

Implementación de Infraestructura de Nube privada: Un enfoque centrado en la seguridad y alta disponibilidad

UOC

Alejandro Mauricio Mellado Gatica

Máster en Ciberseguridad y Privacidad

Nombre Tutor/a de TF

Jordi Guijarro Olivares

Profesor/a responsable de la asignatura

Josep Jorba Esteve

Fecha Entrega

4 de Junio 2024

Universitat Oberta de Catalunya

GNU Free Documentation License (GNU FDL)

Copyright © 2024 Alejandro Mellado.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "GNU Free Documentation License".

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Servicios en nube privada bajo la perspectiva de la seguridad</i>
Nombre del autor:	<i>Alejandro Mellado Gatica</i>
Nombre del consultor/a:	<i>Jordi Guijarro Olivares</i>
Nombre del PRA:	<i>Josep Jorba Esteve</i>
Fecha de entrega (mm/aaaa):	<i>06/2024</i>
Titulación o programa:	<i>Máster en Ciberseguridad y Privacidad</i>
Área del Trabajo Final:	<i>Seguridad en cloud computing</i>
Idioma del trabajo:	<i>castellano</i>
Palabras clave	<i>Disponibilidad, Seguridad e Infraestructura</i>

Resumen del Trabajo

El presente documento da cuenta del desarrollo de una plataforma de infraestructura de servicios en nube privada, considerando elementos como la sostenibilidad y la escalabilidad económica, estableciendo de manera transversal la seguridad en cada uno de sus componentes o capas, destacando además el concepto de alta disponibilidad como eje central del proyecto. Con solo 3 meses para el desarrollo, se optó por una metodología ágil. Se desplegaron 6 servidores con Ubuntu 22.04 utilizando Canonical MaaS. Sobre estos servidores, se implementó una capa IaaS en nube privada con OpenNebula, usando como almacenamiento compartido un clúster Ceph. La capa de PaaS fue implementada con un clúster Incus y, sobre este, en contenedores Debian, un servicio Moodle HA que fue utilizado como prueba de concepto. Si bien el diseño fue una fase inicial referencial y de utilidad, la implementación en cada una de sus fases ajustó el diseño para hacer posibles los objetivos. El rendimiento y la estructura funcional de Moodle en esta plataforma presentaron problemas que debieron ser superados, requiriendo ajustes en su configuración durante el proceso de pruebas. La fase de pruebas y resultados se focalizó en la funcionalidad, el rendimiento, la alta disponibilidad y la seguridad de toda la infraestructura, validando así la arquitectura propuesta. La extensión del trabajo se refleja en las conclusiones, donde las tecnologías libres y/o abiertas permiten contar con una opción de implementación de infraestructura de nube privada, que pueden fortalecer el uso de esta tecnología, considerando aspectos de sostenibilidad con una mirada ético-social.

Abstract

The present document describes the development of a private cloud infrastructure services platform, considering elements such as sustainability and economic scalability, and establishing security across each of its components or layers. It also highlights the concept of high availability as the central axis of the project. With only 3 months for development, an agile methodology was chosen. Six servers with Ubuntu 22.04 were deployed using Canonical MaaS. On these servers, a private cloud IaaS layer was implemented with OpenNebula, using a Ceph cluster as shared storage. The PaaS layer was implemented with an Incus cluster and, on top of it, in Debian containers, a HA Moodle service was used as a proof of concept. Although the design was initially a referential and useful phase, the implementation in each of its stages adjusted the design to achieve the objectives. The performance and functional structure of Moodle on this platform presented problems that needed to be overcome, requiring adjustments in its configuration during the testing process. The testing and results phase focused on the functionality, performance, high availability, and security of the entire infrastructure, thus validating the proposed architecture. The scope of the work is reflected in the conclusions, where open source and/or free technologies provide an option for implementing a private cloud infrastructure services platform that can strengthen the use of technology considering aspects of sustainability from an ethical and social perspective.

Índice

1. Introducción.....	1
1.1. Contexto y justificación del Trabajo.....	2
1.2. Objetivos del Trabajo.....	3
1.3. Impacto en sostenibilidad, ético-social y de diversidad.....	3
1.4. Enfoque y método seguido.....	3
1.5. Planificación del Trabajo.....	4
1.6. Breve sumario de productos obtenidos.....	5
1.7. Breve descripción de los otros capítulos de la memoria.....	5
1.8 Estado del arte.....	5
1.9 Alcance del proyecto.....	6
2. Materiales y métodos.....	7
2.1. Análisis de plataformas.....	7
2.2. Diseño de arquitectura.....	9
2.3 Implementación, capa de Infraestructura como Servicio.....	14
2.4 Implementación, capa de Plataforma como Servicio.....	31
3. La seguridad en la infraestructura instalada.....	38
4. Pruebas y Resultados.....	45
5. Conclusiones y trabajos futuros.....	55
6. Glosario.....	58
7. Bibliografía.....	61
8. Anexos.....	64

Lista de figuras

Figura 1: Arquitectura de hardware y conectividad en la capa IaaS.....	10
Figura 2: Arquitectura genérica de plataforma.....	11
Figura 3: Arquitectura genérica para Moodle como servicio.....	12
Figura 4: Arquitectura de contenedores con redes aisladas.....	13
Figura 5: Arquitectura del Clúster OpenNebula en producción.....	14
Figura 6: Estructura de implementación de Moodle en alta disponibilidad.....	32
Figura 7: Directorios Moodle.....	35
Figura 8: Esquema de red por capas de aislamiento.....	37
Figura 9: Arquitectura de redes aisladas.....	41
Figura 10: Acceso a la GUI de Syncthing desde una red autorizada.....	42

Lista de imágenes

Imagen 1: Carta Gantt del Proyecto.....	4
Imagen 2: Zona 0 del Cluster OpenNebula.....	22
Imagen 3: Hook, hosts y máquinas virtuales.....	30
Imagen 4: Caché en RAM del contenedor nodo2.....	34
Imagen 5: Reglas de filtrado PfSense, acceso desde LAN1.....	39
Imagen 6: Regla de filtrado, acceso desde otras interfaces.....	40
Imagen 7: Vista de OpenNebula en producción.....	46
Imagen 8: Vista de la lista de contenedores.....	47
Imagen 9: Vista de dos cursos restaurados.....	47
Imagen 10: Resultados del benchmark realizado en Moodle.....	48
Imagen 11: VIP asumidas por keepalived en 'lvs2'.....	49
Imagen 12: Lista de máquinas virtuales LVS y acceso a Moodle.....	49
Imagen 13: Lista de contenedores del Clúster Incus.....	50
Imagen 14: Estado de las máquinas virtuales y acceso a Moodle.....	50
Imagen 15: Advertencia de salud de Ceph con un nodo apagado.....	51
Imagen 16: Contenedores del Clúster Incus.....	51
Imagen 17: Hosts y máquinas virtuales, respuesta del servicio Moodle.....	51
Imagen 18: Secuencia de comando de exploración de IPs y puertos.....	53
Imagen 19: Exploración desde la red de administradores.....	53
Imagen 20: Exploración de puertos de moo.inf.uct.cl desde Internet.....	54
Imagen 21: Vista del Storage estándar en Canonical Maas (Anexo A).....	64
Imagen 22: Configuración de red estándar en Canonical Maas (Anexo A).....	65
Imagen 23: Vista de Sunstone en un navegador Web (Anexo D).....	71

1. Introducción

En términos metafóricos, se podría decir que el primero de enero de 1970 a las cero horas comienza el Big Bang de la era de la información. Sus inicios fueron marcadamente precarios; los computadores eran prácticamente calculadoras o máquinas de escribir avanzadas y ocasionalmente conectadas a una red experimental en desarrollo. Veinte años más tarde, se liberaron las restricciones de la conectividad experimental, masificando de manera explosiva el uso de una red formalmente llamada Internet. La informática de esos años 90 del siglo XX estuvo marcada por muchas innovaciones; el desarrollo del World Wide Web por Tim Berners-Lee facilitó el acceso a recursos de información distribuidos, que junto a las listas de correo de USENET y servicios como correo electrónico, FTP y Telnet, pavimentaron los flujos de datos de una estructura física de red y un trabajo colaborativo de muchos sectores de la tecnología (científicos, ingenieros, centros educativos, empresas, etc.). Desde el punto de vista del desarrollo evolutivo, la principal característica de Internet y toda su tecnología derivada, fue y es que se mejora a sí misma. El siglo XXI sin duda generó que la informática y el uso de las tecnologías de información se transformaran en una herramienta transversal a todas las áreas del quehacer humano, desde lo productivo a lo educativo, en la medicina y el sector público, incluso en el entretenimiento. En la actualidad, toda actividad humana que involucre almacenamiento, procesamiento y transmisión de datos, depende de las tecnologías de información. En consecuencia, todos los comportamientos humanos presentes en la sociedad están presentes en Internet y en el uso de las tecnologías digitales. En sus inicios, la seguridad de los sistemas digitales se enfocaba básicamente en listas de acceso o credenciales de ingreso; en la actualidad, el campo de acción es mucho más complejo, acuñando el nombre de ciberseguridad. Así como la informática o las tecnologías de información son transversales a distintas actividades humanas, la ciberseguridad es transversal a distintos componentes, usos y políticas de la informática. Por tanto, toda implementación informática o de sistemas informáticos conlleva, explícita o implícitamente, elementos de seguridad o ciberseguridad que los resguarden. Desde las primeras generaciones de máquinas de Tanenbaum hasta los grandes centros de datos de hoy, la base de toda implementación es la infraestructura distribuida en componente de hardware y software, hoy segmentada en capas por el Metal como Servicio (MaaS), la Infraestructura como Servicio (IaaS), la Plataforma como Servicio (PaaS) y el Software como Servicio (SaaS). Cada una de estas componentes necesita las correspondientes salvaguardas que permitan niveles de seguridad.

Si bien la capa SaaS es el producto final, es decir, lo que está disponible para los usuarios, las salvaguardas mencionadas anteriormente dependen directamente de la configuración de las capas subyacentes. En este punto, cobran relevancia las configuraciones de seguridad en la infraestructura de servicio en nube, tanto a nivel IaaS como en la capa PaaS.

Esta actividad documentada se centra en la implementación de una infraestructura de servicios en nube privada (on-premise) que resguarde la Confiabilidad, Integridad y Disponibilidad en las capas IaaS y PaaS, con una arquitectura de requisitos mínimos viables y bajo la implementación de un caso de servicio destinado a la producción.

1.1. Contexto y justificación del Trabajo

La concentración del mercado de los servicios de infraestructura en la nube en unos pocos actores globales puede limitar la competencia, resultando a menudo en menos opciones y precios potencialmente más altos para los consumidores, empresas y organizaciones. Esta concentración también puede frenar la innovación y limitar la capacidad local de los países para desarrollar y controlar sus propias infraestructuras tecnológicas, representando un desafío significativo para la soberanía tecnológica y el desarrollo tecnológico sostenible, tanto en lo económico como en lo ambiental.

Una plataforma de servicios en la nube basada en software libre es una estrategia clave para implementar nubes privadas o públicas geo-localizadas en territorio nacional. Esto permite a las organizaciones y naciones tener plataformas más autónomas, diversificando el mercado y fomentando la innovación local. Además, administrar sus estándares de ciberseguridad mejora la protección de datos e infraestructura. El software libre y de código abierto promueve la colaboración y el intercambio de conocimientos, esenciales para el avance tecnológico sostenible.

En el ámbito de Cloud Computing, AWS es uno de los actores más prominentes, junto con competidores destacados como Google, Microsoft y Oracle, que son algunas de las empresas más dominantes del mercado. Estos servicios se conocen en la literatura técnica como servicios de nube pública. La popularidad de estos servicios en la nube los ha posicionado en el mercado como un negocio muy rentable a nivel global, concentrando prácticamente el almacenamiento, procesamiento y transmisión de información de gobiernos, empresas y usuarios.

Para romper la concentración de mercado mencionada anteriormente, Arthur (1996) sugiere la aplicación de subsidios e intervención en los mercados. Según él, estas medidas pueden fomentar la diversidad y evitar que pocas empresas dominen completamente el mercado, limitando así la innovación y la competencia [2].

La Unión Europea aprobó en diciembre de 2023 el proyecto PIICE sobre tecnologías de computación en la nube y en el borde, con una inversión de 1.200 millones de euros para apoyar la investigación, el desarrollo y la utilización industrial de estas tecnologías en Europa, incluyendo a España como miembro participante. PIICE es parte de los esfuerzos de la Comisión Europea por una economía digital más verde, segura, resiliente y soberana. Según el Comisario Thierry Breton, "El PIICE es crucial para lograr una innovación puntera en tecnologías de computación en la nube y en el borde que cumplan los requisitos europeos de interoperabilidad, protección de datos, sostenibilidad y ciberseguridad" [3].

Si bien se percibe una tendencia mundial en la que los procesos de transformación digital se asocian a la nube, por regla general esta asociación se establece en referencia a la nube pública. De hecho, el PIICE podría percibirse como un apoyo a la implementación de servicios en la nube pública en territorio europeo. No obstante, para lograr esa computación en el borde, podría ser necesario que aquellas empresas u organizaciones con capacidad para contar con sus propios centros de datos (datacenters) puedan implementar sus servicios de nube privada (on-premise).

