario crear con KVM son los nodos virtuales que se utilizaron para instalar Openstack. En la instalación realizada se utilizan 5 nodos con las especificaciones detalladas a continuación:

■ Nodo deploy:

- 1 interfaz en NAT-Open
- 2 CPUs
- 200 GB de disco
- 8 GB de RAM

■ Nodo haproxy1:

- 2 interfaces: una en NAT-Open y otra conectada al bridge br-mgmt de renata
- 4 CPUs
- 200 GB de disco
- 32 GB de RAM

■ Nodo infra1:

- 4 interfaces: una en cada NAT creada
- 8 CPUs
- 200 GB de disco
- 32 GB de RAM

■ Nodo compute1:

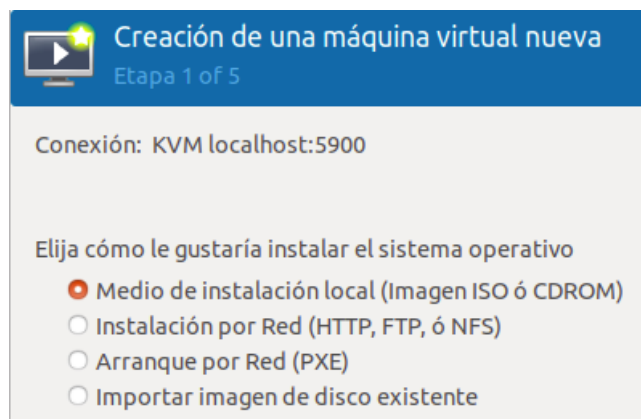
- 4 interfaces: una en cada NAT creada
- 8 CPUs
- 200 GB de disco
- 32 GB de RAM

■ Nodo storage1:

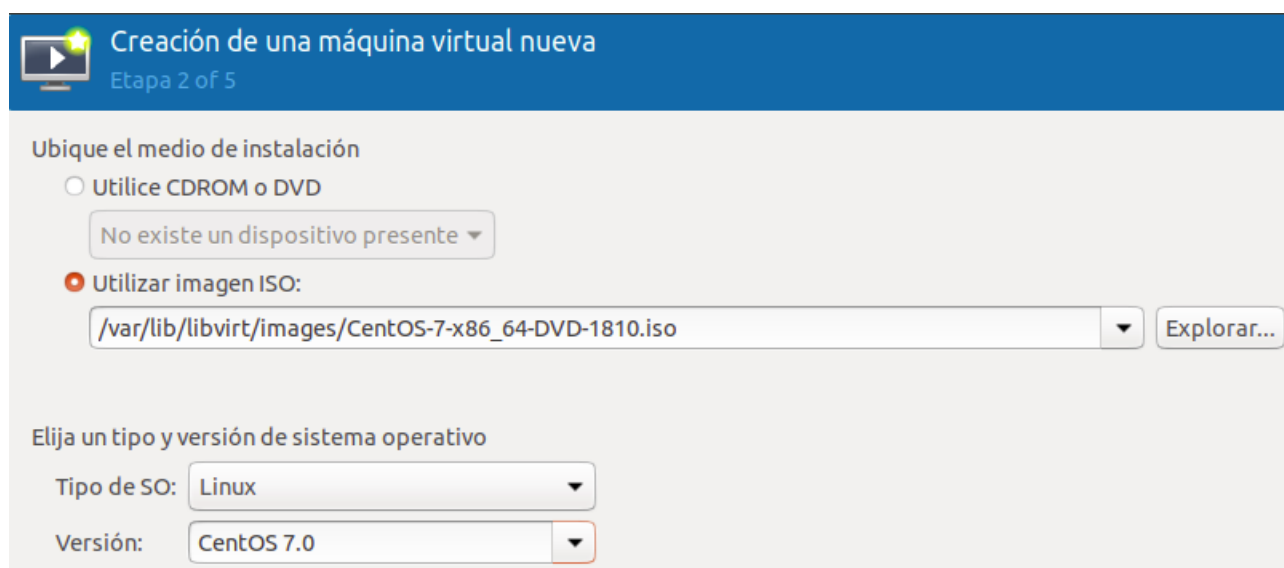
- 2 interfaces: una en NAT-Open y otra en NAT-Open-Storage
- 4 CPUs
- 2 discos: uno de 40 GB para el SO y otro de 200 GB para el volumen de cinder
- 32 GB de RAM

A continuación se detalla paso a paso cómo crear el nodo deploy, el resto se realiza de forma análoga salvando las diferencias de recursos e interfaces de red.

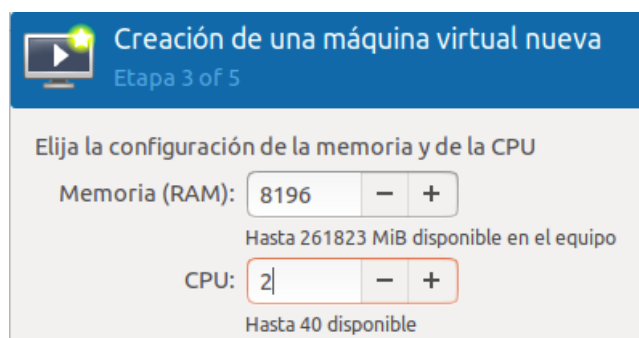
1. En el presente ejemplo se crea una VM suponiendo que se instalará el SO.



2. Seleccionar la imagen a utilizar y el tipo de SO. La misma deberá estar en un directorio del servidor físico.

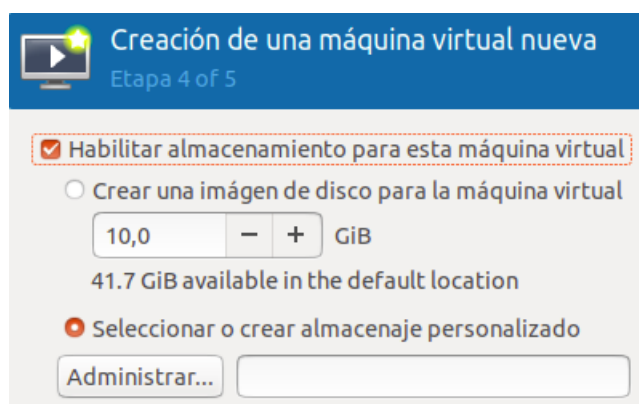


3. Seleccionar RAM y CPUs.

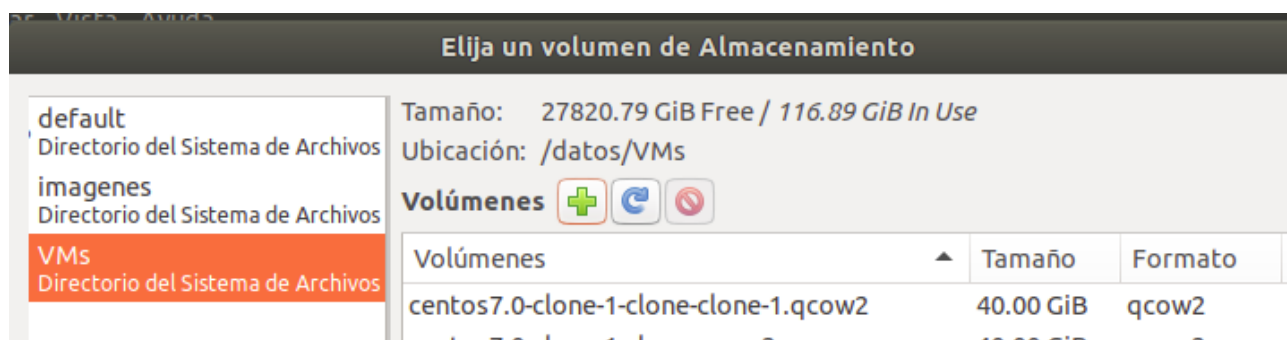


4. Para el almacenamiento se deberá crear un nuevo volumen de la siguiente forma:

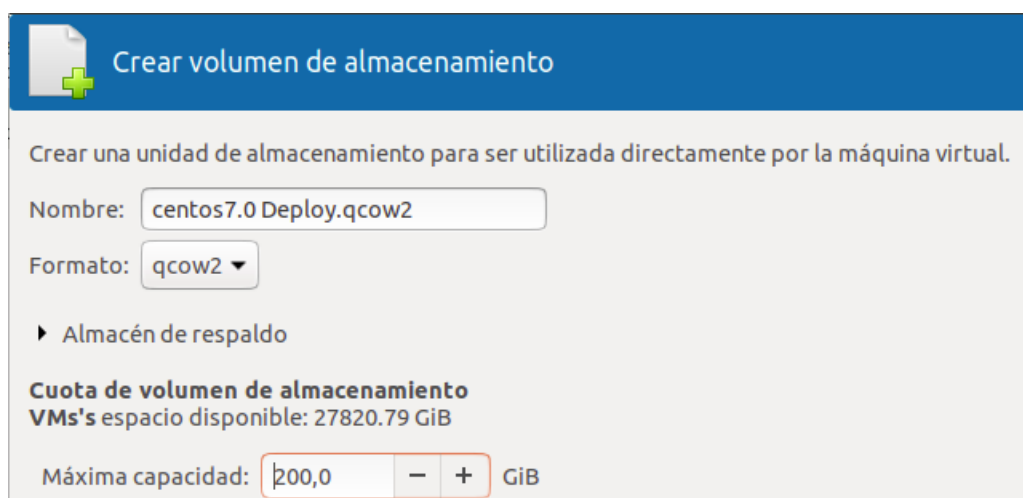
4.1. Elegir la segunda opción y presionar Administrar.



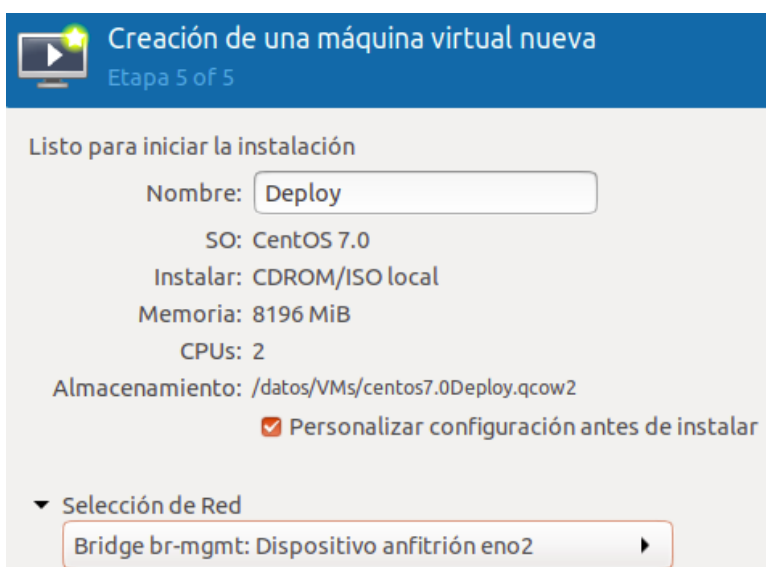
4.2. Esto desplegará una ventana mostrando directorios del sistema de archivos de renata. Se puede elegir un volumen existente o crear uno nuevo con el botón (+).



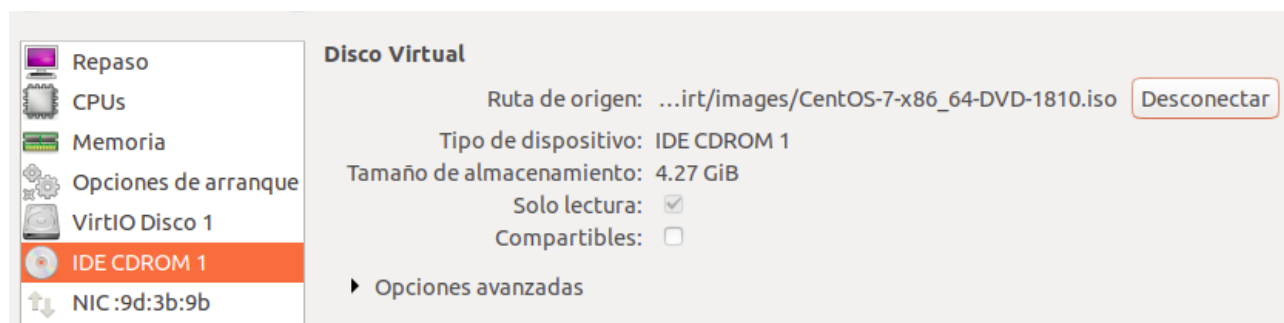
4.3. Al crearlo se debe especificar el nombre, el tipo y la capacidad.



5. Luego se debe ingresar el nombre de la máquina y seleccionar la red que conecta el host directamente al bridge del servidor físico en lugar de la red NAT-Open para que el host tenga conexión a internet durante la instalación y poder realizar las primeras configuraciones en el mismo.



6. Se debe verificar en la opción IDE CDROM 1 que esté correctamente seteado la imagen a utilizar para instalar el SO.



7. Verificar en las opciones de arranque que el IDE CDROM 1 esté en primer lugar para que la máquina bootee con la imagen seleccionada.