Por todo lo anterior, avanzar en la implementación de infraestructura de servicios en nube con niveles de garantía en seguridad, buscando además la mejor configuración económicamente sostenible, puede ser un eje clave en el desarrollo tecnológico local, tanto para países desarrollados como en vías de desarrollo.

1.2. Objetivos del Trabajo

Objetivo General

Realizar la implementación de una infraestructura de servicios en nube con niveles de seguridad, económicamente escalable y sostenible, documentando la experiencia para que sea implementada en cualquier empresa u organización.

Objetivos Parciales

- Analizar las tecnologías que provee el software libre o abierto para la implementación de infraestructura de servicios en nube privada.
- Realizar la implementación de la tecnología de servicios en nube seleccionada buscando el mínimo funcional para la producción.
- Configurar la implementación de los servicios considerando los aspectos que proveen una plataforma segura.
- Realizar una prueba de concepto que verifique la viabilidad de solución planteada.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

En este trabajo final de máster, es observable un impacto ético, sustentado en la elección de plataformas de software libre o abierto, donde se promueve la interoperatividad y se fomenta el uso equitativo de las tecnologías. Esto asegura que los beneficios de la transformación digital estén disponibles para un mayor número de usuarios y comunidades. En el aspecto social, se observa que la infraestructura de servicios en nube, ya sea privada o pública geo-localizada localmente, puede potenciar la innovación y el desarrollo en diversos sectores, beneficiando a la sociedad en general. Esto podría traducirse en la mejora de servicios, el fomento de la educación, el conocimiento y el apoyo a la inclusión digital. En cuanto al plano ambiental, aunque el enfoque principal del proyecto no es la eficiencia energética, su implementación se alinea tangencialmente con los objetivos de eficiencia energética y sostenibilidad del PIICE de la Comisión Europea. Este enfoque indirecto contribuye a minimizar el impacto ambiental de las tecnologías de información, buscando reducir la huella de carbono y promover un uso más consciente de los recursos tecnológicos.

1.4. Enfoque y método seguido

De acuerdo con los tiempos y las particularidades del proyecto, se estima que la metodología debería ser ágil. Por tanto, se propone usar Kanban[4] combinado con principios de gestión de proyectos ágiles[5]. Kanban facilitará la adaptación del flujo de trabajo mediante un tablero que visualizará cada tarea a realizar, divididas en tres categorías: 'Por hacer', 'En progreso' y 'Completado'. Esto permitirá priorizar las distintas tareas, enfocándose en las que están en curso y visualizando el progreso del

proyecto. Como complemento a Kanban, la gestión de proyectos ágiles implicará establecer microtareas para alcanzar cada objetivo específico, con una evaluación y adaptación de enfoque al final de cada ciclo para asegurar la alineación con los objetivos del proyecto.

1.5. Planificación del Trabajo

Tareas a realizar

Evaluación de plataformas: En primer lugar, se llevará a cabo una revisión de la documentación teórica de OpenStack, OpenNebula y Apache CloudStack para entender sus características, ventajas y desventajas. Posteriormente, se implementarán prototipos virtualizados para cada tecnología, lo que permitirá evaluar la curva de aprendizaje, los requerimientos mínimos de hardware y la calidad del dashboard.

Selección de la plataforma: Con base en la evaluación anterior, se elegirá la plataforma que mejor se ajuste a los requerimientos esperados.

Diseño de arquitectura: Se procederá a diseñar la arquitectura de la infraestructura en la nube, considerando aspectos como escalabilidad, seguridad, y la integración con otros sistemas. Este diseño debe incluir la distribución de recursos, la planificación de redes y la estrategia de almacenamiento.

Implementación y configuración: Se configurará la infraestructura en modo clúster para asegurar la alta disponibilidad. Además, se buscará una integración que sea económica y eficiente en términos de recursos de hardware.

Configuración de seguridad: Se establecerán niveles de seguridad adecuados, incluyendo la configuración de firewalls, políticas de acceso y aislamiento de recursos. También se implementarán mecanismos de monitoreo y alerta para detectar y responder a incidentes de seguridad.

Validación mediante una prueba de concepto: Se desarrollará una prueba de concepto (PoC) que demuestre la funcionalidad y seguridad de la plataforma elegida. Esta prueba debe simular escenarios reales de uso y evaluar la respuesta de la plataforma ante diversas situaciones.

Documentación: Se documentará la configuración, la arquitectura y los procedimientos operativos. Esta tarea será continua durante todo el desarrollo del proyecto.

Carta Gantt

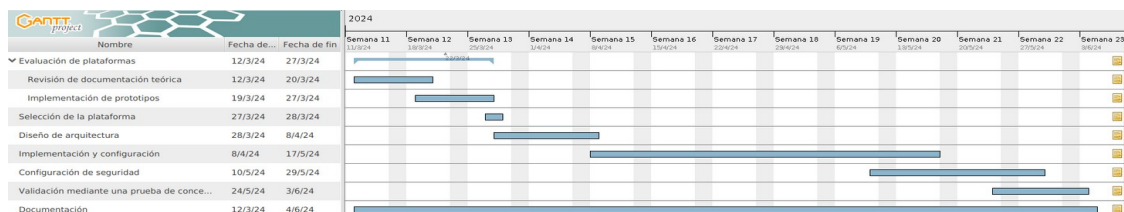


Imagen 1: Carta Gantt del Proyecto

1.6. Breve resumen de productos obtenidos

El producto obtenido es una plataforma de servicios en nube segura, funcional y de alta disponibilidad que actualmente se encuentra en producción. La arquitectura de esta plataforma ha sido debidamente documentada, permitiendo ser replicada en cualquier organización que disponga de recursos similares a los descritos. Este diseño asegura que la plataforma no solo cumpla con los requisitos operativos y de seguridad, sino que también sea accesible y adaptable para otras entidades.

1.7. Breve descripción de los otros capítulos de la memoria

El siguiente capítulo de materiales y métodos considera, como primera sección, el análisis de plataformas. Esta sección es fundamental para el proyecto, ya que establecerá la base de los componentes de software para la implementación de la infraestructura de servicios en la nube. Dado que el tiempo de desarrollo es limitado, las distintas opciones pueden ser determinantes para lograr los objetivos planteados. La segunda sección es el Diseño de Arquitectura, que servirá como guía para los distintos componentes técnicos que conformarán la infraestructura. La tercera sección se centra en la implementación de la capa IaaS, una plataforma que permitirá el despliegue de máquinas virtuales como infraestructura base de alta disponibilidad. La cuarta sección trata sobre la capa PaaS, donde se implementará un clúster de contenedores. Posteriormente, se desplegará un servicio Moodle de alta disponibilidad en múltiples instancias, ejecutadas en contenedores con sistema operativo Debian 12. El siguiente capítulo aborda la seguridad en la infraestructura de nube instalada, tratando la seguridad en términos de confidencialidad, integridad y disponibilidad de toda la infraestructura, en línea con los objetivos del proyecto. El capítulo que sigue se enfoca en las pruebas y resultados, abarcando aspectos como la funcionalidad, el rendimiento, la alta disponibilidad y la seguridad, que son elementos clave para contar con una plataforma de servicios en la nube altamente disponible y escalable. El último capítulo corresponde a las conclusiones, que permiten visualizar los aspectos clave del proyecto desde los puntos de vista técnico, operativo y ambiental.

1.8 Estado del arte

Para la implementación de infraestructura de servicios en nube, existen diversas opciones y plataformas enmarcadas bajo el concepto de Infraestructura como Servicio (IaaS). Las plataformas más desarrolladas son de código abierto, permitiendo a las organizaciones construir su propia infraestructura en la nube con flexibilidad y control no comunes en las soluciones de nube pública. Esto incluye la auto-atención para los usuarios, que pueden aprovisionar recursos, administrar cargas de trabajo y monitorear su uso sin intervención directa del personal de TI. La API abierta de estas plataformas permite la personalización de la interfaz de usuario (GUI) y facilita la integración con otras herramientas y sistemas. Esta personalización y flexibilidad son valiosas para las organizaciones con requisitos únicos o que desean mantener un control riguroso sobre su entorno de TI por razones de seguridad o política interna.

Entre las plataformas IaaS más usadas y conocidas se encuentran: OpenStack, ApacheCloud, OpenNebula, CloudStack y Eucalyptus. De estas, la tres primeras parecen ser las que poseen un mayor grado de desarrollo y documentación para su implementación. A continuación un resumen y/o descripción de cada plataforma:

OpenStack es un sistema operativo en la nube que controla grandes grupos de recursos informáticos, de almacenamiento y de redes en todo el centro de datos, todos administrados y aprovisionados a través de API con mecanismos de autenticación comunes. También está disponible un panel que brinda control a los administradores y al mismo tiempo permite a sus usuarios aprovisionar recursos a través de una interfaz web. Más allá de la funcionalidad estándar de infraestructura como servicio, los componentes adicionales brindan orquestación, gestión de fallas y gestión de servicios, entre otros servicios, para garantizar una alta disponibilidad de las aplicaciones de usuario [6].

OpenNebula es una plataforma de código abierto potente pero fácil de usar para la infraestructura de nube privada, híbrida o perimetral de su empresa. OpenNebula unifica la simplicidad y agilidad de la nube pública con el rendimiento, la seguridad y el control de la nube privada; y brinda flexibilidad, escalabilidad, simplicidad e independencia del proveedor para respaldar las crecientes necesidades de sus desarrolladores y prácticas de DevOps[7].

Apache CloudStack es un software de código abierto diseñado para implementar y administrar grandes redes de máquinas virtuales como una plataforma de computación en la nube IaaS, altamente disponible y escalable. Es utilizado por proveedores de servicios para ofrecer servicios de nube pública y por empresas para proporcionar nubes privadas o híbridas. CloudStack incluye una pila completa de características deseadas en una nube IaaS: orquestación informática, red como servicio, administración de cuentas y usuarios, una API nativa abierta y una interfaz de usuario (UI) de primera clase. Los usuarios pueden administrar su nube con una interfaz web, herramientas de línea de comandos y/o una API RESTful [8].

Las tecnologías descritas, además de implementaciones de nube privada, también se utilizan en nubes públicas, siendo una oportunidad para la innovación y el desarrollo alineado con el enfoque de la comisión europea y su proyecto PIICE.

1.9 Alcance del proyecto

La complejidad y el amplio espectro de opciones en la estructuración y despliegue de una infraestructura de servicios en nube exigen una delimitación precisa del alcance de este proyecto. El objetivo principal es implementar una infraestructura de servicios en nube segura, escalable y sostenible. Este proyecto se centrará en desarrollar una solución mínimamente viable que permita a las organizaciones evaluar y, potencialmente, adoptar la arquitectura sugerida.

Dado que el enfoque es la ciberseguridad, es crucial que los requisitos de la solución estén alineados, directa o indirectamente, con estrategias y procedimientos destinados a garantizar la confidencialidad, integridad y disponibilidad de los recursos digitales. Esto es fundamental para proteger el procesamiento, almacenamiento y transmisión de datos.

En este contexto, se considera que la prueba de concepto debe abarcar hasta el nivel de aplicación (SaaS). Por ende, se propone la ejecución de Moodle como servicio sobre la infraestructura de nube propuesta, realizando las evaluaciones pertinentes para confirmar que se alcanzan los niveles de seguridad y continuidad del servicio esperados.

2. Materiales y métodos

2.1. Análisis de plataformas

En el mundo de la infraestructura como servicio (IaaS), existen distintas perspectivas o enfoques. En esta evaluación se ha optado por evaluar OpenStack y OpenNebula, descartando Apache CloudStack por limitaciones de tiempo y alcance. De este modo, la evaluación se realizó siguiendo dos cursos de acción:

1. Mediante la búsqueda de documentación que contenga comparativas de estas plataformas.
2. Intentando el despliegue simulado de cada tecnología.

Documentación técnica encontrada

Principales características

OpenStack y OpenNebula comparten características fundamentales como plataformas de IaaS. Ambas permiten a los usuarios desplegar y administrar máquinas virtuales, redes y almacenamiento en un entorno de nube. Al ser de código abierto, estas plataformas permiten la colaboración y la personalización según las necesidades específicas de cada usuario. Además, proporcionan APIs para la automatización e integración con otras herramientas y sistemas, optimizando la gestión de recursos. Estas plataformas soportan la multitenencia, permitiendo a diversos usuarios o departamentos operar de forma aislada dentro de la misma infraestructura física. Asimismo, son compatibles con varios hipervisores, brindando la flexibilidad de elegir la tecnología de virtualización que mejor se ajuste a sus requerimientos [9][10].

Estas tecnologías también poseen diferencias en cuanto a enfoque y usabilidad. OpenNebula se centra en la simplicidad y es conocida por su facilidad de despliegue y gestión, lo que la hace atractiva para entornos que valoran la eficiencia. En contraste, OpenStack ofrece una gama más amplia de servicios y funcionalidades, siendo ideal para entornos más grandes y complejos que requieren una mayor escalabilidad. OpenStack tiene una arquitectura modular que puede escalar para manejar datacenters de gran tamaño, mientras que OpenNebula ofrece una arquitectura más ligera, adecuada para diferentes escalas, incluyendo pequeñas y medianas empresas. OpenStack tiene una de las mayores comunidades y bases de usuarios entre las plataformas de IaaS, lo que se traduce en una amplia gama de documentación, soporte y recursos. OpenNebula cuenta con una comunidad activa, aunque más pequeña en comparación con OpenStack, y es a menudo preferida por organizaciones que buscan soluciones más ágiles y menos complejas [9][10][11].

Requisitos de hardware

Existe escasa información para poder dimensionar los requisitos de hardware para la implementación de un clúster HA para infraestructura de servicios en nube en cada una de las tecnologías analizadas. Sin embargo, la documentación de OpenNebula sugiere como mínimo 3 nodos sin especificar las características de cada servidor [12]. En el caso de OpenStack, se encontraron referencias a los requisitos para un despliegue de un clúster HA OpenStack con Canonical Juju, cuya configuración requiere: Un nodo servidor para Canonical MaaS, un nodo servidor para Canonical

Juju y 4 nodos servidores con 3 discos cada uno para distribuir los distintos módulos o servicios [13]. En este caso, se concluye que dada la estructura modular de OpenStack con múltiples servicios integrados, no sería viable usar una estructura con 3 nodos.

Curva de aprendizaje

Si bien las características de cada una de las tecnologías de nube dan una visión de sus potencialidad en la implementación de servicios, un factor que puede afectar el objetivo del proyecto es la curva de aprendizaje.

Comúnmente, la documentación disponible indica que la implementación de OpenStack como plataforma de servicios en nube representa un desafío considerable. La modularidad de OpenStack se percibe como compleja durante el despliegue, resultando en una curva de aprendizaje pronunciada, especialmente para quienes se enfrentan a su estructura por primera vez. Además, su gestión y mantenimiento pueden demandar habilidades avanzadas en línea de comandos y un profundo entendimiento de sus múltiples componentes y la interacción entre ellos [12].

Respecto a OpenNebula, se estima que su curva de aprendizaje es menos pronunciada que la de OpenStack. Orientada hacia la simplicidad y la facilidad de uso, OpenNebula brinda una interfaz amigable que permite a usuarios, incluso aquellos sin una formación técnica profunda, gestionar eficientemente los recursos en la nube. Esta filosofía se manifiesta en la gestión de operaciones comunes, como la recuperación ante fallos, que en OpenNebula tiende a ser más automatizada en comparación con la necesidad de OpenStack de integrar automatizaciones externas o realizar intervenciones manuales para tareas similares [15][11].

Simulaciones y prueba de infraestructura

Según la documentación consultada, es factible probar las plataformas de infraestructura, como OpenStack y OpenNebula, en un entorno de virtualización anidada, aunque de manera mínima. Específicamente para OpenStack, existen diversas opciones de instalación en entornos virtualizados. Sin embargo, simular un entorno de alta disponibilidad exige una cantidad considerable de recursos para obtener resultados satisfactorios. Debido a esta limitación, se optó por realizar pruebas de funcionalidades a nivel de nube en microStack, utilizando un único nodo de prueba. Aunque esta aproximación no permite evaluar la alta disponibilidad, facilita la exploración de las funcionalidades completas de la nube, que se espera sean similares en un ambiente de clúster, a pesar de la imposibilidad de probar la resiliencia de una configuración de alta disponibilidad.

Utilizando OpenNebula, se creó un entorno de simulación con tres nodos, empleando el hipervisor KVM y la asistencia de Qemu. En un período de no más de dos semanas, se lograron avances significativos, quedando pendiente únicamente el inicio automático de una máquina virtual en caso de fallo de algún hipervisor. La arquitectura simulada incorporó Ceph como solución de almacenamiento, un clúster de bases de datos Galera MariaDB y la interfaz SunStone en cada uno de los nodos virtualizados. Además, se implementó una IP flotante para garantizar la continuidad del servicio ante la caída de un nodo. Durante esta simulación, se constató la curva de aprendizaje de OpenNebula, evidenciando sus características mencionadas anteriormente. La flexibilidad, simplicidad y facilidad de operación de OpenNebula, junto con los requisitos de hardware para su implementación, confirman que esta plataforma es una excelente alternativa para alcanzar los objetivos de este proyecto.

2.2. Diseño de arquitectura

De acuerdo a los objetivos planteados y el alcance del proyecto, el diseño de arquitectura se debería enfocar en los distintos aspectos destinados a desplegar una infraestructura de servicios en nube que considere como base la confidencialidad, integridad y disponibilidad de los servicios. Observando que dada la funcionalidad de los servicios en nube, las distintas alternativas de software de infraestructura subyacente que soportan los servicios IaaS y PaaS, ofrecen flexibilidad en sus configuración para que sea el administrador o arquitecto quien estructure la arquitectura que provea la mejor disponibilidad posible, siendo la tecnología de software subyacente la que contiene mecanismos de confidencialidad. No obstante, no es posible asegurar la Integridad ya que la capa SaaS depende de los usuarios desarrolladores y escapa por tanto del control de los administradores de las capas inferiores (IaaS y PaaS).

Dado el análisis realizado en el apartado anterior, se ha determinado que OpenNebula sería la tecnología más adecuada para esta implementación. El diseño de arquitectura será basado en esta tecnología.

En las distintas pruebas de prototipo realizadas a nivel de virtualización, se encontró que la recuperación de fallos implica un tiempo que puede llegar a ser significativamente perceptible por los usuarios. Este tiempo se debe a que la máquina que es virtualizada cuenta con sistema operativo completo, en el momento en que la infraestructura de nube detecta la pérdida de actividad de un hipervisor, se inicia la máquina virtual en algún nodo hipervisor activo. El tiempo por tanto dependerá de la carga de trabajo que posea el sistema operativo y para efectos de continuidad del servicio podría llegar a ser importante. Más allá de este tiempo, es posible hablar en este caso, de alta disponibilidad en la capa IaaS, ya que frente a un fallo la recuperación es automática y dependiendo del nivel de uso de los servicios, la falla en la capa SaaS podría llegar a ser imperceptible para el usuario. Por tanto, la arquitectura debe considerar como mínimo la disponibilidad del nivel IaaS, en donde una estructura mínima en cualquier arquitectura de tipo cluster es 3 nodos.

Contar en este contexto con 3 hipervisores no asegura la alta disponibilidad porque un gran problema es almacenamiento. Todo sistema de virtualización (tradicional o en nube), tiene como opción la migración de máquinas virtuales en vivo, siendo el almacenamiento común a todos los nodos, el requisito necesario para su implementación. De hecho, para iniciar una máquina virtual en un hipervisor activo, como se describe en el párrafo anterior requiere un repositorio de imágenes común accesible por todos los hipervisores que participan del clúster. Es por eso que la arquitectura propuesta considera un clúster de almacenamiento aprovechando los 3 nodos hipervisores.

La Figura 1 esquematiza un modelo de arquitectura mínimo, complementando la infraestructura de OpenNebula con un Clúster Ceph que ofrece un almacenamiento compartido necesario para alta disponibilidad [29]. Ceph como arquitectura de almacenamiento de alta disponibilidad requiere como mínimo 3 nodos con 2 OSD por nodos.

En general, a nivel de red el mínimo de interfaces requeridas es 2: una interfaz que se usa para tareas de gestión, que comunica internamente los nodos que conforman el clúster; y, una interfaz externa que es usada en muchos casos acceder a la red subyacente por medio de Open vSwitch, otorgando flexibilidad y autonomía al usuario que puede estructurar sus propias redes en el entorno virtual que la nube le provee. A

nivel físico estas interfaces se conectan a un Switch que idealmente debería ser de 10Gbps, sobre todo para mejorar el acoplamiento de los hipervisores con el almacén Ceph. No obstante, se estima que para una pequeña empresa podría ser de 1 Gbps. Cabe señalar que el switch debe contar con la posibilidad de usar VLAN, ya que es precisamente el etiquetado a nivel Ethernet el que otorga esta flexibilidad en el entorno de Open vSwitch.

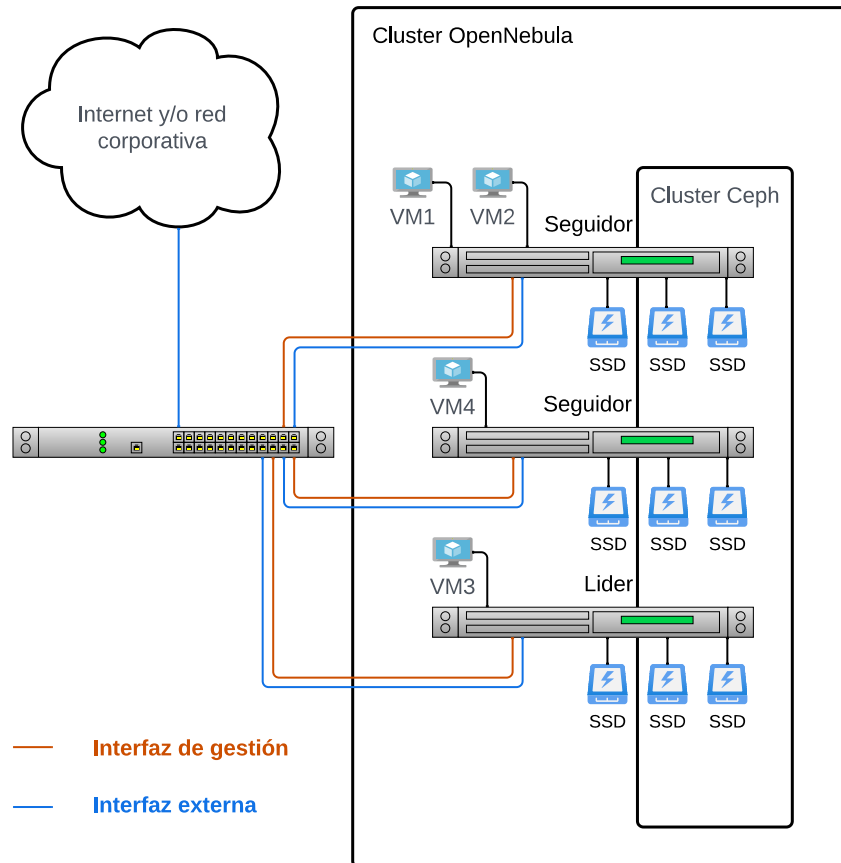


Figura 1: Arquitectura de hardware y conectividad en la capa IaaS.

Debido a la temporalidad frente la recuperación de fallos anteriormente descrita y a modo de mejorar la alta disponibilidad, se hace necesario que la arquitectura considere una segunda capa de plataforma. En este caso se observan varias alternativas en las que destacan dos:

1. Montar la plataforma de servicio en un clúster usando contenedores LXD con una estructura de gestión y escalamiento asistido.
2. Usar para la plataforma de servicio con un orquestador Kubernetes en donde el escalamiento puede ser automatizado.

Aunque estás dos alternativas están orientadas a mejorar la disponibilidad de los servicios en un alto grado. El uso de los contenedores LXD se acerca más al objetivo económicamente escalable y sostenible ya que está tecnología de contención es más simple de implementar y mantener. No obstante, considerando que OpenNebula como plataforma base posee un módulo de servicio de Kubernetes (OneKE), esta opción

podría considerarse en una segunda etapa de actualización en el caso de requerir escalamiento automatizado y en el caso que la empresa u organización en donde será desplegada puede asumir los costos de implementación y mantenimiento.

Las plataformas de servicios en la nube, como OpenNebula, ofrecen a los usuarios, incluidos los profesionales de DevOps, una amplia gama de opciones de configuración, permitiendo la creación de diversas estructuras de red y entornos de virtualización. En este marco, la disponibilidad y la seguridad de los servicios se verán influenciadas significativamente por la arquitectura desarrollada en la capa de plataforma (PaaS). Por lo tanto, la arquitectura que se propone en esta capa debe considerarse como un modelo de referencia inicial para pruebas, sujeto a optimizaciones basadas en las necesidades específicas del servicio que se va a implementar.