8. Finalmente al confirmar todos los cambios se lanzará la VM creada y se instalará el SO.

4.2.4. Especificaciones servidor renata

La red en la que se encuentra el servidor Renata no cuenta con acceso a internet. Para solucionar este problema se realizó un túnel ssh reverso desde el host lulu.fing.edu.uy al servidor Renata para establecer como proxy de Renata el host proxy.fing.edu.uy, siendo este último el proxy de la FING.

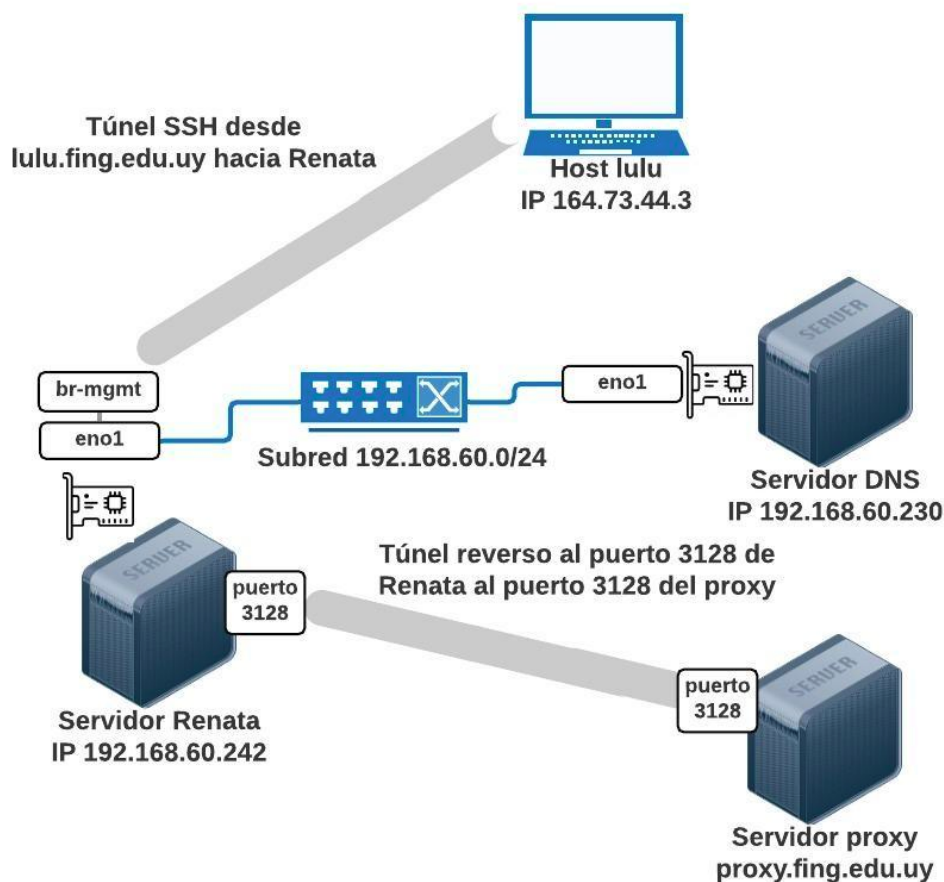


Figura 4.6: Túnel reverso y esquema de servidores.

El funcionamiento de esta configuración se detalla a continuación:

- El host lulu.fing.edu.uy inicia una conexión SSH con renata en donde indica que cree una conexión inversa con el proxy de la FIng.
- Al establecer la conexión, el servidor renata crea el túnel inverso en donde envía todos los paquetes destinados al puerto 3128 al servidor proxy en el puerto 3128.
- Finalmente para que renata tenga acceso a Internet se setean las variables de entorno `http_proxy` y `https_proxy` con el valor `http://localhost:3128`. De esta forma todo el tráfico http será enviado al puerto 3128 local y gracias al túnel reverso todos los pedidos http serán enviados al proxy logrando tener acceso a Internet.

Para lograr esto se debe ejecutar el siguiente comando al iniciar la conexión SSH desde lulu hacia renata:

```
$ ssh -R 3128:proxy.fing.edu.uy:3128 openstack@192.168.60.242
```

Como se muestra en la figura 4.6 el servidor DNS utilizado se encuentra en la misma red local.

4.2.5. Acceso al exterior desde nodos

Los nodos virtualizados en el servidor renata con la configuración por defecto no tendrán acceso a Internet. Esto sucede por que al realizar un pedido http o https será enviado al gateway de estos nodos virtuales que es la IP 10.0.1.1 de la interfaz virtual de renata, en donde solamente forwardea los paquetes recibidos en el puerto 3128 al proxy de la FIng. Para solucionar esto basta con configurar

las variables de entorno `http_proxy` y `https_proxy` en los nodos virtualizados con el siguiente valor: `http://10.0.1.1:3128`. Esto aplica para los 5 nodos utilizados en la instalación.

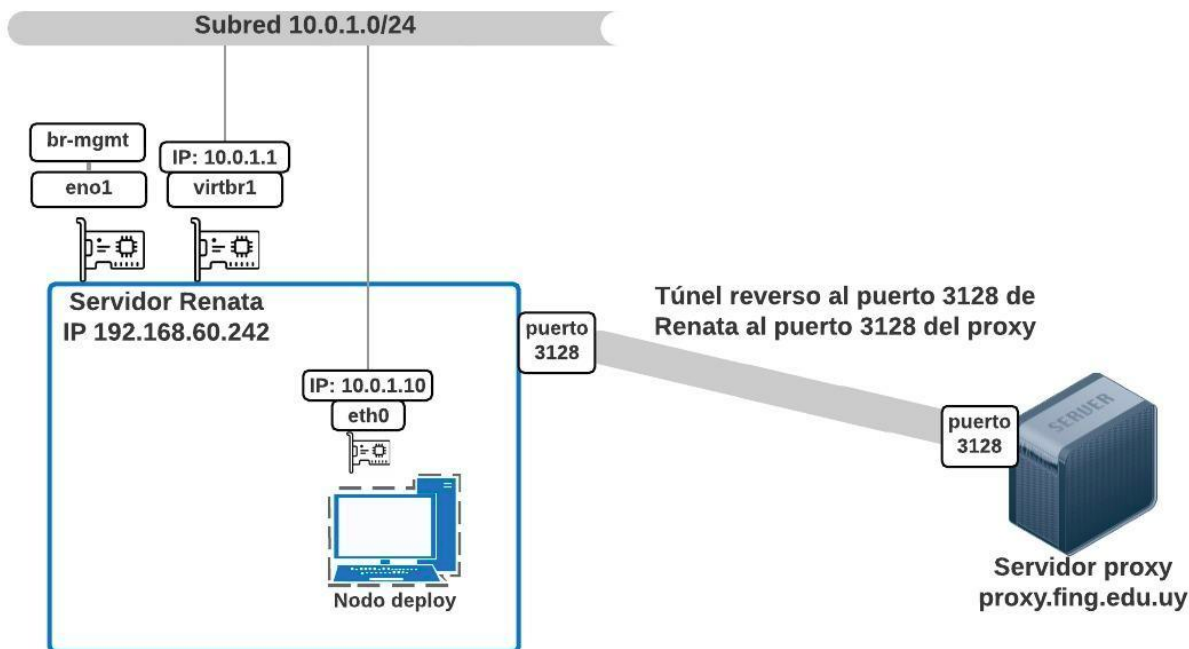


Figura 4.7: Salida a Internet en los nodos de Openstack.

4.3. Preparación de nodos

En la siguiente sección se detallan los pasos necesarios a seguir en cada uno de los nodos para iniciar con la instalación de Openstack. Para realizar dicha configuración inicial será necesario que los nodos cuenten con conexión a internet. En el ambiente de trabajo actual, esto es equivalente a verificar que las variables del proxy estén bien configuradas.

Deploy

1. Configurar la interfaz de red `eth0` de la siguiente forma:

```
TYPE=Ethernet
BOOTPROTO=none
DEFROUTE=yes
NAME=eth0
DEVICE=eth0
ONBOOT=yes
IPADDR=10.0.1.10
PREFIX=24
GATEWAY=10.0.1.1
DNS1=192.168.60.230
```

2. Asociar, en el `virt-manager`, la interfaz de red `eth0` a la red virtual de management `10.0.1.0/24`.

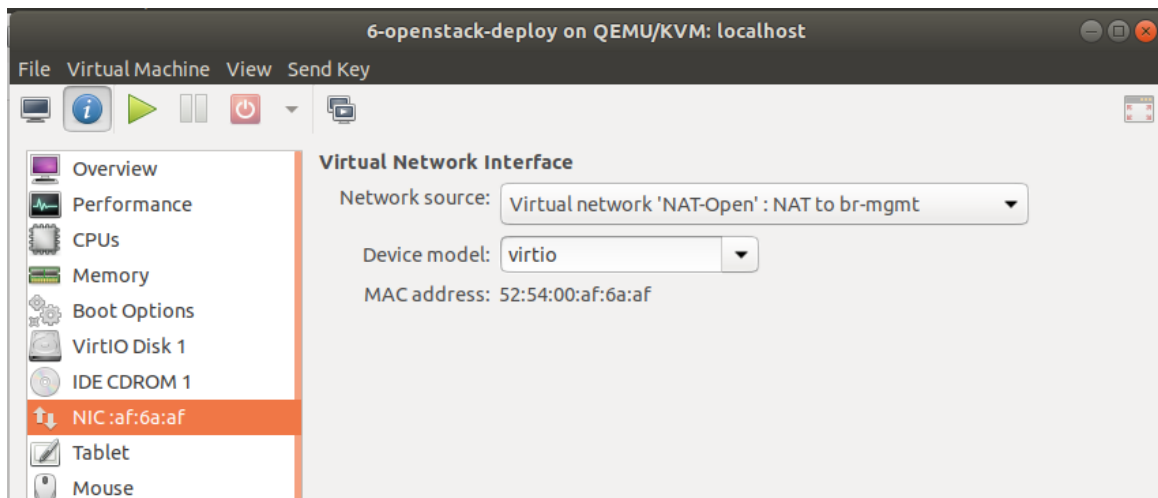


Figura 4.8

3. Por simplificación , cambiar el hostname de la máquina a ‘deploy’ con el comando:

```
$ hostname deploy
```

4. Configurar las variables de proxy en `/etc/environment` con los siguientes comandos:

```
$ echo "http_proxy=http://10.0.1.1:3128" >> /etc/environment
$ echo "https_proxy=http://10.0.1.1:3128" >> /etc/environment
$ echo "HTTP_PROXY=http://10.0.1.1:3128" >> /etc/environment
$ echo "HTTPS_PROXY=http://10.0.1.1:3128" >> /etc/environment
$ source /etc/environment
```

Verificar el acceso a internet mediante el comando:

```
$ curl www.google.com.
```

5. Instalaciones necesarias:

- 5.1. Actualizar repositorios y reiniciar:

```
$ yum upgrade -y
$ reboot
```

- 5.2. Instalar repositorio openstack queens:

```
$ yum install -y https://rdoproject.org/repos/openstack-queens/rdo-release-queens.rpm
```

- 5.3. Instalar herramientas auxiliares:

```
$ yum install -y git ntp nano net-tools ntpdate openssh-server python-devel sudo '
@Development Tools'
```

6. Deshabilitar firewall de CentOS:

```
$ systemctl stop firewalld
$ systemctl mask firewalld
```

7. Configurar el servicio de NTP mediante Chrony, sustituyendo en el archivo `/etc/chrony.conf` las siguientes líneas:

```
server 0.south-america.pool.ntp.org
server 1.south-america.pool.ntp.org
server 2.south-america.pool.ntp.org
server 3.south-america.pool.ntp.org
```

Reiniciar el servicio mediante:

```
$ systemctl restart chronyd
```

8. Clonar el repositorio de Openstack Ansible Queens

```
$ git clone -b 17.1.10 https://git.openstack.org/openstack/openstack-ansible /opt/
  openstack-ansible
```

9. Preparar openstack-ansible:

```
$ /opt/openstack-ansible/scripts/bootstrap-ansible.sh
```

Este script se utiliza para asegurar que Ansible y todas las dependencias necesarias para la instalación de OSA están instaladas en el nodo de deploy.