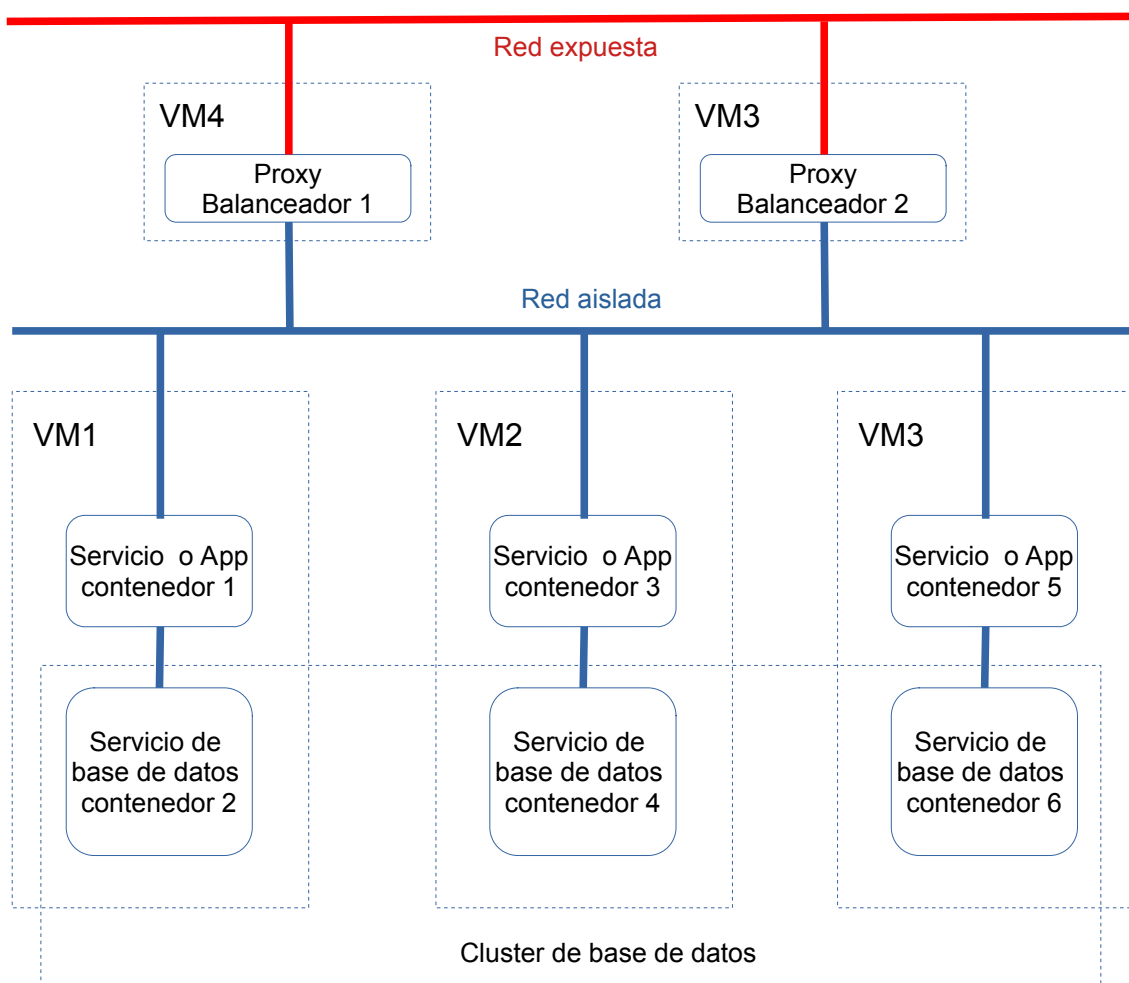


Figura 2: Arquitectura genérica de plataforma.

En general los diseños de arquitectura en la capa de plataforma poseen varios elementos comunes tendientes a ofrecer alto desempeño y disponibilidad, considerando también el aislamiento a nivel de red para minimizar los impactos a otros servicios frente a posibles vulnerabilidades a nivel de software.

La figura 2, propone un diseño genérico de arquitectura dividido en distintas capas:

1. Proxies y Balanceadores de carga: pueden estar implementadas en contenedores o máquinas virtuales.
2. Servicios: que pueden estar desplegados en uno o más contenedores.
3. Base de datos: que pueden estar desplegados en un o más contenedores o máquinas virtuales.
4. Capa de red aislada y capa de red expuesta.

Dado que el proyecto se enfoca en una prueba de concepto que alcanza el nivel SaaS, la arquitectura presentada en la Figura 2 sirve como un modelo de referencia. Este modelo debe ser personalizado según las características específicas del servicio de software que se va a implementar. En el caso de Moodle, este servicio se define como una plataforma de gestión de aprendizaje (LMS, por sus siglas en inglés), cuya interacción con los usuarios, principalmente docentes y estudiantes, se facilita a través de un sitio web. Los componentes esenciales para la implementación de Moodle incluyen un motor de base de datos, que comúnmente es MariaDB; un intérprete PHP; y un servidor web, que generalmente es Apache o Nginx. Además, requiere un directorio, usualmente denominado 'moodledata', que almacena archivos de usuario, materiales de los cursos, configuraciones de idioma y otros datos en caché. La configuración de Moodle en una plataforma para alta disponibilidad puede variar, pero es crucial que el directorio 'moodledata' sea accesible y común a todas las instancias de Moodle en ejecución.

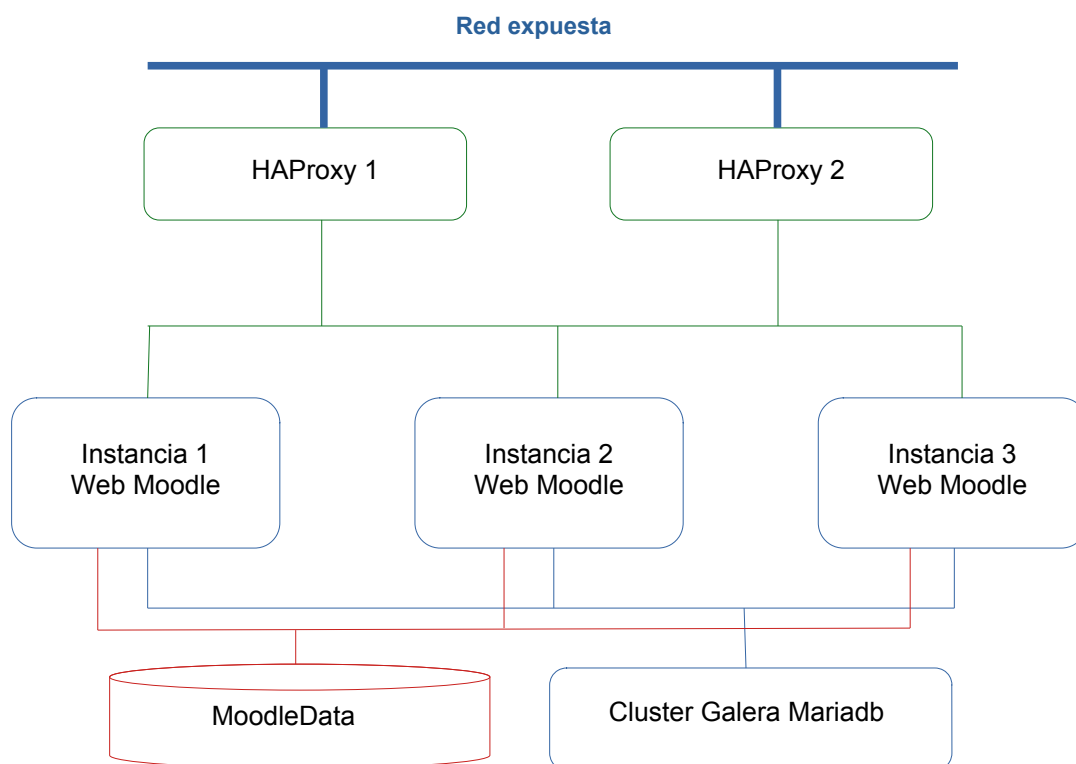


Figura 3: Arquitectura genérica para Moodle como servicio.

Existen diversas alternativas para diseñar una arquitectura de Moodle en alta disponibilidad. Para el presente proyecto se ha optado por una estructura que utiliza contenedores persistentes y efímeros. La propuesta incluye:

- Un clúster de base de datos **Galera MariaDB**, compuesto por **3 contenedores persistentes**, diseñados para garantizar la integridad y disponibilidad de los datos.
- Un clúster de caché **Redis Sentinel**, también integrado por **3 contenedores persistentes**, asegurando la alta disponibilidad y gestión eficiente de la caché.
- **3 instancias efímeras de Moodle**, cada una con **HAProxy**, **Nginx** y **PHP**, así como el repositorio raíz del ambiente de Moodle. Estas instancias también incluyen un montaje del directorio “moodledata”, el cual se configura sobre un espacio de almacenamiento compartido y persistente (CephFS), garantizando la continuidad de los datos críticos del sistema.

Aunque inicialmente se propone implementar tres instancias efímeras de Moodle, la arquitectura diseñada permite escalar a un mayor número de instancias según sea necesario. Esto es factible gracias al nivel de persistencia establecido en los contenedores de base de datos y almacenamiento, que aseguran la integridad y disponibilidad de los datos independientemente del número de instancias de la aplicación.

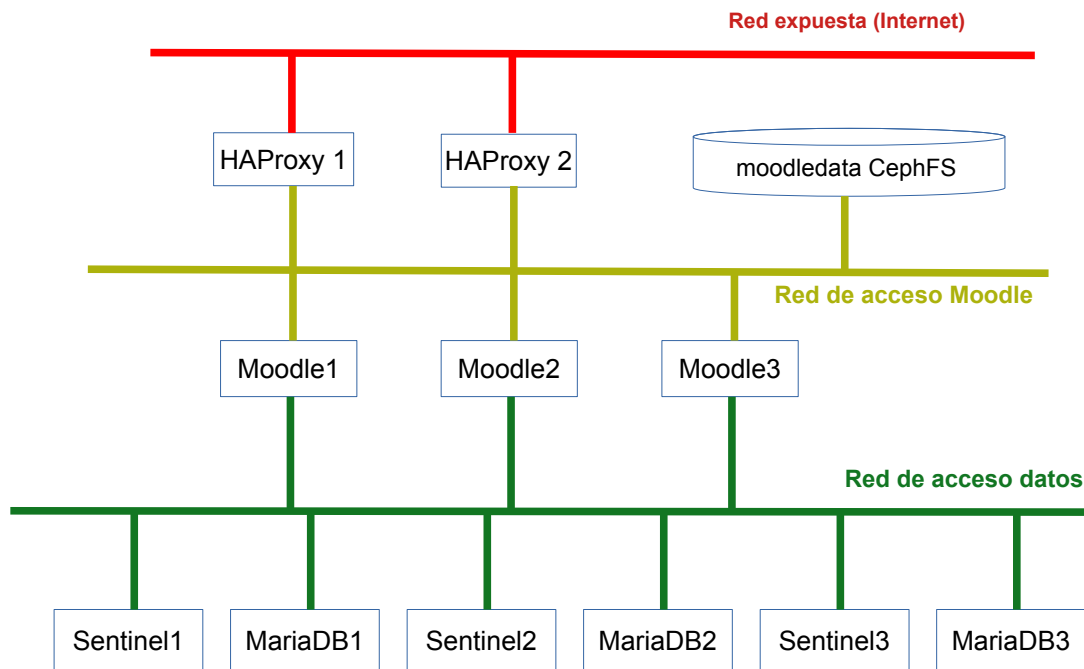


Figura 4: Arquitectura de contenedores con redes aisladas.

La Figura 4 ilustra una arquitectura de servicios estructurada por capas. En la **capa de datos**, se destaca la utilización de **Redis Sentinel** para los servicios de caché y **MariaDB (Galera)** para la gestión de bases de datos, ambos configurados para operar en una red aislada que garantiza una transferencia de datos segura y exclusiva entre contenedores y el almacenamiento de datos. La **segunda capa** consta de contenedores **Moodle** junto con **HAProxy**, los cuales acceden al directorio moodledata ubicado en un repositorio común dentro de un cluster de **Ceph**, accesible a través de un montaje **Ceph-fuse**. Finalmente, la **tercera capa** incluye una configuración de **HAProxy** dedicada a proveer y exponer el servicio Moodle a Internet.

2.3 Implementación, capa de Infraestructura como Servicio

Si bien en el proceso de simulación se realizó con tres nodos como el mínimo viable. En el caso de implementación para producción se cuenta con 6 servidores físicos. De estos, 4 tienen capacidades para funcionar como nodos de almacenamiento y 3 están configurados para actuar como hipervisores KVM. Además, 3 servidores están dedicados a gestionar OpenNebula y su interfaz de gestión basada en Web.

Para la implementación, se utilizó una infraestructura gestionada por Canonical MAAS y se desplegaron 6 máquinas físicas con Ubuntu Server 22.04. Las especificaciones de hardware de los servidores son las siguientes: cuatro servidores cuentan con un total de 512 GB de RAM, equipados cada uno con un disco SSD de 480 GB para el sistema operativo y dos discos SSD de 4 TB para un clúster de almacenamiento distribuido; los dos servidores restantes tienen 16 GB de RAM y un disco SSD de 480 GB cada uno. Todos los servidores están equipados con al menos dos interfaces de red Gigabit Ethernet: una en el puente 'br0' para la gestión de la red de infraestructura y otra en el puente 'br-ex' con Open vSwitch para gestionar las diversas VLAN que serán accesibles desde OpenNebula. La estandarización de los la máquinas físicas desplegadas puede ser consultada en el Anexo A.

El clúster de infraestructura de servicios en nube de alta disponibilidad se compone de un conjunto de servidores de hardware heterogéneo. A pesar de esta heterogeneidad, la configuración de cada uno de los nodos fue estandarizada, adoptando una estructura de almacenamiento y particionado uniforme, así como una configuración de red consistente para todos los servidores.

Arquitectura del Clúster OpenNebula en Producción

En la Figura 5, se presenta la arquitectura del Cluster OpenNebula en Producción desplegado con Canonical MaaS cuya implementación se describe a continuación.