10. Copiar el contenido de configuración al /etc:

```
$ cp -r /opt/openstack-ansible/etc/openstack_deploy /etc/
```

11. Crear la carpeta para logs de la instalación:

```
$ mkdir /var/log/openstack
```

Infra1

1. Configuraciones de red:

1.1. Configurar los bridges br-mgmt, br-storage y br-vxlan.

DEVICE=br-mgmt	DEVICE=br-storage	DEVICE=br-vxlan
BOOTPROTO=none	BOOTPROTO=none	BOOTPROTO=none
IPADDR=10.0.1.11	IPADDR=10.0.2.11	IPADDR=10.0.10.11
PREFIX=24	PREFIX=24	PREFIX=24
GATEWAY=10.0.1.1	ONBOOT=yes	ONBOOT=yes
DNS1=192.168.60.230	TYPE=Bridge	TYPE=Bridge
ONBOOT=yes	NM_CONTROLLED=no	NM_CONTROLLED=no
TYPE=Bridge		
NM_CONTROLLED=no		

1.2. Configurar las interfaces eth0, eth1, eth2 y eth3.

TYPE="Ethernet"	TYPE="Ethernet"	TYPE="Ethernet"	TYPE="Ethernet"
BOOTPROTO=none	BOOTPROTO=none	BOOTPROTO=none	BOOTPROTO=none
DEVICE=eth0	DEVICE=eth1	DEVICE=eth2	DEVICE=eth3
ONBOOT=yes	ONBOOT=yes	ONBOOT=yes	ONBOOT=yes
NM_CONTROLLED=no	NM_CONTROLLED=no	NM_CONTROLLED=no	NM_CONTROLLED=no
BRIDGE=br-mgmt	BRIDGE=br-storage	BRIDGE=br-vxlan	

2. Asociar, en el virt-manager, las interfaces de red:

2.1. eth0 a la red virtual de management 10.0.1.0/24.

2.2. eth1 a la red virtual de storage 10.0.2.0/24.

2.3. eth2 a la red virtual de vxlan 10.0.10.0./24.

2.4. eth3 a la red virtual de vlan 10.0.4.0./24.

3. Cambiar el hostname de la máquina a 'infra1' con el comando:

```
$ hostname infra1
```

4. Configurar las variables de proxy en la sesión pero no en el sistema, porque luego el nodo deploy configura el `/etc/environment` en todos los nodos.

```
$ export http_proxy=http://10.0.1.1:3128
$ export https_proxy=http://10.0.1.1:3128
$ export HTTP_PROXY=http://10.0.1.1:3128
$ export HTTPS_PROXY=http://10.0.1.1:3128
```

Verificar el acceso a internet mediante el comando:

```
$ curl www.google.com
```

5. Actualizar repositorios y reiniciar:

```
$ yum upgrade -y
$ reboot
```

6. Instalar herramientas auxiliares:

```
$ yum install bridge-utils iputils lsof lvm2 ntp ntpdate openssh-server sudo tcpdump
python net-tools nano
```

7. Deshabilitar el Network Manager:

```
$ chkconfig NetworkManager off
$ chkconfig network on
$ service NetworkManager stop
$ service network start
```

8. Deshabilitar el SELinux, cambiando en `/etc/sysconfig/selinux`, "SELINUX=enforcing" por "SELINUX=disabled".

9. Habilitar bonding de interfaces y vlans:

```
$ echo 'bonding' >> /etc/modules-load.d/openstack-ansible.conf
$ echo '8021q' >> /etc/modules-load.d/openstack-ansible.conf
```

10. Configurar el servicio de NTP mediante Chrony, sustituyendo en el archivo `/etc/chrony.conf` las siguientes líneas:

```
server 0.south-america.pool.ntp.org
server 1.south-america.pool.ntp.org
server 2.south-america.pool.ntp.org
server 3.south-america.pool.ntp.org
```

Reiniciar el servicio mediante:

```
$ systemctl restart chronyd
```

11. Eliminar reglas de firewall que bloquean el tráfico:

- 11.1. Eliminar las reglas manualmente:

```
$ iptables -D INPUT -j REJECT --reject-with icmp-host-prohibited
$ iptables -D FORWARD -j REJECT --reject-with icmp-host-prohibited
```

11.2. Exportar las reglas a un archivo:

```
$ iptables-save > /etc/sysconfig/iptables
```

11.3. En cada reboot de la máquina, importar las reglas del archivo anterior:

```
$ iptables-restore < /etc/sysconfig/iptables
```

Compute1

1. Configuraciones de red:

1.1. Configurar los bridges br-mgmt, br-storage y br-vxlan.

DEVICE=br-mgmt	DEVICE=br-storage	DEVICE=br-vxlan
BOOTPROTO=none	BOOTPROTO=none	BOOTPROTO=none
IPADDR=10.0.1.12	IPADDR=10.0.2.12	IPADDR=10.0.10.12
PREFIX=24	PREFIX=24	PREFIX=24
GATEWAY=10.0.1.1	ONBOOT=yes	ONBOOT=yes
DNS1=192.168.60.230	TYPE=Bridge	TYPE=Bridge
ONBOOT=yes	NM_CONTROLLED=no	NM_CONTROLLED=no
TYPE=Bridge		
NM_CONTROLLED=no		

1.2. Configurar las interfaces eth0, eth1, eth2 y eth3.

TYPE="Ethernet"	TYPE="Ethernet"	TYPE="Ethernet"	TYPE="Ethernet"
BOOTPROTO=none	BOOTPROTO=none	BOOTPROTO=none	BOOTPROTO=none
DEVICE=eth0	DEVICE=eth1	DEVICE=eth2	DEVICE=eth3
ONBOOT=yes	ONBOOT=yes	ONBOOT=yes	ONBOOT=yes
NM_CONTROLLED=no	NM_CONTROLLED=no	NM_CONTROLLED=no	NM_CONTROLLED=no
BRIDGE=br-mgmt	BRIDGE=br-storage	BRIDGE=br-vxlan	

2. Asociar, en el virt-manager, las interfaces de red:

2.1. eth0 a la red virtual de management 10.0.1.0/24.

2.2. eth1 a la red virtual de storage 10.0.2.0/24.

2.3. eth2 a la red virtual de vxlan 10.0.10.0/24.

2.4. eth3 a la red virtual de vlan 10.0.4.0/24.

3. Cambiar el hostname de la máquina a 'compute1' con el comando:

```
$ hostname compute1
```

A partir de este punto, el procedimiento es idéntico al nodo infra1. Se deben realizar los puntos 4 al 11.

Storage1

1. Configuraciones de red:

1.1. Configurar los bridges br-mgmt y br-storage.


```

DEVICE=br-mgmt
BOOTPROTO=none
IPADDR=10.0.1.13
PREFIX=24
GATEWAY=10.0.1.1
DNS1=192.168.60.230
ONBOOT=yes
TYPE=Bridge
NM_CONTROLLED=no

```

```

DEVICE=br-storage
BOOTPROTO=none
IPADDR=10.0.2.13
PREFIX=24
ONBOOT=yes
TYPE=Bridge
NM_CONTROLLED=no

```

1.2. Configurar las interfaces eth0 y eth1.

```

TYPE="Ethernet"
BOOTPROTO=none
DEVICE=eth0
ONBOOT=yes
NM_CONTROLLED=no
BRIDGE=br-mgmt

```

```

TYPE="Ethernet"
BOOTPROTO=none
DEVICE=eth1
ONBOOT=yes
NM_CONTROLLED=no
BRIDGE=br-storage

```

2. Asociar, en el virt-manager, las interfaces de red:

2.1. eth0 a la red virtual de management 10.0.1.0/24.

2.2. eth1 a la red virtual de storage 10.0.2.0./24.

3. Cambiar el hostname de la máquina a 'storage1' con el comando:

```
$ hostname storage1
```

A partir de este punto, el procedimiento es idéntico al nodo infra1. Se deben realizar los puntos 4 al 11.

12. Crear el volumen de LVM para utilizar Cinder:

12.1. Listar los devices en la máquina:

```
$ lvm diskscan
```

12.2. Formatear la pieza física de almacenamiento de 200 GB:

```
$ pvcreate --metadatasize 2048 physical_volume_device_path
```

12.3. Crear el nuevo grupo de almacenamiento que será utilizado por Openstack:

```
$ vgcreate cinder-volumes physical_volume_device_path
```

12.4. Verificar que el grupo quedó creado correctamente:

```
$ vgdisplay
```

HAproxy1

1. Configurar las interfaces eth0 y eth1.

```

TYPE=Ethernet
BOOTPROTO=none
DEFROUTE=yes
DEVICE=eth0

```

```

ONBOOT=yes
IPADDR=10.0.1.15
PREFIX=24
GATEWAY=10.0.1.1

```

```
DNS1=192.168.60.230
```

```
TYPE=Ethernet
BOOTPROTO=none
DEFROUTE=yes
DEVICE=eth1
ONBOOT=yes
IPADDR=192.168.60.160
PREFIX=24
```

2. Asociar, en el virt-manager, las interfaces de red:

2.1. eth0 a la red virtual de management 10.0.1.0/24.

2.2. eth1 al bridge alojado en el servidor renata en la red 192.168.60.0/24.

3. Cambiar el hostname de la máquina a 'haproxy1' con el comando:

```
$ hostname haproxy1
```

A partir de este punto, el procedimiento es similar al nodo infra1. Se deben realizar los puntos 4 al 11 con la excepción del punto 7 en donde se deshabilita el NetworkManager.

4.4. Configuración

Luego de completar la preparación de los nodos y verificar la conectividad entre los mismos, los últimos pasos antes de iniciar con la instalación de Openstack son configurar los archivos que OSA utiliza y los requerimientos extras de la herramienta utilizada para la instalación.

4.4.1. Configuración claves SSH

Como se menciona en la sección de Ansible, el nodo de deploy requiere de una conexión SSH con cada uno de los servidores que componen el datacenter, para poder configurar y operar directamente sobre cada uno de ellos. Para esto se utilizan un par de claves SSH público-privada con el fin de brindarle al nodo de deploy mayor flexibilidad al momento de acceder a los servidores. Se deberá propagar la clave del usuario root dado que es este usuario el que llevará a cabo la instalación. El comando que se debe ejecutar para crear las claves es el siguiente:

```
$ ssh-keygen -t rsa -f ~/.ssh/id_rsa -N ""
```

Donde:

- **-t:** especifica el tipo de encriptación.
- **-f:** determina el archivo en donde quedará la clave privada (por defecto, la pública será igual con la extensión .pub).
- **-N ""** indica que no se utilizará ninguna passphrase en la clave.

Esto genera dos archivos (`id_rsa` y `id_rsa.pub`) donde el primero es la clave privada (que no se deberá compartir) y el segundo la clave pública que se copiará al resto de los servidores con los siguientes comandos:

```
$ ssh-copy-id root@10.0.1.11
$ ssh-copy-id root@10.0.1.12
$ ssh-copy-id root@10.0.1.13
$ ssh-copy-id root@10.0.1.15
```

Para verificar la configuración bastará con realizar un ssh a cada servidor desde el deploy con el usuario root, accediendo en forma directa al prompt de dicho servidor.

4.4.2. Archivos de configuración OSA

openstack_user_config.yml

A continuación se detalla cada bloque de configuración utilizado en la instalación realizada.

En la primera sección, `cidr_networks`, se describen las subredes utilizadas en la instalación de Openstack.

```
cidr_networks:
  container: 10.0.1.0/24
  tunnel: 10.0.10.0/24
  storage: 10.0.2.0/24
```

La red de container permite a los distintos servicios , ejecutados dentro de contenedores o servidores físicos, comunicarse entre sí. La red tunnel es utilizada cuando un usuario crea una nueva red tennant dentro de un proyecto de Openstack, en el caso que corresponda se creará una red VXLAN en este red física. Por último la red de storage la utilizaran los nodos que alojan el backend y los servicios de storage de Openstack. Los rangos elegidos son totalmente arbitrarios, a excepción de la subred 10.0.3.0/24 debido a que es utilizada internamente por OSA durante su ejecución como se mencionó en la sección de arquitectura de red.

Lo siguiente a configurar son las direcciones IP que están siendo utilizadas por los hosts físicos de la infraestructura en las redes definidas previamente, con el fin de reservarlas y que la instalación no utilice ninguna de ellas para la estructura de Openstack.

```
used_ips:
  - "10.0.1.1,10.0.1.20" # red de management
  - "10.0.2.1,10.0.2.20" # red de storage
  - "10.0.10.1,10.0.10.20" # red de vxlan
```

En la sección `global_overrides` se describen los bridges utilizados y detalles de las interfaces del ambiente, por ejemplo, cómo los containers se conectan a la red física de los hosts.

```
global_overrides:
  internal_lb_vip_address: 10.0.1.15
  external_lb_vip_address: 192.168.60.160
```

Estos parámetros refieren a las direcciones IPs pública y privada del balanceador de carga. La privada es utilizada internamente por los servicios de Openstack y la pública es la puerta de acceso para los usuarios.

```
tunnel_bridge: "br-vxlan"
management_bridge: "br-mgmt"
storage_bridge: "br-storage"
```

Con estas variables se define el nombre asignado a cada uno de los bridges creados en los hosts físicos.