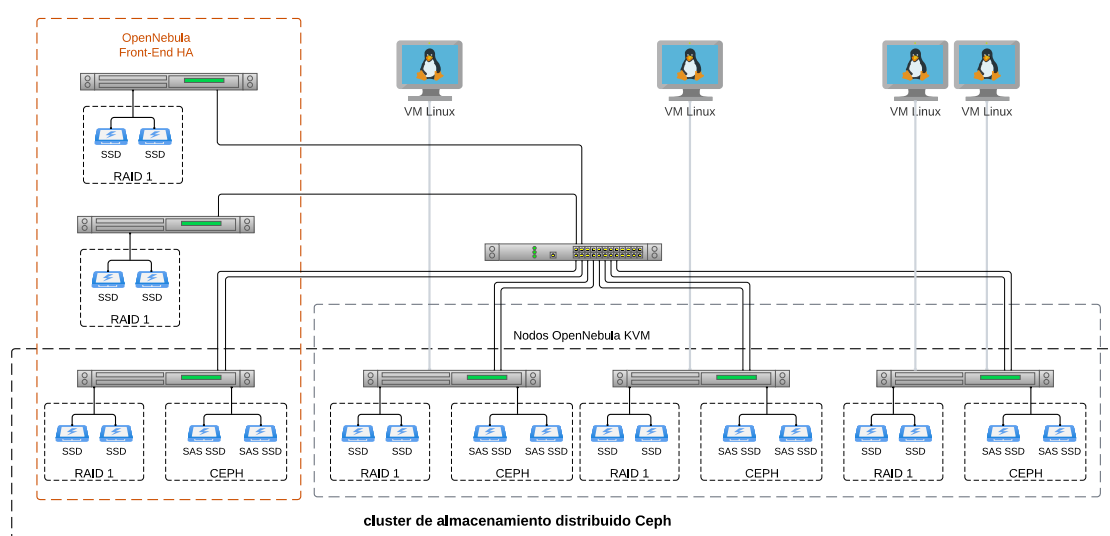


Figura 5: Arquitectura del Clúster OpenNebula en producción.

Configuración de Red Subyacente

La red subyacente está estructurada en varias VLANs, destacando especialmente la VLAN de infraestructura y la VLAN de Internet. Para facilitar una mayor independencia en la interacción de OpenNebula con la red a nivel de capa 2, se han configurado y autorizado 20 VLANs en los enlaces trunk de cada servidor del clúster de OpenNebula. Esta capacidad de etiquetado de VLAN permite a OpenNebula ofrecer a los usuarios una mayor independencia para configurar sus propias estructuras de red, mejorando así la personalización y la seguridad en el acceso a los servicios de red.

Implementación de OpenNebula en alta disponibilidad¹

El proceso de implementación de OpenNebula en un entorno de alta disponibilidad se organiza en cuatro fases esenciales. La primera fase involucra el despliegue del 'front-end' en cada uno de los tres nodos destinados al panel de control ('dashboard'). La segunda fase se enfoca en configurar estos nodos para garantizar la alta disponibilidad. En la tercera fase, se lleva a cabo el despliegue de tres nodos con el hipervisor KVM de OpenNebula. Finalmente, la cuarta fase consiste en establecer los 'hooks' necesarios para activar automáticamente las máquinas virtuales en caso de fallo de un nodo KVM.

Como se ha mencionado anteriormente, el sistema operativo base para la implementación es Ubuntu 22.04, el cual fue desplegado utilizando Canonical MaaS. En consecuencia, la descripción general del proceso de implementación presupone la disponibilidad de un sistema operativo adecuadamente configurado y preparado para este fin.

Front-end de OpenNebula en Alta Disponibilidad

En el proceso de instalación de OpenNebula, es fundamental comenzar actualizando los repositorios y proceder con la instalación de ciertos paquetes que son requisitos previos específicos, no cubiertos automáticamente por el gestor de paquetes en términos de resolución de dependencias. Según el manual de OpenNebula, antes de iniciar la instalación propiamente dicha, se deben instalar los paquetes gnupg, wget y apt-transport-https. Los comandos necesarios para esta instalación son:

```
$ sudo apt update
$ sudo apt -y install gnupg wget apt-transport-https
```

Carga del repositorio OpenNebula

Antes de la instalación de los paquetes necesarios para la ejecución del Front-End, es imperativo incorporar los repositorios de OpenNebula al sistema operativo. Este proceso incluye la descarga e instalación de la llave pública, la inicialización de las variables de entorno², y la adición del repositorio al sistema. Los comandos para llevar a cabo estas tareas son los siguientes:

-
- 1 Para este despliegue es necesario contar con un storage en alta disponibilidad, en este caso se ha usado un clúster Ceph, cuya instalación está disponible en el Anexo B.
 - 2 En el caso de las variables de entorno solo es usada \$VERSION_ID

```
$ curl -fsSL https://downloads.opennebula.io/repo/repo2.key | \
sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/opennebula.gpg
$ source /etc/os-release
$ echo "deb https://downloads.opennebula.org/repo/6.8/Ubuntu/$VERSION_ID \
stable opennebula" | sudo tee /etc/apt/sources.list.d/opennebula.list
```

Instalación de paquetes de OpenNebula

Para el correcto funcionamiento de la interfaz de gestión basada en Web de OpenNebula (Sunstone), es necesario instalar los siguientes paquetes, cada uno cumpliendo funciones específicas dentro del ecosistema:

OpenNebula (Paquete Principal): Este paquete contiene los componentes esenciales del sistema de gestión de recursos en la nube, encargados de la orquestación y administración del centro de datos virtual.

OpenNebula Sunstone (opennebula-sunstone): Proporciona la interfaz gráfica web para la administración de la infraestructura de nube, permitiendo a los usuarios gestionar recursos de manera visual.

OpenNebula Gate (opennebula-gate): Actúa como un puente entre los distintos componentes de OpenNebula y las aplicaciones externas, facilitando la integración y la extensibilidad.

OpenNebula Flow (opennebula-flow): Ofrece herramientas para la automatización y el orquestamiento de servicios a través de múltiples zonas de nube, mejorando la gestión de grandes entornos.

OpenNebula FireEdge (opennebula-fireedge): Aporta funcionalidades avanzadas para la gestión de redes y la provisión de aplicaciones a través de una interfaz simplificada.

Comando de instalación:

```
$ sudo apt install opennebula opennebula-sunstone \
opennebula-gate opennebula-flow opennebula-fireedge
```

Instalación de Gemas de Ruby

Diversos componentes de OpenNebula dependen de librerías específicas de Ruby para su funcionamiento. Para facilitar la instalación de estas librerías y algunos paquetes de librerías de desarrollo necesarios, OpenNebula incluye un script dedicado. Este script asegura que todas las gemas necesarias estén disponibles y correctamente configuradas, simplificando significativamente el proceso de configuración inicial. La instalación de las gemas se realiza mediante el siguiente comando:

```
$ sudo /usr/share/one/install_gems
```

Este comando ejecuta el script `install_gems`, ubicado en `/usr/share/one`, que gestiona automáticamente la instalación de las gemas y las dependencias de desarrollo requeridas.

Configurar OpenNebula

Configuración de OpenNebula con MariaDB

Selección del Motor de Base de Datos:

OpenNebula requiere de una base de datos robusta para almacenar su estado y configuración. En este caso, se ha seleccionado MariaDB como el motor de base de datos, configurado en un clúster Galera de tres nodos para garantizar la alta disponibilidad. Esta configuración de clúster Galera se encuentra disponible en el Anexo C.

Configuración de la Base de Datos para OpenNebula:

Una vez confirmado que el clúster está operativo, se procede a crear la base de datos específica para OpenNebula:

```
$ sudo mysql -u root -p
CREATE DATABASE opennebula;
GRANT ALL PRIVILEGES ON opennebula.* TO 'oneadmin' IDENTIFIED BY
'NebulaDBPass';
FLUSH PRIVILEGES;
EXIT;
```

Es recomendable documentar el nombre de la base de datos, el usuario y la contraseña para futuras referencias durante la configuración de OpenNebula.

Modificación de la Configuración Predeterminada:

Por defecto, OpenNebula está configurado para usar SQLite. Para adaptar OpenNebula al uso de MariaDB, es necesario ajustar la configuración en el archivo `/etc/one/oned.conf`. Este cambio implica comentar la configuración predeterminada de SQLite y descomentar y actualizar la configuración para MySQL (compatible con MariaDB), como se muestra a continuación:

Cambios en el Archivo de Configuración `/etc/one/oned.conf`:

Comentar la línea de SQLite para desactivarla:

```
#DB = [ BACKEND = "sqlite" ]
```

Decomentar y configurar las líneas para MySQL, ajustándolas a la configuración específica del clúster MariaDB:

```
DB = [ backend = "mysql",
server = "localhost",
port = 3306,
user = "oneadmin",
passwd = "M1PaS5w0Rd", # Reemplazar con la contraseña real
db_name = "opennebula"
]
```

Alta Disponibilidad de Sunstone y Fireedge en OpenNebula

Para garantizar la continuidad y la disponibilidad del servicio del dashboard de OpenNebula, la configuración en alta disponibilidad se establece mediante un líder y dos seguidores. Esta configuración permite que, en caso de fallo del líder o de alguno de los seguidores, el sistema pueda seguir operativo sin interrupciones significativas.

Líder: El líder es el nodo principal que gestiona y coordina las operaciones del dashboard. Es el punto de contacto inicial para las operaciones y el mantenimiento del clúster.

Seguidores: Los seguidores son réplicas del líder que mantienen estados sincronizados con el líder. En caso de fallo del líder, uno de los seguidores puede asumir automáticamente el rol de líder para mantener la operatividad del sistema.

IP Flotante: Se configura una dirección IP flotante que siempre apunta al nodo que actualmente actúa como líder. Esto asegura que los usuarios y aplicaciones siempre tengan acceso al dashboard, independientemente de cuál nodo esté funcionando como líder en un momento dado.

Esta estructura de alta disponibilidad es crítica para mantener un acceso constante y confiable al dashboard de OpenNebula, crucial para la gestión y supervisión eficiente de los recursos de la nube.

Configuración del Líder Inicial en OpenNebula

El proceso de configuración en un ambiente de alta disponibilidad comienza con la configuración del servidor líder, que será responsable de iniciar y gestionar las operaciones del clúster. Este proceso se realiza en los siguientes pasos:

1. Iniciar OpenNebula en el Servidor Líder: El servidor designado como líder arranca OpenNebula para comenzar la configuración del clúster.

2. Añadir el Propio Servidor a la Zona: Para integrar el servidor líder en la gestión del clúster, se añade a la zona predeterminada usando los siguientes comandos:

2.1. Listar las Zonas Disponibles:

El primer paso es verificar las zonas existentes, para asegurar que el servidor se añada a la zona correcta.

```
$ onezone list
```

Salida esperada:

C	ID	NAME	ENDPOINT
*	0	OpenNebula	http://localhost:2633/RPC2

Aquí, C indica la zona actualmente activa, ID es el identificador de la zona, NAME es el nombre asignado a la zona, y ENDPOINT muestra la dirección de conexión al servicio.

2.2. Añadir el Servidor a la Zona 0:

Utilizando el ID de la zona obtenido anteriormente, el siguiente comando añade el servidor líder a esta zona.

```
$ onezone server-add 0 --name server-0 \  
--rpc http://172.24.250.9:2633/RPC2
```

2.3. Verificar la Configuración de la Zona:

Finalmente, para confirmar que el servidor ha sido correctamente añadido y configurado, se utiliza el comando para mostrar los detalles de la zona.

```
$ onezone show 0
```

Salida esperada:

```
ZONE 0 INFORMATION  
ID          : 0  
NAME       : OpenNebula  
STATE      : ENABLED  
  
ZONE SERVERS  
ID  NAME          ENDPOINT  
server-0  http://172.24.250.9:2633/RPC2  
  
HA & FEDERATION SYNC STATUS  
ID  NAME          STATE      TERM      INDEX      COMMIT      VOTE  
FED_INDEX  
ZONE TEMPLATE  
ENDPOINT="http://localhost:2633/RPC2"
```

En esta sección, se muestra información detallada sobre la zona, incluyendo el estado de la zona (ENABLED indica que está activa), los servidores que forman parte de la zona con sus respectivos puntos de conexión y la configuración de sincronización y federación.

Habilitar el Identificador de Servidor y Configuración de IP Flotante

En un clúster OpenNebula no federado, es crucial establecer adecuadamente el identificador de servidor. Este ajuste se realiza en el archivo de configuración `/etc/one/oned.conf` bajo la sección FEDERATION. Para una configuración no federada, el `Server_ID` debe establecerse en 0. Aquí se muestra la configuración relevante:

```
FEDERATION = [  
  MODE          = "STANDALONE",  
  ZONE_ID       = 0,  
  SERVER_ID     = 0, # cambiar de -1 a 0  
  MASTER_ONED  = ""  
]
```

Esta configuración especifica que el clúster opera de manera autónoma sin federación con otros clústers, identificándose como la zona principal (`ZONE_ID = 0`) y el servidor principal (`SERVER_ID = 0`).