La sección de `provider_networks` describe la relación entre la red de los contenedores y el ambiente físico de los servidores.

```
provider_networks:
  - network:
      group_binds:
        - all_containers
        - hosts
      type: "raw"
      container_bridge: "br-mgmt"
      container_interface: "eth1"
```

```

        container_type: "veth"
        ip_from_q: "container"
        is_container_address: true
        is_ssh_address: true
- network:
    group_binds:
        - glance_api
        - cinder_api
        - cinder_volume
        - nova_compute
    type: "raw"
    container_bridge: "br-storage"
    container_type: "veth"
    container_interface: "eth2"
    container_mtu: "9000"
    ip_from_q: "storage"
- network:
    group_binds:
        - neutron_linuxbridge_agent
    container_bridge: "br-vxlan"
    container_type: "veth"
    container_interface: "eth10"
    container_mtu: "9000"
    ip_from_q: "tunnel"
    type: "vxlan"
    range: "1:1000"
    net_name: "vxlan"
- network:
    group_binds:
        - neutron_linuxbridge_agent
    container_bridge: "br-vlan"
    container_type: "veth"
    container_interface: "eth12"
    host_bind_override: "eth3"
    type: "flat"
    net_name: "flat"

```

La última sección describe en qué servidor o grupo de servidores corre cada servicio de Openstack y de infraestructura.

```

### Infrastructure
# galera, memcache, rabbitmq, utility
shared-infra_hosts:
    infra1:
        ip: 10.0.1.11
# repository (apt cache, python packages, etc)
repo-infra_hosts:
    infra1:
        ip: 10.0.1.11
# load balancer
# Dedicated hardware is best for improved performance and security.
haproxy_hosts:
    balancer1:
        ip: 10.0.1.15
# rsyslog server
log_hosts:
    infra1:
        ip: 10.0.1.11

```

```

### OpenStack
# keystone
identity_hosts:
    infra1:
        ip: 10.0.1.11
# cinder api services
storage-infra_hosts:
    infra1:
        ip: 10.0.1.11
# glance
image_hosts:
    infra1:
        ip: 10.0.1.11
# nova api, conductor, etc services
compute-infra_hosts:
    infra1:
        ip: 10.0.1.11
# heat
orchestration_hosts:
    infra1:
        ip: 10.0.1.11
# horizon
dashboard_hosts:
    infra1:
        ip: 10.0.1.11
# neutron server, agents (L3, etc)
network_hosts:
    infra1:
        ip: 10.0.1.11
# nova hypervisors
compute_hosts:
    compute1:
        ip: 10.0.1.12
# cinder volume hosts
storage_hosts:
    storage1:
        ip: 10.0.1.13
    container_vars:
        cinder_backends:
            lvm:
                volume_backend_name: LVM
                volume_driver: cinder.volume.drivers.lvm.LVMVolumeDriver
                volume_group: cinder-volumes
                iscsi_ip_address: "10.0.2.13"

```

Particularmente se resalta, en el último punto, la configuración de cinder necesaria para desplegar un backend de almacenamiento basado en LVM.

user__variables.yml

Debido a las limitaciones de red mencionadas anteriormente, es necesario configurar un proxy de salida que será propagado por ansible hacia todos los contenedores y hosts durante la instalación. Para esto se deben configurar las siguientes variables:

```

## Example environment variable setup:
## This is used by apt-cacher-ng to download apt packages:
proxy_env_url: http://10.0.1.1:3128/

```

```

## (1) This sets up a permanent environment, used during and after deployment:
no_proxy_env: "localhost,127.0.0.1,{{ internal_lb_vip_address }},{{ external_lb_vip_address
    }},{% for host in groups['all_containers'] %}{{ hostvars[host]['container_address']
    }}{% if not loop.last %},{% endif %}{% endfor %}"
global_environment_variables:
    HTTP_PROXY: "{{ proxy_env_url }}"
    HTTPS_PROXY: "{{ proxy_env_url }}"
    NO_PROXY: "{{ no_proxy_env }}"
    http_proxy: "{{ proxy_env_url }}"
    https_proxy: "{{ proxy_env_url }}"
    no_proxy: "{{ no_proxy_env }}"
#
## (2) This is applied only during deployment, nothing is left after deployment is complete:
deployment_environment_variables:
    http_proxy: "{{ proxy_env_url }}"
    https_proxy: "{{ proxy_env_url }}"
    no_proxy: "localhost,127.0.0.1,{{ internal_lb_vip_address }},{{ external_lb_vip_address
    }},{% for host in groups['keystone_all'] %}{{ hostvars[host]['container_address']
    }}{% if not loop.last %},{% endif %}{% endfor %}"

```

Además, para poder crear una imagen desde el Horizon remotamente se debe agregar sobre el final del archivo la siguiente directiva:

```
horizon_images_upload_mode: "legacy"
```

Si bien en [49] se menciona que el valor por defecto de esta variable es *legacy*, durante el proceso de instalación se detectó que es configurada como *direct*. La diferencia entre estos modos es que *legacy* permite subir archivos locales desde la máquina del usuario al servidor web de Horizon y luego de este hacia el módulo Glance. Por su parte *direct* evita esta sobrecarga de red y almacenamiento en el servidor web, conectado a través de una API al usuario con el módulo Glance. Sin embargo, esto último requiere de configuraciones extras como tener acceso al puerto 9292 (API de Glance) y un correcto uso de CORS.

cinder.yml

En caso de utilizar un backend de storage LVM se debe indicar que este debe ser deployado en metal, para esto se debe configurar el archivo `/etc/openstack_deploy/env.d/cinder-volume.yml` con lo siguiente:

```

container_skel:
    cinder_volumes_container:
        properties:
            is_metal: true

```

4.4.3. Generación de claves

Se deben configurar las passphrases requeridas por Openstack durante su instalación y posterior uso. Esto se alcanza mediante:

```

$ cd /opt/openstack-ansible
$ ./scripts/pw-token-gen.py --file /etc/openstack_deploy/user_secrets.yml

```

4.4.4. Correcciones

SELinux

Durante el proceso de instalación se detectó un bug asociado a SELinux, el cual fue verificado en [16]. Debido a que en el proyecto OSA se dejó de mantener el uso de SELinux por falta de personal,

se debió aplicar el commit indicado en [29] de la versión Rocky de OSA que desactiva por completo la utilización de este módulo.

Concretamente el commit consiste en:

- Eliminar los siguientes archivos:

```
$ rm /etc/ansible/roles/os_nova/files/osa-nova.te
$ rm /etc/ansible/roles/os_nova/tasks/nova_selinux.yml
```

- Modificar el archivo `/etc/ansible/roles/os_nova/tasks/nova_post_install.yml` eliminando las siguientes líneas:

```
include_tasks: nova_selinux.yml
  when:
    - ansible_selinux.status == "enabled"
```

4.5. Ejecución de playbooks

Finalmente para instalar Openstack con Ansible es necesario correr las playbooks principales del proyecto, las cuales se encuentran en el directorio `/opt/openstack-ansible/playbooks`.

En primer lugar se ejecutan tres scripts para realizar un chequeo de sintaxis de la configuración preparada y los scripts a utilizar. Esto se realiza de la siguiente forma:

```
$ openstack-ansible setup-hosts.yml --syntax-check
$ openstack-ansible setup-infrastructure.yml --syntax-check
$ openstack-ansible setup-openstack.yml --syntax-check
```

En caso de no tener errores, se comienza la ejecución de las playbooks en el orden que se describen:

setup-hosts.yml

Esta playbook se encarga de configurar todos los hosts descritos en el archivo `openstack_user_config.yml`. Con el siguiente comando se ejecuta la playbook:

```
$ openstack-ansible -vvv setup-hosts.yml 2>&1 | tee /var/log/openstack/hostsXX.log
```

La opción `-vvv` es para que la salida sea más verbosa y el final es para mostrar la salida del comando en la consola y almacenarla en un archivo de log.

install-haproxy.yml

La configuración de los balanceadores de carga es lo siguiente a realizar. Esto se puede realizar sin tener explícitamente los contenedores o servidores físicos con los servicios instalados gracias a Ansible. OSA como se mencionó en la sección del inventario, a partir de los archivos de configuración conoce exactamente cómo quedará instalado cada uno de los servicios de Openstack, por ejemplo las IPs y puertos de cada servicio. Con el siguiente comando se ejecuta la playbook:

```
$ openstack-ansible -vvv haproxy-install.yml 2>&1 | tee /var/log/openstack/haproxyXX.log
```

setup-infrastructure.yml

En este paso demora un poco más que el primer setup y se encargará de construir todos los contenedores donde luego se instalarán los servicios de Openstack. Esta script además se encarga de instalar los servicios de infraestructura como son RabbitMQ o Galera DB para luego ser configurados en la playbook final. Con el siguiente comando se ejecuta la playbook:

```
$ openstack-ansible -vvv setup-infrastructure.yml 2>&1 | tee /var/log/openstack/
  infrastructureXX.log
```

setup-openstack.yml

En este paso final es cuando se configuran todos los servicios, indicados en los archivos de configuración de OSA, de Openstack. Esta playbook es la que demora más tiempo en su ejecución, en el orden de las horas. Con el siguiente comando se ejecuta la playbook:

```
$ openstack-ansible -vvv setup-openstack.yml 2>&1 | tee /var/log/openstack/openstackXX.log
```

4.6. Verificación

Luego de que la última playbook haya terminado su ejecución sin error, debemos verificar que la instalación fue exitosa. Esto se realizará de forma manual siguiendo los pasos que se indican a continuación:

1. Acceder al nodo de infra1 como usuario root.
2. La instalación que realiza OSA crea contenedores de utilidad los cuales proveen de todas las herramientas desde la consola para utilizar Openstack. En primer lugar se deben listar todos los containers del nodo físico ejecutando:

```
$ lxc-ls -f
```

3. El contenedor que se utiliza es el que tiene en su nombre la palabra utility, para acceder al mismo es necesario ejecutar el siguiente comando de lxc:

```
$ lxc-attach -n <nombre_contenedor>
```

4. Para utilizar los servicios de Openstack es necesario enviar las credenciales del usuario que invocará a las APIs de los servicios. Esto se debe al funcionamiento de Openstack en donde cada llamada a una API debe ser validada por el módulo de keystone. Para evitar escribir las credenciales en cada comando, OSA genera un archivo llamado openrc para cargar la información del usuario como variables de entorno. El archivo se carga con el siguiente comando:

```
$ source openrc
```

5. Algunos comandos que se pueden ejecutar son:

- Para listar usuarios:

```
$ openstack user list --os-cloud=default
```

- Para listar servidores:

```
$ openstack server list
```

- Para listar redes:

```
$ openstack network list
```

- Para listar los agentes de red:

```
$ openstack network agent list
```

6. Por otro lado se puede verificar el dashboard de horizon accediendo a la ip definida en `external_lb_vip_address` en el archivo `/etc/openstack_deploy/openstack_user_config.yml` en el puerto 443 dado que utiliza HTTPS. Para autenticarse como admin es necesaria la password que se encuentra definida en la opción `keystone_auth_admin_password` del archivo `/etc/openstack_deploy/user_secrets.yml`.

5

Interacción

Esta sección se realizó utilizando el dashboard de Openstack, brindado por el módulo Horizon. Debido a las limitaciones de red ya mencionadas, es necesario realizar un reenvío de puertos mediante SSH para acceder al mismo. Esto se logra en dos pasos mediante:

1. Primero se realiza un forwarding desde la máquina local hasta lulu:

```
$ ssh -L 2443:localhost:2443 <usuario_fing>@lulu.fing.edu.uy
```

2. Una vez dentro de lulu, se realiza un nuevo forwarding desde la misma hasta la IP pública de el balanceador, a través del servidor renata.

```
$ ssh -L 2443:192.168.60.160:443 openstack@192.168.60.242 -4
```

Luego, accediendo a través de un navegador a la dirección <https://localhost:2443> se llega a la vista de login de Openstack.

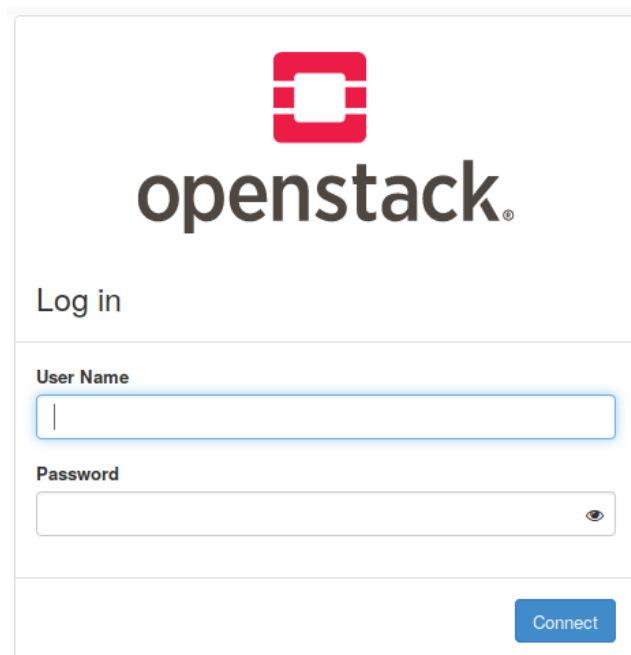


Figura 5.1: Vista del login de Horizon.

Una vez allí, se debe acceder con la cuenta de administrador para realizar las configuraciones iniciales utilizando las credenciales indicadas en el último paso de la sección de verificación.

5.1. Configuraciones de administrador

Para que los usuarios finales puedan operar sobre la plataforma Openstack, se deben configurar ciertos aspectos como lo son proyectos, usuarios, flavors y redes provider, entre otros. A continuación se presenta un instructivo básico para ello.