Configuración de Hooks para IP Flotante

Para garantizar la alta disponibilidad del servicio de monitoreo, es necesario configurar acciones automáticas (hooks) que se ejecuten cuando un servidor cambie su estado de líder a seguidor y viceversa. Estas acciones gestionarán la IP flotante, permitiendo que el servicio de monitoreo siempre apunte al líder actual del clúster. La configuración para estos hooks se define como sigue:

```
# Ejecutar cuando un servidor transita de follower->leader
RAFT_LEADER_HOOK = [
    COMMAND = "raft/vip.sh",
    ARGUMENTS = "leader eth0 172.24.250.250/24"
]

# Ejecutar cuando un servidor transita de leader->follower
RAFT_FOLLOWER_HOOK = [
    COMMAND = "raft/vip.sh",
    ARGUMENTS = "follower eth0 172.24.250.250/24"
]
```

Estos hooks ejecutan un script vip.sh, que ajusta la configuración de red para que la dirección IP 172.24.250.250/24 se asigne al líder actual, asegurando que todas las conexiones dirigidas al clúster se redirijan apropiadamente.

Replicación de Configuraciones en Nodos Seguidores en un Entorno de Alta Disponibilidad

Tras establecer la configuración inicial en el servidor líder, resulta imperativo replicar estas configuraciones en todos los nodos seguidores del clúster de OpenNebula. Esta replicación garantiza que, ante un fallo del nodo líder, cualquier nodo seguidor pueda asumir el liderazgo de manera inmediata, asegurando la continuidad del servicio sin interrupciones. Los pasos necesarios para asegurar una replicación efectiva incluyen:

1. **Sincronización de Archivos de Configuración:** Es esencial que los archivos de configuración, como `/etc/one/oned.conf`, sean idénticos en todos los nodos. Esta sincronización puede efectuarse mediante herramientas automatizadas o mediante procesos manuales establecidos en la fase inicial de configuración.
2. **Implementación de Hooks de RAFT:** Los scripts destinados a la gestión de la IP flotante, incluyendo `RAFT_LEADER_HOOK` y `RAFT_FOLLOWER_HOOK`, deben estar instalados y configurados correctamente en cada nodo. Estos scripts deben incluir los argumentos necesarios para manejar adecuadamente las transiciones de liderazgo.
3. **Pruebas de Conmutación por Error (Failover):** Se deben llevar a cabo pruebas rigurosas para verificar que los nodos seguidores pueden efectivamente asumir el liderazgo sin interrupciones. Estas pruebas implican la simulación controlada de fallos en el nodo líder para observar si el seguidor asume correctamente el liderazgo.

Incorporación de Nuevos Servidores al Clúster de OpenNebula

Para expandir la capacidad y mejorar la resiliencia del clúster de OpenNebula, es posible añadir nuevos servidores. Es esencial realizar este proceso de manera

secuencial, incorporando un solo host a la vez para asegurar una integración correcta y evitar complicaciones en la configuración del clúster. El procedimiento para añadir un nuevo servidor al clúster se describe a continuación:

1. **Preparación del Nuevo Servidor:** Antes de incorporar el servidor al clúster, se debe verificar que está correctamente configurado y operativo, con OpenNebula y todas las dependencias necesarias instaladas. Además, debe tener una dirección IP estática asignada, en este caso 172.24.250.10.
2. **Adición del Servidor al Clúster:** En el nodo líder del clúster, el nuevo servidor se agrega utilizando el comando `onezone server-add`. Este comando requiere especificar el identificador de la zona (en la mayoría de los casos, 0 para la zona principal), el nombre asignado al nuevo servidor y la dirección RPC que el líder utilizará para comunicarse con él:

```
$ onezone server-add 0 --name server-1 --rpc http://172.24.250.10:2633/RPC2
```

En este comando, `--name server-1` designa el identificador único para el nuevo servidor dentro del clúster, y `--rpc http://172.24.250.10:2633/RPC2` especifica la interfaz a través de la cual el servidor líder se comunicará con el nuevo servidor.

Verificación de la Incorporación de Nuevos Servidores al Clúster de OpenNebula

Una vez incorporado el nuevo servidor al clúster, es esencial realizar una verificación para asegurar que la integración se ha completado correctamente. Esta verificación puede realizarse mediante el comando `onezone show 0`, que proporciona información detallada sobre la configuración del clúster, incluyendo los servidores que lo componen:

```
$ onezone show 0
```

El resultado esperado debe mostrar que el nuevo servidor ha sido añadido y está correctamente configurado dentro del clúster:

```
ZONE 0 INFORMATION
ID           : 0
NAME         : OpenNebula
STATE        : ENABLED

ZONE SERVERS
ID  NAME           ENDPOINT
server-0  http://172.24.250.9:2633/RPC2
server-1  http://172.24.250.10:2633/RPC2

HA & FEDERATION SYNC STATUS
ID  NAME           STATE   TERM   INDEX  COMMIT  VOTE  FED_INDEX
server-0  leader   1      19     19     -1     -1
server-1  error    -      -      -      -      -

ZONE TEMPLATE
ENDPOINT="http://localhost:2633/RPC2"
```

Este informe muestra cada servidor con su identificador, nombre y punto de conexión. Además, el estado de sincronización y federación de cada servidor es crucial para entender su rol y estado actual dentro del clúster. En caso de observar estados como error en algún servidor, es necesario investigar y resolver estos problemas para

asegurar la funcionalidad plena del clúster.

Este paso de verificación es fundamental para confirmar que la estructura del clúster mantiene su integridad y está preparada para operar de manera eficiente y continua.

Inicio de Servicios de OpenNebula

Para garantizar una operatividad completa y eficiente del panel de control de OpenNebula y sus componentes asociados, se recomienda iniciar todos los servicios principales al mismo tiempo. Este enfoque asegura que todas las funcionalidades estén disponibles de inmediato y reduce la posibilidad de errores o problemas de dependencia entre servicios. Basado en observaciones prácticas y discusiones en foros especializados, el comando recomendado para iniciar los servicios es:

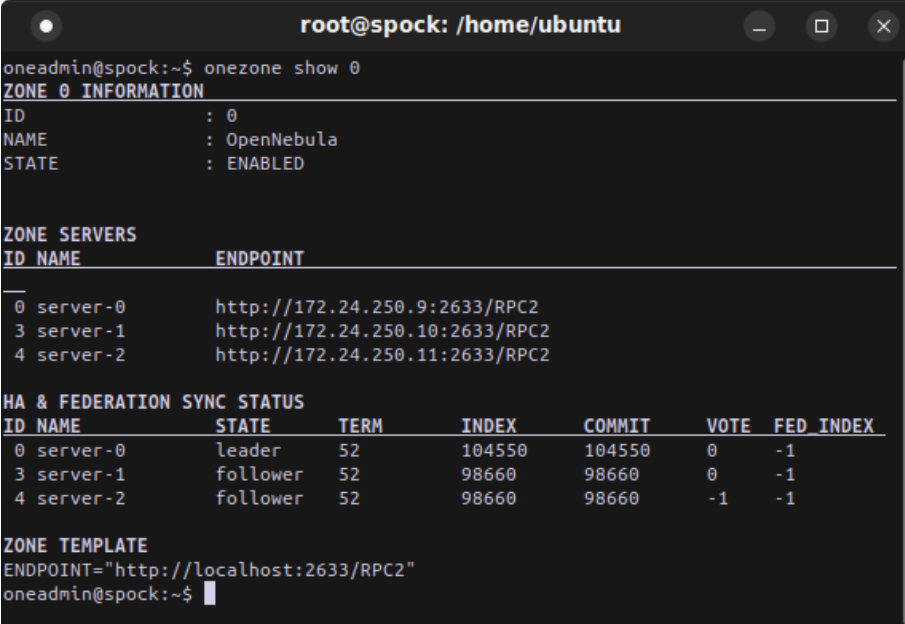
```
$ sudo systemctl start opennebula opennebula-sunstone opennebula-fireedge \
opennebula-gate opennebula-flow
```

Además, para asegurar que estos servicios se inicien automáticamente en el arranque del sistema, deben ser habilitados utilizando el siguiente comando:

```
$ sudo systemctl enable opennebula opennebula-sunstone opennebula-fireedge \
opennebula-gate opennebula-flow
```

Este procedimiento contribuye a una configuración robusta y fiable del entorno de OpenNebula, fundamental para mantener la alta disponibilidad y la gestión eficaz de la infraestructura de nube.

Tras completar la instalación del Front-End, la zona de OpenNebula ha sido configurada adecuadamente. La Imagen 2 muestra la captura de terminal con la configuración de la zona 0, identificando a los servidores con direcciones 172.24.250.9, 172.24.250.10 y 172.24.250.11. En esta configuración, server-0 actúa como líder, mientras que server-1 y server-2 funcionan como seguidores.



```
root@spock: /home/ubuntu
oneadmin@spock:~$ onezone show 0
ZONE 0 INFORMATION
-----
ID          : 0
NAME        : OpenNebula
STATE       : ENABLED

ZONE SERVERS
-----
ID  NAME      ENDPOINT
--  -
0   server-0  http://172.24.250.9:2633/RPC2
3   server-1  http://172.24.250.10:2633/RPC2
4   server-2  http://172.24.250.11:2633/RPC2

HA & FEDERATION SYNC STATUS
-----
ID  NAME      STATE   TERM   INDEX  COMMIT  VOTE  FED_INDEX
--  -
0   server-0  leader  52     104550 104550  0     -1
3   server-1  follower 52     98660  98660  0     -1
4   server-2  follower 52     98660  98660 -1     -1

ZONE TEMPLATE
ENDPOINT="http://localhost:2633/RPC2"
oneadmin@spock:~$
```

Imagen 2: Zona 0 del Cluster OpenNebula.

Verificación del Funcionamiento de OpenNebula Sunstone

Para verificar que OpenNebula Sunstone está operativo, se puede acceder a través de un navegador web utilizando la dirección IP flotante configurada. No obstante, para incrementar la seguridad del servicio, es recomendable habilitar un proxy de acceso que utilice el protocolo HTTPS. En este caso la URL sería `http://172.24.250.250:9869`. La captura de pantalla correspondiente puede ser vista en el Anexo D.

Instalación de OpenNebula KVM

Antes de proceder con la instalación de OpenNebula KVM, es imprescindible asegurar que el panel de control ('dashboard') de OpenNebula esté completamente operativo. Dado el diseño de la arquitectura propuesta, se utilizarán tres hipervisores. Por lo tanto, es necesario realizar la instalación en cada uno de los servidores que conforman el clúster.

Preparación de Repositorios: Al igual que en la instalación del Front-End, la configuración de OpenNebula KVM comienza con la incorporación de los repositorios de paquetes necesarios. Los pasos para preparar los repositorios son idénticos a los utilizados previamente:

1. **Descarga e Importación de Llaves Públicas:** Para garantizar la autenticidad de los paquetes descargados, se debe descargar e importar la llave pública del repositorio de OpenNebula. Utilice el siguiente comando para realizar esta tarea:

```
$ curl -fsSL https://downloads.opennebula.io/repo/repo2.key | \
sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/opennebula.gpg
```

2. **Definición de Variables de Entorno y Adición de Repositorios:** Es necesario definir las variables de entorno apropiadas y añadir el repositorio de OpenNebula al sistema operativo. Ejecute los siguientes comandos para completar esta configuración:

```
$ source /etc/os-release
$ echo "deb https://downloads.opennebula.org/repo/6.8/Ubuntu/$VERSION_ID
stable \ opennebula" | sudo tee /etc/apt/sources.list.d/opennebula.list
```

Estos preparativos son esenciales para asegurar que todos los componentes necesarios estén disponibles y que los paquetes descargados sean seguros y confiables. Una vez completados estos pasos, se puede proceder con la instalación propiamente dicha de OpenNebula KVM en cada uno de los servidores del clúster.

Instalación de los paquetes OpenNebula KVM

Para instalar OpenNebula KVM en cada nodo del clúster, es necesario ejecutar una serie de comandos que permitirán la instalación del paquete y la configuración adecuada de los servicios necesarios. A continuación se detallan los pasos específicos:

1. **Actualización de Paquetes y Instalación de OpenNebula KVM:** Antes de proceder con la instalación específica, es esencial actualizar el índice de paquetes del sistema y luego instalar el paquete que contiene los

componentes necesarios para el nodo de KVM:

```
$ sudo apt update
$ sudo apt install opennebula-node -y
```

2. **Configuración de Libvirt para Integración con OpenNebula:** Para asegurar que el demonio libvirt funcione correctamente con los privilegios de usuario de OpenNebula (oneadmin), se deben modificar las configuraciones en el archivo `/etc/libvirt/libvirtd.conf`. Utilice un editor como vi para editar el archivo:

```
$ sudo vi /etc/libvirt/libvirtd.conf
```

Asegúrese de que el archivo contenga las siguientes líneas, que configuran los permisos adecuados para el grupo oneadmin:

```
unix_sock_group = "oneadmin"
unix_sock_rw_perms = "0777"
```

3. **Reinicio de Libvirt:** Cada vez que se realicen cambios en la configuración de libvirt, es necesario reiniciar el servicio para que los cambios tengan efecto. Ejecute el siguiente comando para reiniciar el servicio libvirtd:

```
$ sudo systemctl restart libvirtd.service
```

Este procedimiento garantiza que los nodos del clúster están correctamente configurados para operar con OpenNebula KVM, manteniendo los estándares de seguridad y funcionalidad necesarios para una gestión eficaz de la infraestructura virtual.