Crear proyecto

Lo primero a configurar es crear un proyecto sobre el cual se realizarán las pruebas siguientes. Para ello en el menú lateral se accede a Identity >Projects >Create Project, completando los campos como se muestra en la figura 5.2.

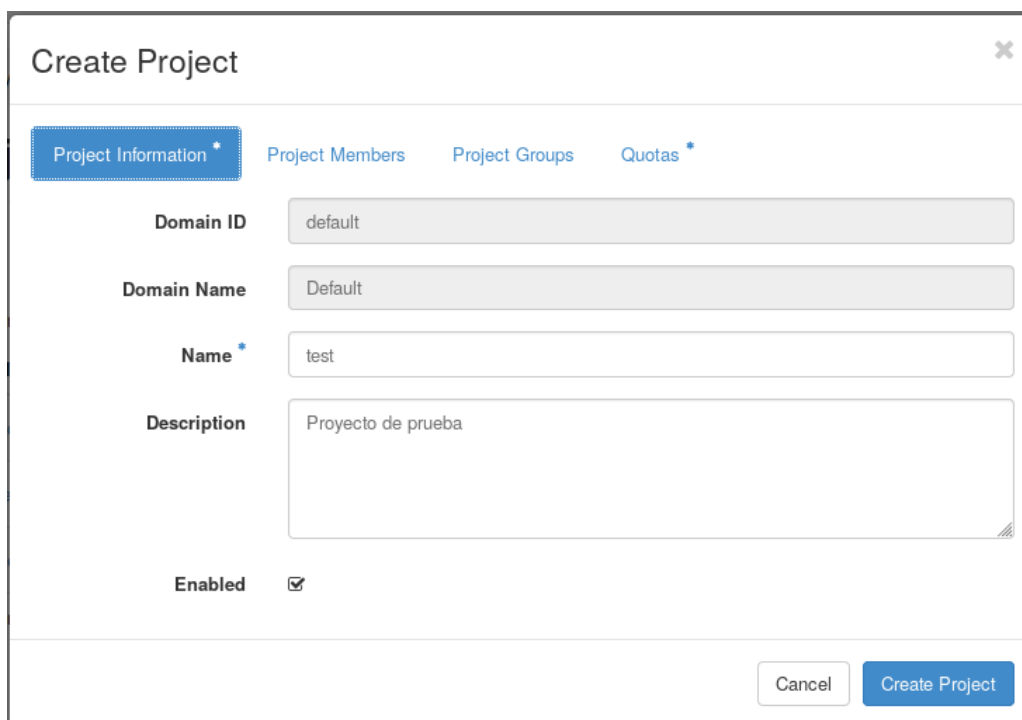


Figura 5.2: Creación de un proyecto (1/2).

Antes de confirmar la creación, en la pestaña Project Members agregamos al admin como miembro del proyecto.

Create Project

Project Information * **Project Members** Project Groups Quotas *

All Users Filter Q

cinder	+
placement	+
nova	+
keystone	+
neutron	+
heat	+
glance	+

Project Members Filter Q

admin	member ▾	-
-------	----------	---

Cancel Create Project

Figura 5.3: Creación de un proyecto (2/2).

Crear usuario

Luego se crea un usuario de pruebas accediendo nuevamente por la pestaña lateral a Identity > Users > Create User, rellenando los campos solicitados como se presenta en la figura 5.4.

Create User [X]

Domain ID
default

Domain Name
Default

User Name *
test

Description
Usuario de prueba

Email

Password *

Confirm Password *

Primary Project
test

Role
member

☒ **Enabled**

Description:
Create a new user and set related properties including the Primary Project and Role.

Cancel Create User

Figura 5.4: Creación de un usuario.

Por más detalles se puede visitar [39].

Crear flavor

El siguiente paso será crear un flavor de pruebas con algunas características básicas y permitirle acceso al proyecto de test creado. Se accede desde Admin > Compute > Flavors. Esto se ilustra en las figuras 5.5 y 5.6. Más detalles en [38].

Create Flavor

Flavor Information *

Flavor Access

Name *

test-1

ID ⓘ

auto

VCPUs *

2

RAM (MB) *

4

Root Disk (GB) *

50

Ephemeral Disk (GB)

0

Swap Disk (MB)

0

RX/TX Factor

1

Flavors define the sizes for RAM, disk, number of cores, and other resources and can be selected when users deploy Instances.

Cancel

Create Flavor

Figura 5.5: Creación de un flavor (1/2).

Create Flavor

Flavor Information *

Flavor Access

Select the projects where the flavors will be used. If no projects are selected, then the flavor will be available in all projects.

All Projects

Filter

Q

admin

+

service

+

Selected Projects

Filter

Q

test

-

Cancel

Create Flavor

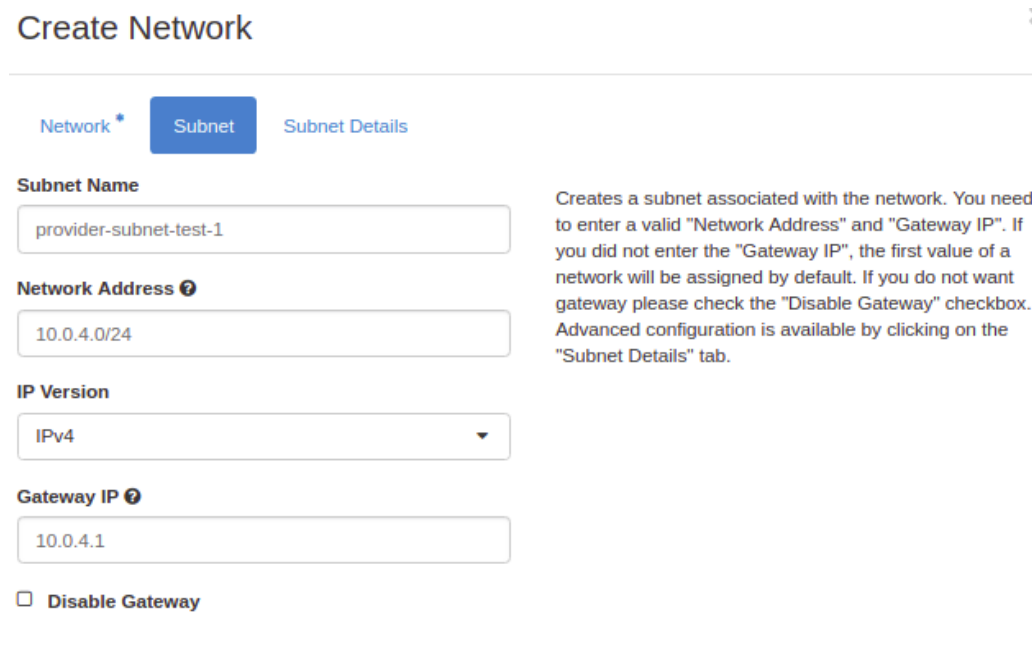
Figura 5.6: Creación de un flavor (2/2).

Crear provider network

Este tipo de red es manejado por los administradores y para crearlas se debe acceder a Admin >Network >Networks >Create Network, completando el formulario como se muestra en las imágenes 5.7 y 5.8.

Figura 5.7: Creación de una red provider (1/2).

El valor del campo Physical Network es el especificado en la red de tipo “flat” en la sección de provider networks del archivo `openstack_user_config.yml`.



Create Network

Network * Subnet Subnet Details

Subnet Name

provider-subnet-test-1

Network Address ?

10.0.4.0/24

IP Version

IPv4

Gateway IP ?

10.0.4.1

☐ Disable Gateway

Creates a subnet associated with the network. You need to enter a valid "Network Address" and "Gateway IP". If you did not enter the "Gateway IP", the first value of a network will be assigned by default. If you do not want gateway please check the "Disable Gateway" checkbox. Advanced configuration is available by clicking on the "Subnet Details" tab.

Figura 5.8: Creación de una red provider (2/2).

5.2. Interacción de un usuario

A partir de ahora, el ambiente se encuentra con las configuraciones mínimas necesarias para que el usuario test pueda acceder y hacer uso de la plataforma. Por lo tanto, los pasos a continuación se realizan sobre el proyecto de prueba, habiéndose autenticado previamente con el usuario test.

Crear imagen

Lo primero que deberá hacer el usuario para crear una instancia es crear la imagen que será utilizada por esta. Desde [30] se pueden descargar formatos de imágenes soportados por Openstack de la mayoría de los sistemas operativos Linux. La creación de la imagen se realiza desde el menú Project >Compute >Images >Create Image.

Create Image

Image Details

Metadata

Image Details

Specify an image to upload to the Image Service.

Image Name

image-test-1

Image Description

Imagen de prueba

Image Source

Source Type

File

File

Browse...

cirros-0.4.0-x86_64-disk.img

Format

QCOW2 - QEMU Emulator

Image Requirements

Kernel

Choose an image

Ramdisk

Choose an image

Architecture

x86_64

Minimum Disk (GB)

10

Minimum RAM (MB)

2048

Image Sharing

Protected

Yes No

Cancel


< Back

Next >


Create Image


Figura 5.9: Creación de una imagen (1/2).


En la primer pantalla se establecen los datos básicos para la creación de la imagen y en la siguiente se puede establecer metatada más específica de la misma.


You can specify resource metadata by moving items from the left column to the right column. In the left column there are metadata definitions from the Glance Metadata Catalog. Use the "Custom" option to add metadata with the key of your choice. 


Available Metadata


Filter 


Custom 


> CIM Processor Allocation Setting 


> Cinder Volume Type 


> CPU Pinning 

> Database Software 


> Guest Memory Backing 

> Hypervisor Selection 

> Image Signature Verification 

> Instance Config Data 

Existing Metadata

Filter 

No existing metadata

Click each item to get its description here.

< Back

Next >


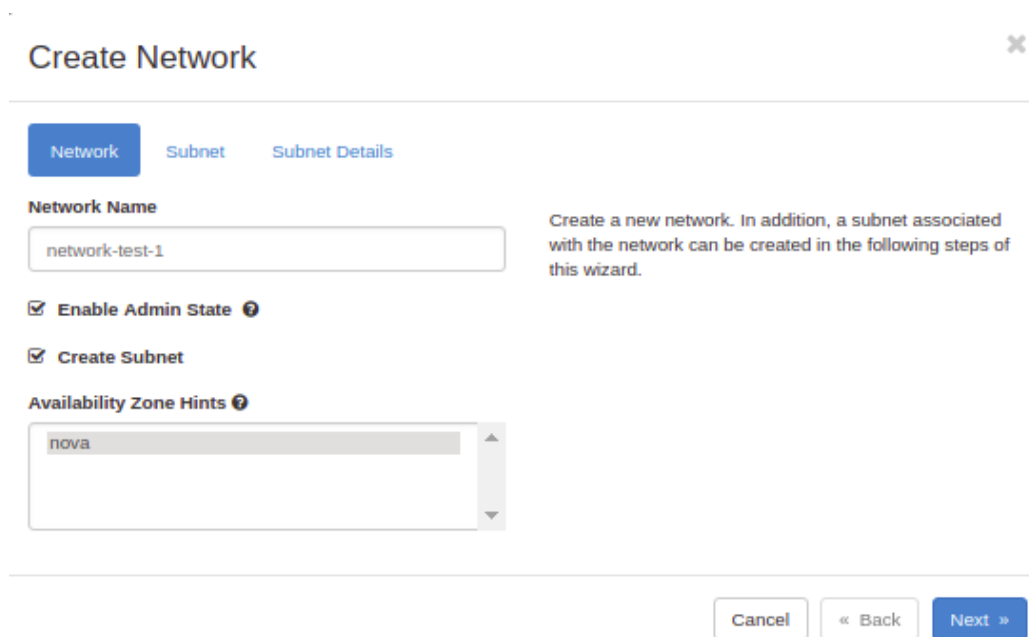
 Create Image

Figura 5.10: Creación de una imagen (2/2).

Crear red

La creación de nuevas subredes se realiza accediendo a Project >Network >Networks >Create Network. Esta funcionalidad consta de tres pasos detallados a continuación:

63



Create Network ✕

Network Subnet Subnet Details

Network Name
network-test-1

☒ **Enable Admin State** ⓘ

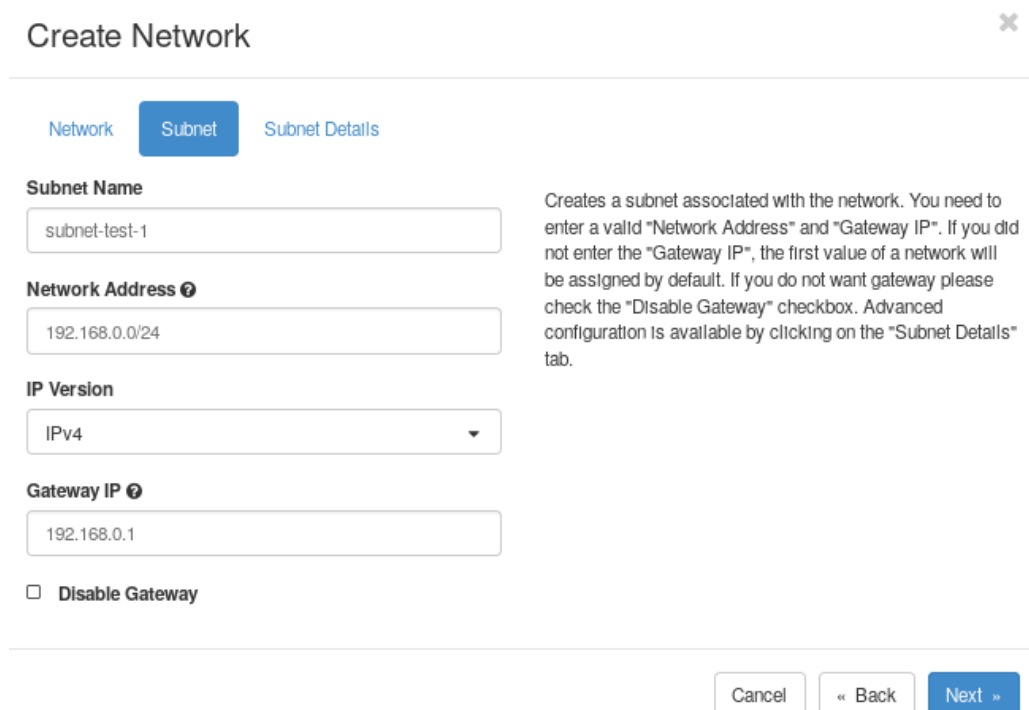
☒ **Create Subnet**

Availability Zone Hints ⓘ
nova

Create a new network. In addition, a subnet associated with the network can be created in the following steps of this wizard.

Cancel « Back Next »

Figura 5.11: Creación de una red (1/3).



Create Network ✕

Network **Subnet** Subnet Details

Subnet Name
subnet-test-1

Network Address ⓘ
192.168.0.0/24

IP Version
IPv4

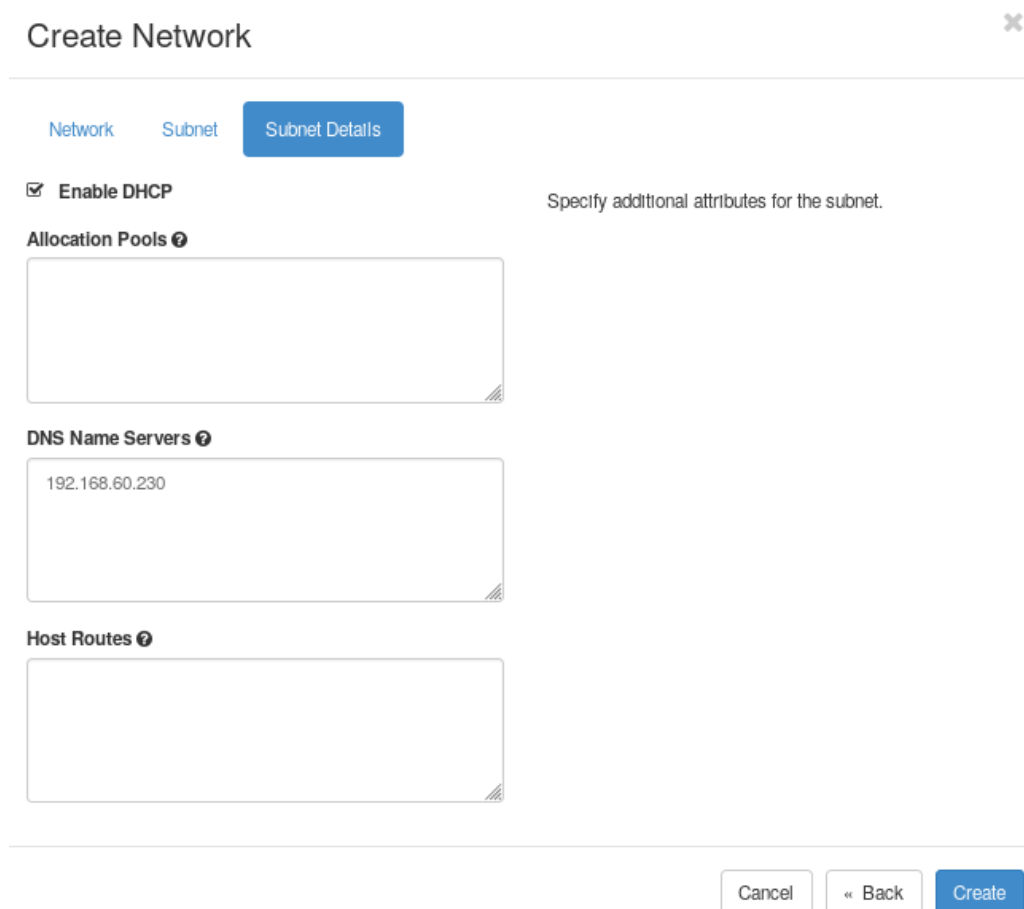
Gateway IP ⓘ
192.168.0.1

☐ **Disable Gateway**

Creates a subnet associated with the network. You need to enter a valid "Network Address" and "Gateway IP". If you did not enter the "Gateway IP", the first value of a network will be assigned by default. If you do not want gateway please check the "Disable Gateway" checkbox. Advanced configuration is available by clicking on the "Subnet Details" tab.

Cancel « Back Next »

Figura 5.12: Creación de una red (2/3).

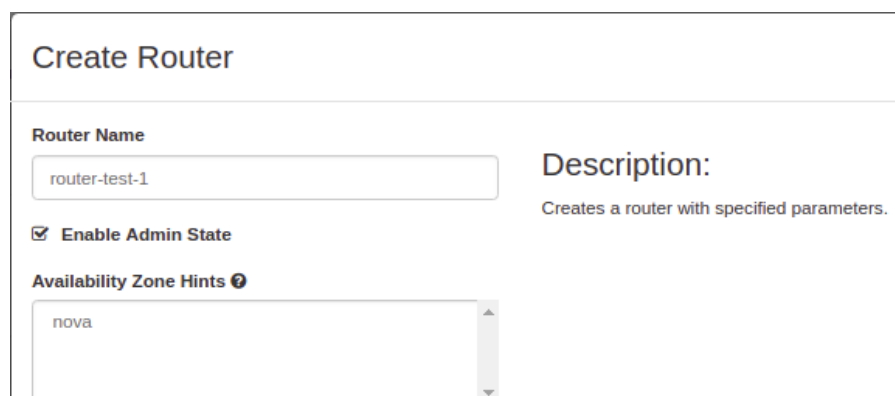


The screenshot shows the 'Create Network' form. At the top, there's a title 'Create Network' and a close button (X). Below the title, there are three tabs: 'Network', 'Subnet', and 'Subnet Details'. The 'Subnet Details' tab is selected. Under this tab, there's a checkbox 'Enable DHCP' which is checked. To the right of this checkbox, it says 'Specify additional attributes for the subnet.' Below this, there are three sections: 'Allocation Pools' with an empty text area, 'DNS Name Servers' with a text area containing '192.168.60.230', and 'Host Routes' with an empty text area. At the bottom right, there are three buttons: 'Cancel', '« Back', and 'Create'.

Figura 5.13: Creación de una red (3/3).

Crear router

La creación de nuevos routers se realiza accediendo a Project >Network >Routers >Create Router.



The screenshot shows the 'Create Router' form. It has a title 'Create Router'. Below the title, there's a 'Router Name' field with the value 'router-test-1'. To the right of this field, there's a 'Description:' label and a text area with the value 'Creates a router with specified parameters.' Below the 'Router Name' field, there's a checkbox 'Enable Admin State' which is checked. Below this, there's a section 'Availability Zone Hints' with a text area containing the value 'nova'.

Figura 5.14: Creación de un router.

Más detalles de estas configuraciones se pueden encontrar en [27].

Crear interfaz de router

Ir a Network >Routers y en la grilla desplegada seleccionar el router. En la pestaña de Interfaces se encuentra la opción para crear una nueva interfaz.

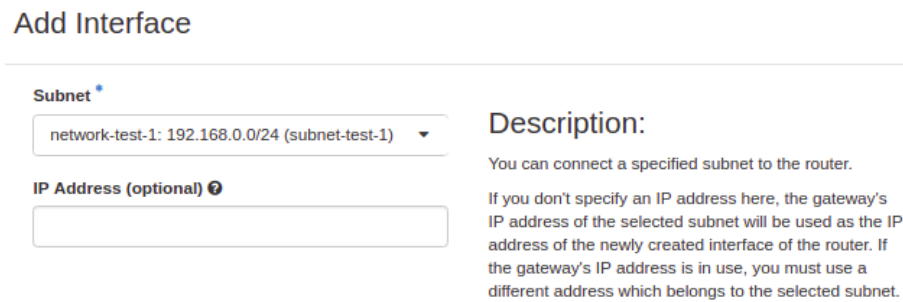


Figura 5.15: Creación de una interfaz en un router.

Para la creación de una interfaz se debe ingresar la subred a la que estará conectada y opcionalmente la IP de la misma.

Crear key pair

La creación de key pairs se realiza en Project > Compute > Key Pairs > Create Key Pair.

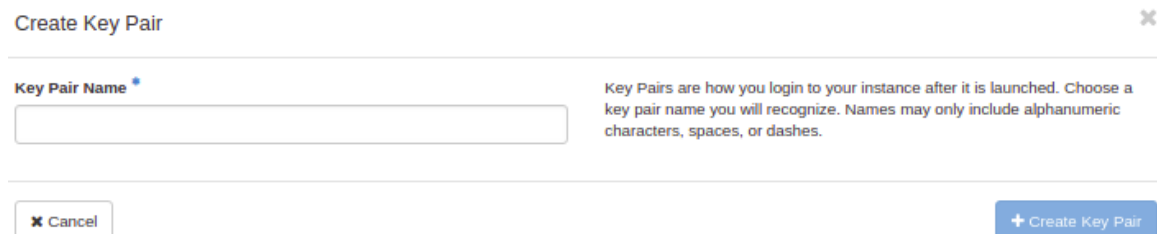


Figura 5.16: Creación de una key pair.

Luego al momento de crear una nueva instancia se debe seleccionar una clave. Se debe tener en cuenta que la asignación de key pairs a las instancias mediante Horizon solamente se puede realizar en el momento de creación de las mismas. Más detalles se encuentran en [24].

Lanzar una instancia

La creación de nuevas instancias se realiza en Project > Instances > Launch Instance. En primer lugar se especifican aspectos básicos como el nombre y la descripción 5.17.

Launch Instance

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.

Instance Name *
Instance-test-1

Description
Instancia de prueba 1

Availability Zone
nova

Count *
1

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

Details
Source *
Flavor *
Networks *
Network Ports
Security Groups
Key Pair
Configuration
Server Groups
Scheduler Hints
Metadata

Buttons: Cancel, < Back, Next >, Launch Instance

Figura 5.17: Lanzar una nueva instancia (1/5).

Luego se deberá indicar la imagen a ser utilizada para bootear la máquina. En este caso se elige la imagen imagen-test-1 creada anteriormente.

Launch Instance

Instance source is the template used to create an instance. You can use an Image, a snapshot of an instance (image snapshot), a volume or a volume snapshot (if enabled). You can also choose to use persistent storage by creating a new volume.

Select Boot Source
Image

Volume Size (GB) *
10

Create New Volume
Yes No

Delete Volume on Instance Delete
Yes No

Allocated

Name	Updated	Size	Type	Visibility
Imagen-test-1	8/3/19 10:08 AM	12.13 MB	Iso	Private

Available 2

Click here for filters.

Name	Updated	Size	Type	Visibility
centos-7	8/3/19 10:28 AM	918.00 MB	Iso	Public
Image1	8/3/19 9:48 AM	12.13 MB	Iso	Public

Buttons: Cancel, < Back, Next >, Launch Instance

Figura 5.18: Lanzar una nueva instancia (2/5).

A continuación se determinan los recursos virtuales de la instancia mediante la selección de un flavor.

Launch Instance

Details

Source

Flavor

Networks *

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

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

Allocated

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
> test-1	2	4 GB	50 GB	50 GB	0 GB	No

Available 0

Select one

Click here for filters.

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
------	-------	-----	------------	-----------	----------------	--------

< Back Next > Launch Instance

Figura 5.19: Lanzar una nueva instancia (3/5).

El siguiente paso requerido es la selección de una red.

Launch Instance

Details

Source

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Networks provide the communication channels for Instances in the cloud.

Allocated 1

Select networks from those listed below.

Network	Subnets Associated	Shared	Admin State	Status
> 1 network-test-1	subnet-test-1	No	Up	Active

Available 1

Select at least one network

Click here for filters.

Network	Subnets Associated	Shared	Admin State	Status
> network-test-2	subnet-test-2	No	Up	Active

< Back Next > Launch Instance

Figura 5.20: Lanzar una nueva instancia (4/5).

Finalmente, la última configuración mínimamente requerida para acceder en forma remota a la

instancia es configurar una key pair.