Configuración de Claves Públicas para SSH en OpenNebula

En el entorno de OpenNebula, la comunicación entre el nodo Front-end y los nodos del hipervisor se realiza a través de SSH. Es esencial que las claves SSH estén adecuadamente configuradas para facilitar una comunicación segura y sin interrupciones. A continuación, se describen los procedimientos necesarios para configurar estas claves:

1. **Distribución de la Clave Pública:** La clave pública del usuario oneadmin se debe distribuir en el archivo `/var/lib/one/.ssh/authorized_keys` en cada host del clúster. Esta distribución se realiza automáticamente durante la instalación del paquete en el Front-end mediante la generación de una clave SSH y la actualización del archivo `authorized_keys`.
2. **Creación del Archivo `known_hosts`:** Para prevenir solicitudes de verificación de identidad del host al establecer conexiones SSH por primera vez, es crucial crear y configurar el archivo `known_hosts`. Se recomienda ejecutar el siguiente comando en el nodo Front-end, actuando bajo el usuario oneadmin y especificando todos los nombres de los nodos involucrados, incluido el del Front-end:

```
$ sudo su - oneadmin
$ ssh-keyscan <frontend> <node1> <node2> ... >>
/var/lib/one/.ssh/known_hosts
```

3. **Sincronización del Directorio .ssh:** Para asegurar que todas las configuraciones de SSH se replican en todos los nodos, se debe copiar el directorio `/var/lib/one/.ssh` desde el Front-end hacia cada uno de los nodos del clúster:

```
$ scp -rp /var/lib/one/.ssh <node1>:/var/lib/one/  
$ scp -rp /var/lib/one/.ssh <node2>:/var/lib/one/
```

4. **Restablecimiento de la Contraseña de oneadmin:** En caso de ser necesario, se puede restablecer la contraseña del usuario `oneadmin` en todos los nodos para asegurar que no existan discrepancias en la autenticación:

```
$ sudo passwd oneadmin
```

5. **Verificación de la Conexión SSH:** Para confirmar que la configuración de SSH ha sido implementada correctamente, es aconsejable verificar que se pueda conectar desde el nodo Front-end a cada nodo del hipervisor sin que se solicite una contraseña:

Estos pasos aseguran que la comunicación entre el Front-end y los nodos del hipervisor en OpenNebula se realice de manera segura y eficiente, facilitando la gestión automatizada del clúster.

Configuración de Red para el Anfitrión de OpenNebula

La infraestructura de OpenNebula requiere una red con una capacidad mínima de 1 Gbps para conectar el nodo Front-end con los hosts del hipervisor, facilitando así la gestión, la monitorización y la transferencia de archivos de imagen. Se recomienda encarecidamente el uso de una red dedicada exclusivamente para propósitos de gestión. OpenNebula admite cuatro modos de red distintos, cada uno adaptado a diferentes requisitos y escenarios de implementación:

1. **Modo Bridge:** En este modo, la máquina virtual se conecta directamente a un puente existente en el hipervisor. Este modo permite la configuración de grupos de seguridad y el aislamiento de red, proporcionando así una capa adicional de seguridad.
2. **VLAN:** En este modo, las redes virtuales se implementan utilizando el etiquetado VLAN según el estándar 802.1Q, que ayuda en la segmentación y el aislamiento de diferentes secciones de la red.
3. **VXLAN:** Similar al modo VLAN, las redes virtuales en VXLAN utilizan una encapsulación UDP y multidifusión IP para extender las redes a través de largas distancias y diferentes redes físicas.
4. **Open vSwitch:** Este modo es similar al VLAN pero emplea Open vSwitch en lugar de un puente tradicional de Linux, ofreciendo una mayor flexibilidad y capacidad de configuración.

En la configuración definida en este proyecto, se utilizan dos interfaces de red: una para propósitos de gestión en modo Bridge, y otra que emplea Open vSwitch para la implementación de VLANs. Esta configuración de red se establece a través de la capa MaaS y se crea específicamente para cada nodo durante el despliegue del sistema operativo en los servidores físicos que conforman el clúster de OpenNebula.

Uso del Ceph en Datastore

Configuración del Clúster Ceph para OpenNebula KVM

Para implementar Ceph como solución de almacenamiento de alta disponibilidad, es esencial configurarlo en cada nodo del clúster OpenNebula KVM. Cada hipervisor en los nodos utilizará Ceph para gestionar las imágenes de disco de las máquinas virtuales.

Creación del Pool en Ceph

El primer paso consiste en crear un pool en Ceph específicamente para los datastores de OpenNebula. Los siguientes comandos CLI de Ceph se utilizan para esta configuración:

```
$ ceph osd pool create one 128
$ ceph osd pool application enable one rbd
$ bd pool init -p one
```

Para confirmar que el pool ha sido correctamente creado, se puede utilizar el siguiente comando y verificar la salida:

```
$ ceph osd lspools
```

Para confirmar que el pool ha sido correctamente creado, se puede utilizar el siguiente comando y verificar la salida:

```
data,1 metadata,2 rbd,6 one,
```

Configuración de Usuario Ceph

Como siguiente paso, se debe definir un usuario en Ceph que gestionará el acceso a los almacenes de datos. Este usuario también será utilizado por libvirt para acceder a las imágenes de disco:

```
$ ceph auth get-or-create client.libvirt \
mon 'profile rbd' osd 'profile rbd pool=one
```

Posteriormente, es necesario obtener y distribuir la clave de este usuario a los nodos OpenNebula:

```
$ ceph auth get-key client.libvirt | tee client.libvirt.key
$ ceph auth get client.libvirt -o ceph.client.libvirt.keyring
```

Recomendaciones para el Formato RBD

Para mejorar el acceso en RBD, se recomienda utilizar el formato de protocolo 2 para RBD, definiendo este formato en la configuración del Clúster Ceph y en cada uno de los nodos KVM. El archivo para esta configuración es `/etc/ceph/ceph.conf`:

```
[global]
rbd_default_format = 2
```


Configuración en el Front-End y Nodos KVM

Para integrar el clúster Ceph, es necesario realizar las siguientes configuraciones en los hosts:

- Las herramientas cliente de Ceph deben estar instaladas en la máquina.
- La configuración del demonio mon debe estar definida en ceph.conf para todos los nodos, eliminando la necesidad de especificar el nombre de host y el puerto en los comandos de Ceph.
- Distribuir el llavero de usuario Ceph (ceph.client.libvirt.keyring) y la clave de usuario (client.libvirt.key) a los nodos apropiados:

```
$ scp ceph.client.libvirt.keyring root@node:/etc/ceph
$ scp client.libvirt.key oneadmin@node:~/
```

Estos pasos aseguran una integración efectiva de Ceph con el clúster OpenNebula KVM, proporcionando un entorno de almacenamiento robusto y de alta disponibilidad.

Configuración de Nodos KVM para Integración con Ceph

Para una integración efectiva de los nodos KVM con el clúster Ceph, es necesario configurar adecuadamente las credenciales en libvirt:

1. **Generación y Distribución de un Secreto para el Usuario Ceph:** Genera un identificador único (UUID) para el secreto que se utilizará para la autenticación y anótalo para su uso posterior:

```
$ UUID=`uuidgen`; echo $UUID
```

Ejemplo de salida: 'c7bdeabf-5f2a-4094-9413-58c6a9590980'.

Crea un archivo XML para definir el secreto en libvirt y cópialo al home de oneadmin en cada nodo:

```
$ cat > secret.xml <<EOF
<secret ephemeral='no' private='no'>
  <uuid>$UUID</uuid>
  <usage type='ceph'>
    <name>client.libvirt secret</name>
  </usage>
</secret>
EOF
$ scp secret.xml oneadmin@node:~/
```

2. **Configuración de Secretos en Libvirt y Eliminación de Archivos de Claves:** Define el secreto en libvirt y establece el valor utilizando la clave del usuario Ceph:

```
$ virsh -c qemu:///system secret-define secret.xml
$ virsh -c qemu:///system secret-set-value --secret $UUID --
base64 $(cat client.libvirt.key)
$ rm client.libvirt.key
```

3. **Verificación de Configuración del Cliente Ceph en el Nodo:** Asegúrate de que el cliente Ceph esté correctamente configurado y pueda acceder a los discos en el pool especificado:

```
$ ssh oneadmin@node
$ rbd ls -p one --id libvirt
```

Creación de Almacenes de Datos en OpenNebula

- **Almacén de Datos del Sistema (System Datastore):** Crea un almacén de datos del sistema en Sunstone o mediante la CLI. Define los atributos necesarios en un archivo de configuración y utiliza el comando de creación:

```
$ cat > system.ds <<EOF
NAME = ceph_system
TM_MAD = ceph
TYPE = SYSTEM_DS
DISK_TYPE = RBD
POOL_NAME = one
CEPH_HOST = "host1 host2:port2"
CEPH_USER = libvirt
CEPH_SECRET = "6f88b54b-5dae-41fe-a43e-b2763f601cfc"
BRIDGE_LIST = cephfrontend
EOF
$ onedatastore create system.ds
ID: 100
```

- **Almacén de Datos de Imágenes (Image Datastore):** Similar al almacén de datos del sistema, configura y crea un almacén de datos de imágenes utilizando los mismos atributos:

```
$ cat > image.ds <<EOF
NAME = "cephds"
DS_MAD = ceph
TM_MAD = ceph
DISK_TYPE = RBD
POOL_NAME = one
CEPH_HOST = "host1 host2:port2"
CEPH_USER = libvirt
CEPH_SECRET = "6f88b54b-5dae-41fe-a43e-b2763f601cfc"
BRIDGE_LIST = cephfrontend
EOF
$ onedatastore create image.ds
ID: 101
```

Estos pasos detallan cómo configurar y verificar los nodos KVM para su uso con Ceph en un entorno OpenNebula, asegurando que todos los aspectos críticos sean cubiertos para un funcionamiento eficiente y seguro.

Incorporación de Hosts al Clúster de OpenNebula Durante el Despliegue Inicial

Durante el despliegue inicial del clúster de OpenNebula, es esencial configurar el número mínimo de nodos requeridos para operar el clúster de manera funcional. Para incorporar estos nodos al clúster, se puede utilizar tanto la interfaz de línea de

comandos (CLI) como la interfaz gráfica Sunstone. A través de la CLI, el procedimiento para registrar un nuevo nodo se realiza con el siguiente comando:

```
$ onehost create <node01> -i kvm -v kvm
```

Este comando agrega un host al clúster especificando que el hipervisor utilizado es de tipo KVM. Es fundamental que OpenNebula esté correctamente configurado en el nodo para que su incorporación al clúster sea exitosa. Una vez que el nodo está correctamente configurado y añadido, comenzará a participar activamente en la gestión y distribución de las cargas de trabajo dentro del clúster.

Configuración de máquinas virtuales en alta disponibilidad

Para asegurar la alta disponibilidad en OpenNebula, es esencial contar con ciertos componentes clave ya establecidos en la arquitectura del sistema:

1. **Tres Hipervisores KVM:** Proporcionan la redundancia necesaria para el mantenimiento y fallos imprevistos.
2. **Almacenamiento Compartido HA:** Esencial para la gestión de datos consistente y continua entre hipervisores.
3. **Red subyacente replicada:** Necesaria para una gestión de red flexible y resiliente a fallos en todos los nodos KVM.

Estos elementos forman la base infraestructural sobre la cual se implementan los mecanismos de alta disponibilidad específicos, como los hooks de OpenNebula, que intervienen en caso de detección de fallos por el sistema de monitoreo y que serán abordados a continuación.

Detección de Fallos de Host en OpenNebula

OpenNebula incluye mecanismos para detectar fallos en los hosts. Cuando el sistema detecta que un host está en estado de ERROR, se puede activar un hook diseñado para manejar esta situación. Este proceso es crucial para reducir el tiempo de inactividad resultante de fallos de hardware, permitiendo la redistribución rápida de máquinas virtuales a otros hosts disponibles.

El sistema proporciona un script de ejemplo que se puede configurar como hook. Este script se activa cuando un host cambia a estado de ERROR. Los pasos para configurar este hook utilizando la plantilla proporcionada por OpenNebula son:

1. Revisar el Script de Hook de Ejemplo:

```
$ cat /usr/share/one/examples/host_hooks/error_hook
ARGUMENTS          = "$TEMPLATE -m -p 5"
ARGUMENTS_STDIN    = "yes"
COMMAND            = "ft/host_error.rb"
NAME               = "host_error"
STATE              = "ERROR"
REMOTE             = "no"
RESOURCE           = HOST
TYPE               = state
```

2. Configurar el Hook: Configurar el hook con los parámetros necesarios para reaccionar ante fallos:

- **ARGUMENTS:** Argumentos pasados al script, incluyendo la migración automática de máquinas virtuales.
- **ARGUMENTS_STDIN:** Indica si los argumentos deben ser leídos de la entrada estándar.
- **COMMAND:** El comando a ejecutar, que es el script Ruby para manejar errores de host.
- **NAME:** Nombre del hook.
- **STATE:** El estado del host que activará el hook, "ERROR".
- **REMOTE:** Define si el comando se ejecuta localmente o en un host remoto.
- **RESOURCE:** El recurso afectado, en este caso, el host.
- **TYPE:** Tipo de evento que dispara el hook, cambio de estado en este contexto.