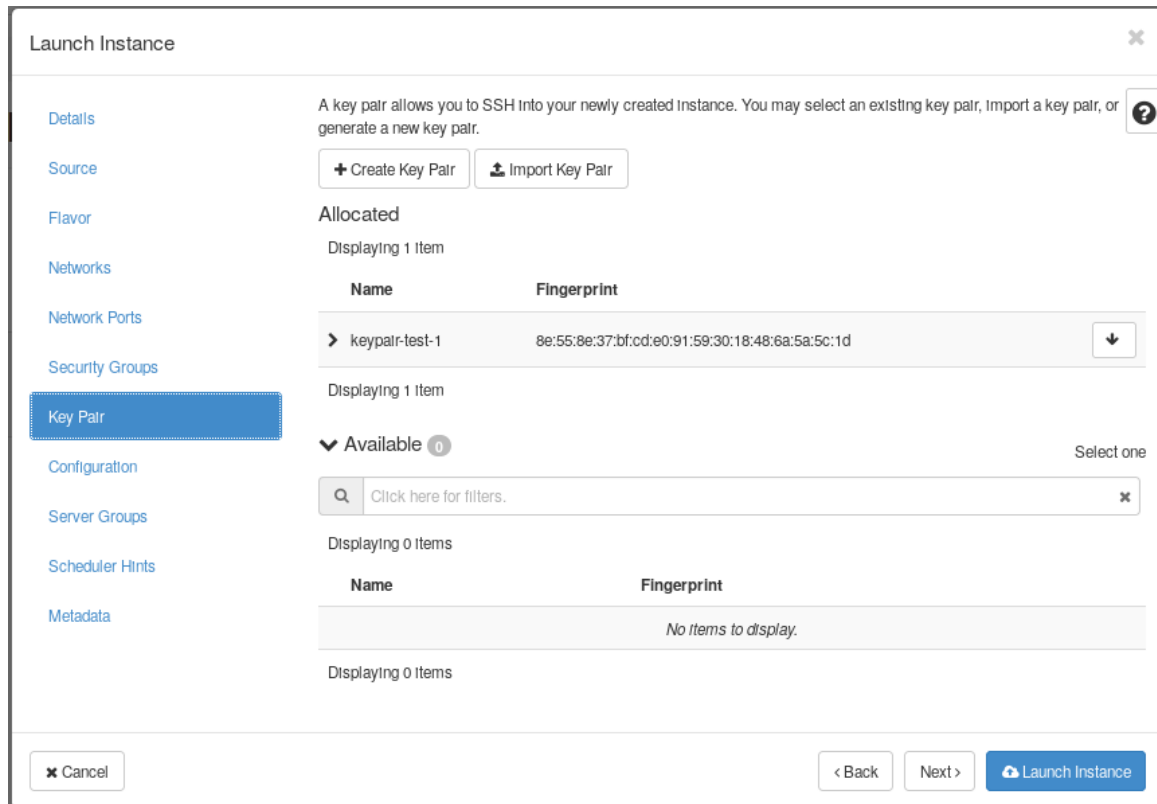


Figura 5.21: Lanzar una nueva instancia (5/5).

Más detalles sobre el lanzamiento de una instancia se pueden ver accediendo a [36].

5.3. Acceso a una instancia

5.3.1. Por SPICE

Para acceder a la consola de la instancia creada a través del protocolo SPICE [57], es necesario dirigirse a Project > Compute > Instances > instances-test-1. Allí en la pestaña console deberíamos obtener el acceso, sin embargo debido a las limitaciones de red ya conocidas es necesario realizar un nuevo forwarding de puertos (en este caso el 6082) y acceder a través de localhost. Para esto:

1. Primero se realiza un forwarding desde la máquina local hasta lulu:

```
$ ssh -L 6082:localhost:6082 <usuario_fing>@lulu.fing.edu.uy
```

2. Una vez dentro de lulu, se realiza un nuevo forwarding desde la misma hasta la IP pública de el balanceador, a través del servidor renata.

```
$ ssh -L 6082:192.168.60.160:6082 openstack@192.168.60.242 -4
```

Luego en la pestaña de console, abrimos el link asociado a '*Click here to show only console*', esto intentará cargar en el navegador una URL similar a la siguiente:
[https://192.168.60.160:6082/spice_auto.html?token=a3703173-3973-440d-babf-f8b662b2fd25&title=instance-test-1\(430c92de-3cab-4b57-be7e-6394793dc423\)](https://192.168.60.160:6082/spice_auto.html?token=a3703173-3973-440d-babf-f8b662b2fd25&title=instance-test-1(430c92de-3cab-4b57-be7e-6394793dc423))

En la misma debemos sustituir la IP 192.168.60.160 por localhost para utilizar el forwarding realizado. Allí lograremos acceder a la consola SPICE.

5.3.2. Por SSH

Para lograr acceder en forma externa por SSH son necesarias algunas configuraciones extras.

Asociar una Floating IP a la instancia

Estas IPs son necesarias para poder acceder a las instancias desde redes externas. A continuación se detalla cómo realizarlo a partir de Project >Network >Floating IPs >Allocate IP To Project:

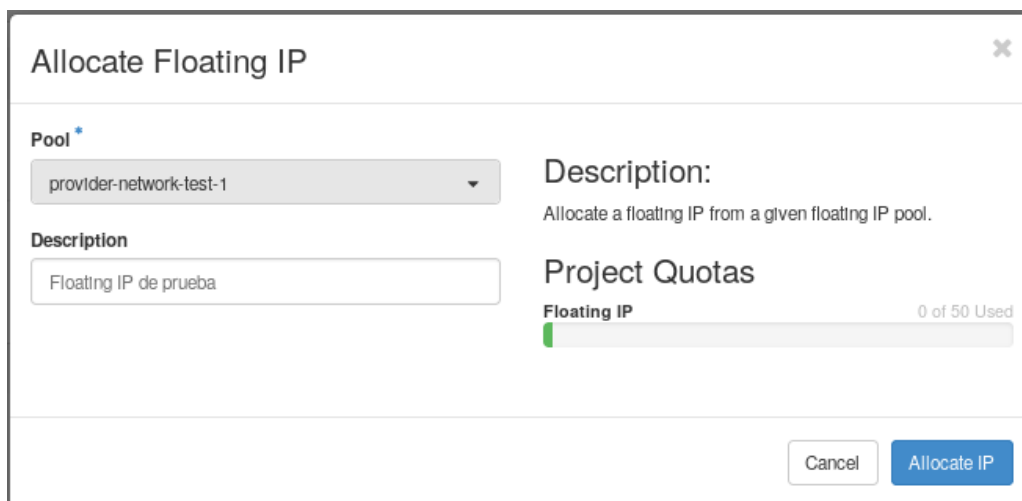


Figura 5.22: Asignación de floating IP.

Luego se debe asociar con el puerto de la instancia creada desde Project >Network >Floating IPs >Associate.

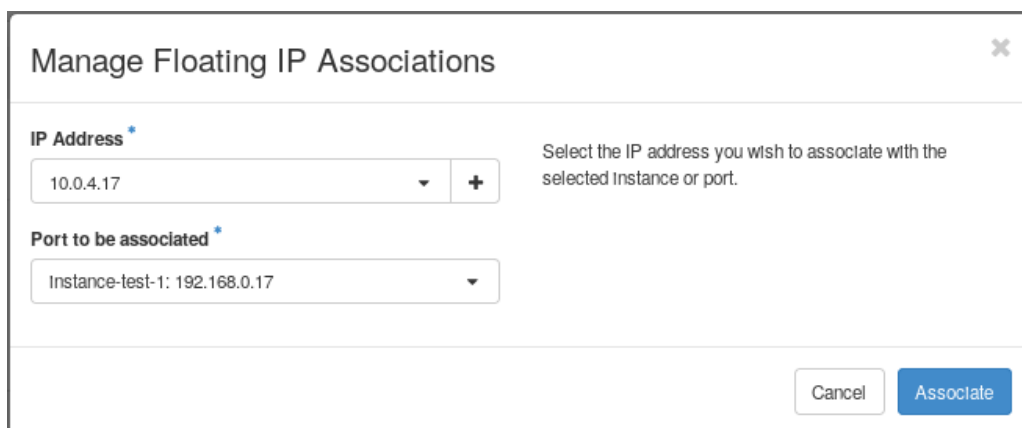


Figura 5.23: Asociación de floating IP.

Modificar security group

Los security groups en Openstack son firewalls virtuales en donde se definen una serie de reglas a ser aplicadas al tráfico de las instancias. El grupo creado por defecto en la instalación de Openstack contiene las siguientes reglas:

Displaying 4 items

<input type="checkbox"/> Direction ▾	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group
<input type="checkbox"/> Egress	IPv4	Any	Any	0.0.0.0/0	-
<input type="checkbox"/> Egress	IPv6	Any	Any	:::0	-
<input type="checkbox"/> Ingress	IPv4	Any	Any	-	default
<input type="checkbox"/> Ingress	IPv6	Any	Any	-	default

Figura 5.24: Reglas security group por defecto.

Estas reglas permiten el tráfico de salida hacia cualquier IP y solamente se permite el tráfico de entrada de instancias del mismo security group.

Para tener conectividad SSH e ICMP con las instancias creadas se deben agregar las siguientes reglas accediendo por Project >Network >Security Groups >Manage Rules >Add Rule.

Add Rule

Rule *

All ICMP ▾

Direction

Ingress ▾

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 5.25: Agregar regla para tráfico ICMP.

Add Rule ✕

Rule *

SSH ▼

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.

Figura 5.26: Agregar regla para tráfico SSH.

SSH

Finalmente, para acceder a la consola de la instancia creada a través de SSH desde el servidor renata es necesario copiar la clave generada en el paso de key pair y luego acceder mediante:

```
$ ssh -i keypair-test-1.pem cirros@10.0.4.17
```

Una vez dentro, para tener conectividad a internet se debe configurar el mismo proxy que fue asignado a todos los nodos de Openstack, es decir:

```
$ export http_proxy="http://10.0.1.1:3128"
```

6

Inconvenientes

Bloqueo de paquetes

En los servidores virtuales y el servidor físico las reglas por defecto del firewall de CentOS bloquean tanto el tráfico utilizado para interconectar los servicios de Openstack como el empleado para las conexiones con redes externas. Para solucionar esto de forma momentánea se eliminaron estas reglas con los comandos:

```
$ iptables -D INPUT -j REJECT --reject-with icmp-host-prohibited
$ iptables -D FORWARD -j REJECT --reject-with icmp-host-prohibited
```

Módulo de seguridad SELinux

Como se mencionó en la sección de correcciones, OSA ha perdido el mantenimiento de este módulo de seguridad, por lo cual fue necesario aplicar los parches realizados para la siguiente versión (Rocky) a la utilizada en la instalación (Queens) para discontinuar el uso de dicho módulo.

Percona-release en playbook setup-infrastructure

Durante la instalación de la playbook mencionada se detecta el siguiente error a la hora de instalar los componentes del contenedor de galera :

```
warning: /var/cache/yum/x86_64/7/percona-release-x86_64/packages/qpress-11-1.el7.x86_64.rpm:
  Header V4 RSA/SHA256 Signature, key ID 8507efa5: NOKEY
The GPG keys listed for the \"Percona-Release YUM repository - x86_64\" repository are
  already installed but they are not correct for this package.
Check that the correct key URLs are configured for this repository.
Failing package is: qpress-11-1.el7.x86_64
GPG Keys are configured as: file:///etc/pki/rpm-gpg/RPM-GPG-KEY-Percona
```

La solución para este problema es actualizar el paquete percona-release, dentro del contenedor de galera. Para esto se deben seguir los siguientes pasos:

1. Acceder por ssh al nodo infra1

```
$ ssh root@10.0.1.11
```

2. Listar los contenedores LXC existentes hasta el momento y obtener el nombre del contenedor pertinente:

```
$ lxc-ls galera
```

3. Acceder a dicho contenedor:

```
$ lxc-attach -n infra1_galera_container-15357d7d
```

4. Actualizar el paquete mencionado

```
$ yum upgrade percona-release -y
```

5. Volver a ejecutar la playbook setup-infrastructure.

Subred reservada

Debido a que no se encuentra especificado en la documentación oficial de Openstack-Ansible, en las primeras instalaciones realizadas la subred definida para la red de tunelización (vxlan) fue la 10.0.3.0/24. Esto generaba grandes inconsistencias de red que no tenían una causa identificada claramente. Luego de un extenso análisis de la situación se logró determinar que la red creada internamente por Openstack para la comunicación entre contenedores LXC utiliza dicho rango. En función de esta observación es que se especificó el rango 10.0.10.0/24 para la red vxlan.

7

Trabajo a futuro

Firewall

Deshabilitar por completo la denegación de paquetes por defecto en el firewall de CentOS deja el servidor expuesto ante vulnerabilidades, lo cual para un ambiente de producción es inadmisibles. Como aspecto a mejorar se propone analizar y documentar todas las conexiones HTTP realizadas por Openstack Ansible durante su instalación y en su posterior ejecución entre cada uno de los nodos involucrados. Una vez identificadas, se debe agregar en cada nodo las reglas iptables correspondientes que habiliten dichas conexiones en particular.

Arquitectura segura

Se propone evaluar diversos diseños de arquitectura que involucren la utilización de un firewall que separe adecuadamente la red interna de la pública utilizando opcionalmente una DMZ. Esta evaluación debe tener en cuenta tanto accesos SSH a los nodos del datacenter por parte de administradores, como accesos directos a las instancias utilizadas por los usuarios finales. Además, debe permitir el tráfico HTTP para la correcta interacción con el dashboard de la plataforma.

A la hora de tomar una decisión se debe considerar que de momento no se cuenta con una conexión directa hacia el exterior, por lo que en caso de que esto ocurra se debe reforzar adecuadamente la seguridad del datacenter.

Brindar conexión directa a Internet

Con el fin de simplificar el acceso hacia el exterior tanto durante la instalación como en su posterior ejecución, se debe evaluar una solución que permita mantener una conexión directa con el proxy de la FIng, evitando así la necesidad de realizar un túnel SSH como fue mencionado en la descripción del ambiente de trabajo.

Gestión de Openstack en operación

Es sumamente relevante investigar la gestión en caliente de un datacenter implementado mediante Openstack. Esto involucra tener un manual con pasos claros a realizar en circunstancias tales como escalar horizontalmente el datacenter, reemplazar nodos core y recuperar el funcionamiento ante eventos externos.

Conclusiones

Luego de haber realizado todo el proceso de despliegue de un datacenter de pruebas mediante Openstack, se tomó conciencia de todos los aspectos y conceptos que involucra la instalación y mantenimiento de una operativa de esta magnitud. Es por esto que para llevar a cabo el trabajo fue necesario adquirir y aplicar conocimientos en diversas áreas computacionales tales como virtualización y contenerización, gestión de redes, almacenamiento de datos y administración de sistemas.

Openstack como herramienta resulta ser muy valorada gracias a que posee una gran flexibilidad para adecuar el despliegue de un datacenter a las necesidades de cada caso, permitiendo instalar módulos con funcionalidades específicas y prescindir de aquellos que no sean necesarios.

El proceso de instalación resulta sumamente complejo debido a la cantidad de herramientas y configuraciones a tener en cuenta, es por eso que la utilización de una herramienta de automatización de tareas resulta inevitable. Como se menciona en el documento, se utilizó Ansible para facilitar el proceso ya que permite ejecutar instalaciones reiteradas veces de forma idéntica y sobre múltiples servidores sin una carga operativa adicional. Aún así, la experiencia no fue sencilla debido a los diversos inconvenientes que se encontraron durante el camino. Esto condujo a que lograr todo el trabajo llevará un tiempo mayor al estimado.

Finalmente se resalta el gran apoyo e inversión a nivel internacional en esta plataforma, y los niveles de crecimiento que tuvo y continúa teniendo en esta década, indicando que es algo por lo que apostar a futuro.

Bibliografía

- [1] 802.1Q-2014 - Bridges and Bridged Networks. <http://www.ieee802.org/1/pages/802.1Q-2014.html>. Accedido: 2019-06-15.
- [2] Containers on Virtual Machines or Bare Metal? White Paper, VMware, 2018.
- [3] Ansible. Module Index. https://docs.ansible.com/ansible/latest/modules/modules_by_category.html. Accedido: 2019-07-05.
- [4] ArchLinux. Linux Containers. https://wiki.archlinux.org/index.php/Linux_Containers. Accedido: 2019-06-20.
- [5] Cisco. What is a Data Center. <https://www.cisco.com/c/en/us/solutions/data-center-virtualization/what-is-a-data-center.html>. Accedido: 2019-06-20.
- [6] James Denton. *Learning Openstack Networking*, chapter 1 Introduction to Openstack Networking, page 15. Packt Publishing, 3rd edition, 2018.
- [7] James Denton. *Learning Openstack Networking*, chapter 1 Introduction to Openstack Networking. Packt Publishing, 3rd edition, 2018.
- [8] James Denton. *Learning Openstack Networking*, chapter 1 Introduction to Openstack Networking, pages 21–23. Packt Publishing, 3rd edition, 2018.
- [9] James Denton. *Learning Openstack Networking*, chapter 1 Introduction to Openstack Networking, page 31. Packt Publishing, 3rd edition, 2018.
- [10] Docker. What is a Container? <https://www.docker.com/resources/what-container>. Accedido: 2019-06-20.
- [11] Kevin Jackson, Cody Bunch, Egle Sigler, and James Denton. *Openstack Cloud Computing Cookbook*, chapter 1 Installing Openstack with Ansible, page 16. Packt Publishing, 4th edition, 2018.
- [12] Mallik Mahalingam, Dinesh Dutt, Kenneth Duda, Puneet Agarwal, Larry Kreeger, T. Sridhar, Mike Bursell, and Chris Wright. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. RFC 7348, August 2014.
- [13] Steve Martinelli, Henry Nash, and Brad Topol. *Identity, Authentication, and Access Management in OpenStack: Implementing and Deploying Keystone*, chapter 1 Fundamental Keystone Topics, page 13. O'Reilly Media, Inc., 1st edition, 2015.
- [14] Steve Martinelli, Henry Nash, and Brad Topol. *Identity, Authentication, and Access Management in OpenStack: Implementing and Deploying Keystone*. O'Reilly Media, Inc., 1st edition, 2015.
- [15] Peter M. Mell and Timothy Grance. SP 800-145, The NIST Definition of Cloud Computing. Special Publication, National Institute of Standards & Technology, 2011.

- [16] Openstack. AIO Build Fails on SELinux File Context Tasks. <https://bugs.launchpad.net/openstack-ansible/+bug/1782798>. Accedido: 2019-07-05.
- [17] Openstack. Appendix E: Container networking. <https://docs.openstack.org/project-deploy-guide/openstack-ansible/ocata/app-networking.html#network-diagrams>. Accedido: 2019-07-05.
- [18] Openstack. Basic architecture. <https://docs.openstack.org/glance/queens/contributor/architecture.html>. Accedido: 2019-07-05.
- [19] Openstack. Basic networking. <https://docs.openstack.org/mitaka/networking-guide/intro-basic-networking.html>. Accedido: 2019-06-15.
- [20] Openstack. Common Image Properties. <https://docs.openstack.org/glance/queens/user/common-image-properties.html>. Accedido: 2019-06-20.
- [21] Openstack. Components. <https://docs.openstack.org/swift/latest/admin/objectstorage-components.html>. Accedido: 2019-07-05.
- [22] Openstack. Compute service overview. <https://docs.openstack.org/nova/latest/install/get-started-compute.html>. Accedido: 2019-06-20.
- [23] Openstack. Conductor as a place for orchestrating tasks. <https://docs.openstack.org/nova/rocky/user/conductor.html>. Accedido: 2019-06-20.
- [24] Openstack. Configure access and security for instances. <https://docs.openstack.org/horizon/latest/user/configure-access-and-security-for-instances.html>. Accedido: 2019-08-03.
- [25] Openstack. Container networking. <https://docs.openstack.org/openstack-ansible/rocky/reference/architecture/container-networking.html>. Accedido: 2019-07-05.
- [26] Openstack. Control plane architecture. <https://docs.openstack.org/arch-design/design-control-plane.html>. Accedido: 2019-06-20.
- [27] Openstack. Create and manage networks. <https://docs.openstack.org/horizon/latest/user/create-networks.html>. Accedido: 2019-08-03.
- [28] Openstack. Design. <https://docs.openstack.org/arch-design/design.html>. Accedido: 2019-06-20.
- [29] Openstack. Drop SELinux support for CentOS 7. <https://review.opendev.org/#/c/603860/>. Accedido: 2019-07-05.
- [30] Openstack. Get images. <https://docs.openstack.org/image-guide/obtain-images.html>. Accedido: 2019-08-04.
- [31] Openstack. Images and instances. <https://docs.openstack.org/glance/queens/admin/troubleshooting.html>. Accedido: 2019-06-20.
- [32] Openstack. Introduction. <https://docs.openstack.org/project-team-guide/introduction.html>. Accedido: 2019-06-15.
- [33] Openstack. Introduction to Object Storage. <https://docs.openstack.org/swift/latest/admin/objectstorage-intro.html>. Accedido: 2019-07-05.
- [34] Openstack. Inventory. <https://docs.openstack.org/openstack-ansible/rocky/reference/inventory/inventory.html>. Accedido: 2019-06-20.

- [35] Openstack. Keystone Architecture. <https://docs.openstack.org/keystone/latest/getting-started/architecture.html>. Accedido: 2019-06-20.
- [36] Openstack. Launch and manage instances. <https://docs.openstack.org/horizon/latest/user/launch-instances.html>. Accedido: 2019-08-03.
- [37] Openstack. LVM. <https://docs.openstack.org/cinder/latest/configuration/block-storage/drivers/lvm-volume-driver.html>. Accedido: 2019-06-20.
- [38] Openstack. Manage flavors. <https://docs.openstack.org/horizon/latest/admin/manage-flavors.html>. Accedido: 2019-08-03.
- [39] Openstack. Manage projects and users. <https://docs.openstack.org/horizon/latest/admin/manage-projects-and-users.html>. Accedido: 2019-08-03.
- [40] Openstack. Memcached. <https://docs.openstack.org/mitaka/install-guide-ubuntu/environment-memcached.html>. Accedido: 2019-06-20.
- [41] Openstack. Message queuing. <https://docs.openstack.org/security-guide/messaging.html>. Accedido: 2019-06-20.
- [42] Openstack. Network architectures. <https://docs.openstack.org/openstack-ansible/stein/user/network-arch/example.html>. Accedido: 2019-07-05.
- [43] Openstack. Networking concepts. <https://docs.openstack.org/arch-design/design-networking/design-networking-concepts.html>. Accedido: 2019-06-15.
- [44] Openstack. Nova System Architecture. <https://docs.openstack.org/nova/latest/user/architecture.html>. Accedido: 2019-07-05.
- [45] Openstack. Object Storage API overview. https://docs.openstack.org/swift/latest/api/object_api_v1_overview.html. Accedido: 2019-07-05.
- [46] Openstack. Object Storage characteristics. <https://docs.openstack.org/swift/latest/admin/objectstorage-characteristics.html>. Accedido: 2019-07-05.
- [47] Openstack. OpenStack-Ansible Documentation. <https://docs.openstack.org/openstack-ansible/latest/>. Accedido: 2019-07-05.
- [48] Openstack. openstack_user_config settings reference. <https://docs.openstack.org/openstack-ansible/queens/reference/inventory/openstack-user-config-reference.html>. Accedido: 2019-06-20.
- [49] Openstack. Settings Reference. <https://docs.openstack.org/horizon/queens/configuration/settings.html>. Accedido: 2019-08-04.
- [50] Openstack. Software. <https://www.openstack.org/software/>. Accedido: 2019-06-15.
- [51] Openstack. Swift Architectural Overview. https://docs.openstack.org/swift/latest/overview_architecture.html. Accedido: 2019-07-05.
- [52] Openstack. The OpenStack Marketplace. <https://www.openstack.org/marketplace/distros/>. Accedido: 2019-07-05.
- [53] Openstack. Useful image properties. <https://docs.openstack.org/glance/latest/admin/useful-image-properties.html>. Accedido: 2019-06-20.
- [54] Openstack. Volume drivers. <https://docs.openstack.org/cinder/latest/configuration/block-storage/volume-drivers.html>. Accedido: 2019-06-20.

- [55] Openstack. Welcome to the Puppet OpenStack Guide! <https://docs.openstack.org/puppet-openstack-guide/latest/>. Accedido: 2019-07-05.
- [56] Oracle. What are Hypervisors? https://docs.oracle.com/cd/E50245_01/E50249/html/vmcon-hypervisor.html. Accedido: 2019-06-20.
- [57] Ken Pepple. *Deploying OpenStack*, chapter 4 Understanding Nova. O'Reilly Media, Inc., 2011.
- [58] Red Hat. Conductor. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/4/html/Configuration_Reference_Guide/section-conductor.html. Accedido: 2019-06-20.
- [59] Red Hat. Openstack Block Storage (Cinder). https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/6/html/Component_Overview/section-blockStorage.html. Accedido: 2019-06-20.
- [60] Red Hat. What is KVM? <https://www.redhat.com/en/topics/virtualization/what-is-KVM>. Accedido: 2019-06-20.
- [61] Red Hat. What is virtualization? <https://www.redhat.com/en/topics/virtualization/what-is-virtualization>. Accedido: 2019-06-20.
- [62] Red Hat. What's a Linux container. <https://www.redhat.com/en/topics/containers/whats-a-linux-container>. Accedido: 2019-06-20.
- [63] Ben Silverman and Michael Solberg. *Openstack for Architects*, chapter 3 Planning for Failure and Success, page 37. Packt Publishing, 2nd edition, 2018.
- [64] Murugiah Souppaya, John Morello, and Karen Scarfone. SP 800-190, Application Container Security Guide. Special Publication, National Institute of Standards & Technology, 2017.
- [65] VMware. Virtualization. <https://www.vmware.com/solutions/virtualization.html>. Accedido: 2019-06-20.
- [66] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1), May 2010.