3. Crear el hook con el comando:

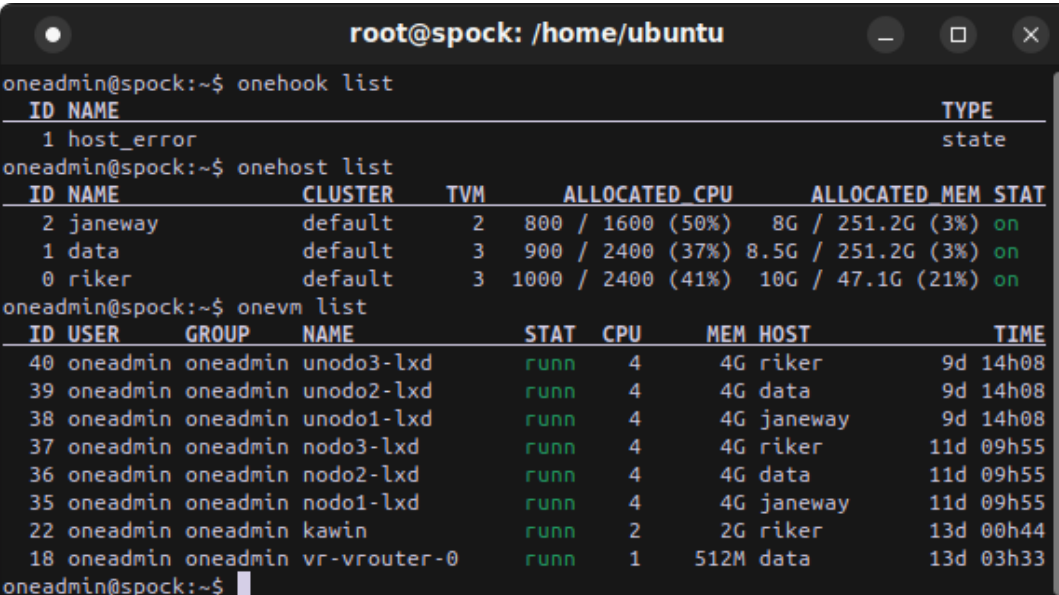
```
$ onehook create /usr/share/one/examples/host_hooks/error_hook
```

Este enfoque directo mejora la respuesta del clúster ante fallos y facilita la gestión eficiente en situaciones de contingencia.

Durante las pruebas, se observó que el hook podría tardar cinco minutos o más en detectar un fallo. Sin embargo, una vez detectado el fallo, la migración de las máquinas virtuales afectadas a otros nodos se ejecutó de manera eficaz. Esta observación será explorada con más detalle en la sección de pruebas del documento.

Listado del hook, los nodos KVM (hosts) y algunas máquinas virtuales

La Imagen 3 muestra una captura de terminal con varios elementos clave que confirman el correcto funcionamiento del clúster OpenNebula. En ella, se visualiza el hook activado, la lista de hosts que corresponden a los nodos OpenNebula KVM del clúster, y un listado de las máquinas virtuales en operación.



```
root@spock: /home/ubuntu
oneadmin@spock:~$ onehook list
  ID NAME                TYPE
  -- --                -
  1  host_error          state
oneadmin@spock:~$ onehost list
  ID NAME        CLUSTER  TVM  ALLOCATED CPU  ALLOCATED MEM  STAT
  -- --        -
  2  janeway      default  2    800 / 1600 (50%)  8G / 251.2G (3%) on
  1  data         default  3    900 / 2400 (37%)  8.5G / 251.2G (3%) on
  0  riker        default  3   1000 / 2400 (41%)  10G / 47.1G (21%) on
oneadmin@spock:~$ onevm list
  ID USER  GROUP  NAME           STAT  CPU  MEM  HOST  TIME
  -- --   -
  40 oneadmin oneadmin unodo3-lxd     runn  4    4G  riker  9d 14h08
  39 oneadmin oneadmin unodo2-lxd     runn  4    4G  data   9d 14h08
  38 oneadmin oneadmin unodo1-lxd     runn  4    4G  janeway 9d 14h08
  37 oneadmin oneadmin nodo3-lxd     runn  4    4G  riker  11d 09h55
  36 oneadmin oneadmin nodo2-lxd     runn  4    4G  data   11d 09h55
  35 oneadmin oneadmin nodo1-lxd     runn  4    4G  janeway 11d 09h55
  22 oneadmin oneadmin kawin      runn  2    2G  riker  13d 00h44
  18 oneadmin oneadmin vr-vrouter-0 runn  1   512M data  13d 03h33
oneadmin@spock:~$
```

Imagen 3: Hook, hosts y máquinas virtuales.

2.4 Implementación, capa de Plataforma como Servicio

La configuración de plataforma presentada a continuación es el resultado de numerosas pruebas que convergieron hacia la obtención de una alta disponibilidad y un rendimiento óptimo. Aunque en la propuesta de diseño se consideraron diversas arquitecturas de referencia, la implementación final se desarrolló a partir de una arquitectura de alta disponibilidad especialmente adaptada para un servicio que, inicialmente, no estaba concebido para este propósito. Sin embargo, se ha ajustado progresivamente para integrar las nuevas tecnologías y satisfacer las exigencias actuales.

Incus como Clúster de contenedores

El proceso de implementación de la capa de contención comenzó con los primeros prototipos en un clúster LXD. La facilidad de provisionamiento proporcionada por OpenNebula facilitó el despliegue paralelo de las tres máquinas virtuales necesarias para establecer un clúster de contenedores, una tarea que se repitió varias veces en la búsqueda de la estructura óptima de aislamiento a nivel de red. Tras una semana de pruebas, surgieron contratiempos debido a un fallo en el repositorio de imágenes de LinuxContainer.org, que ya no ofrecía imágenes compatibles con LXD. Este problema se solucionó eliminando el repositorio de LinuxContainer.org de Debian 12 en la máquina virtual y reemplazándolo por el repositorio oficial de Canonical. A raíz de este incidente, se optó por utilizar Incus en las pruebas subsiguientes, lo que aseguró la continuidad y la calidad de las fuentes de contenedores basados en LinuxContainer.

Maquinas virtuales

Se seleccionó Moodle como el servicio de prueba base para la estructura de alta disponibilidad. Para ello, se configuraron tres máquinas virtuales dentro de un clúster de contenedores Incus, siguiendo el diseño propuesto inicialmente. Aunque el diseño original ubicaba los proxies en la capa superior, fue necesario agregar dos máquinas virtuales adicionales equipadas con Linux Virtual Server (LVS) y Keepalived, configuradas en un esquema LVS NAT con roles de primario y secundario. Este arreglo garantiza la disponibilidad continua de los proxies, además de proporcionar un balanceo de carga efectivo.

Como primer paso, se desplegaron tres máquinas virtuales utilizando OpenNebula Sunstone. La asignación de recursos para cada máquina virtual fue la siguiente:

- 16 GB de RAM
- 1 Core físico configurado al 50% de su capacidad
- 4 vCPUs
- 50 GB de disco duro
- Sistema Operativo: Debian 12.

Descripción de la capa PaaS

La arquitectura de alta disponibilidad de Moodle, como se muestra en la Figura 6, se describe desde un enfoque 'de arriba hacia abajo'. Comenzando en la parte superior, el acceso al sistema inicia con el LVS-NAT con Keepalived, que actúa como el primer punto de contacto para las solicitudes de Internet, ofreciendo un balanceo de carga efectivo y asegurando alta disponibilidad. Seguidamente, los proxies gestionan la distribución y optimización de las solicitudes antes de que estas lleguen al servidor web. Este servidor, configurado con Nginx y PHP-FPM, procesa las solicitudes; aquí se encuentran el `rootdir` de Moodle, que aloja la interfaz web, y el `moodledata` (dataroot), directorio cuya función es el almacenamiento de archivos de datos y de usuarios. En la base de la arquitectura se encuentra el almacenamiento gestionado por Galera MariaDB, que garantiza la persistencia y replicación de los datos, esenciales para la integridad y el funcionamiento continuo de Moodle.

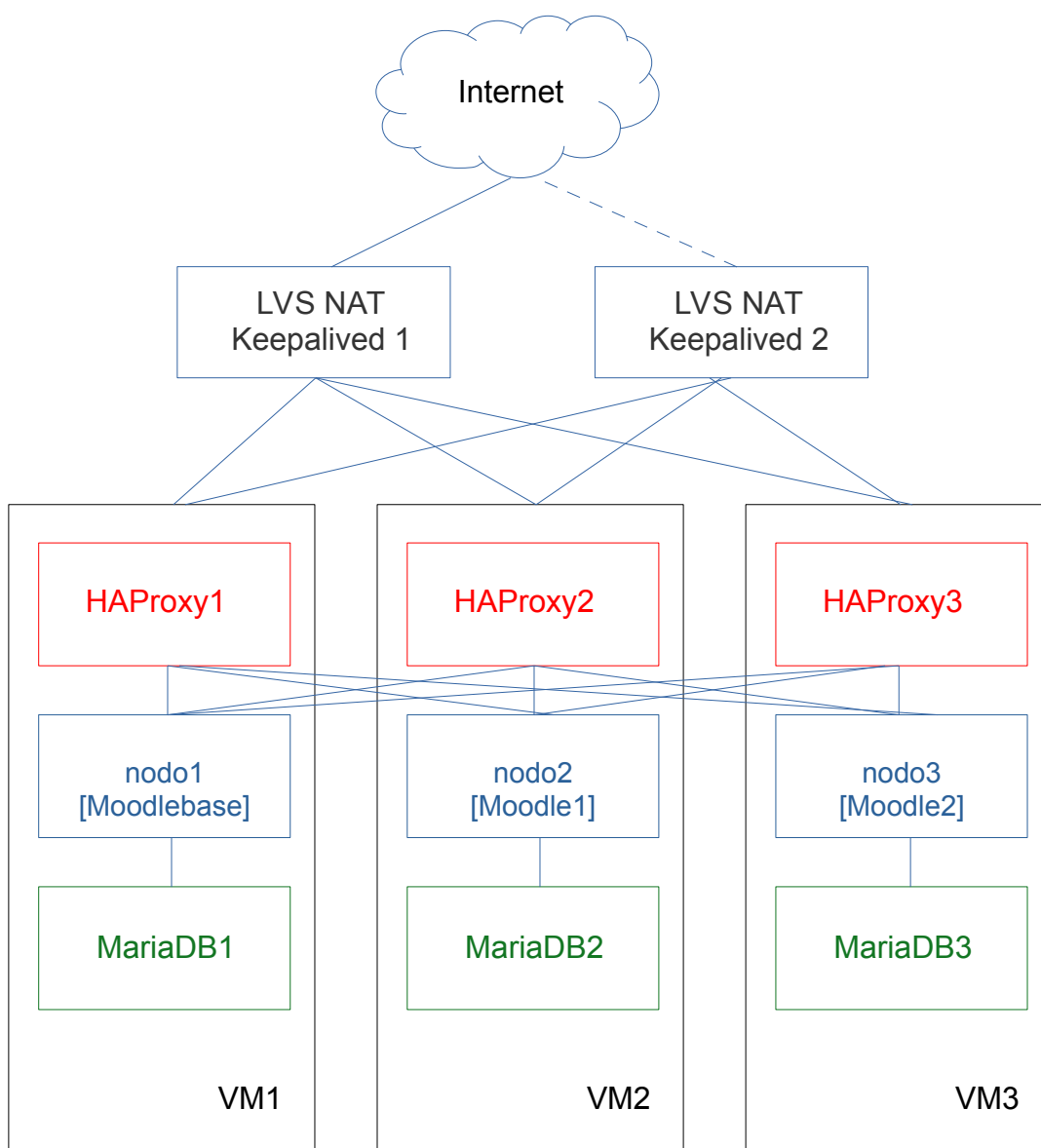


Figura 6: Estructura de implementación de Moodle en alta disponibilidad.

LVS-NAT con Keepalived

LVS-NAT, en conjunto con Keepalived, está configurado en dos máquinas virtuales, cada una con una interfaz de red conectada a una VLAN pública y otra interfaz conectada a una red privada, reservada exclusivamente para máquinas virtuales y/o contenedores. La función principal de LVS-NAT es la traducción de direcciones de red a nivel de transporte, restringiendo el tráfico a los protocolos HTTP y HTTPS, correspondientes a los puertos 80 y 443, respectivamente. Esta traducción se direcciona hacia los tres servidores HAProxy en el nivel inferior (Figura 6). Por su parte, Keepalived asegura la disponibilidad continua del servicio a través de la asignación de una dirección IP virtual, gestionada por un nodo maestro y un nodo de respaldo, que asume el rol de maestro en caso de fallo del primero. Además, LVS-NAT distribuye la carga de manera equitativa entre los tres proxies. El archivo de configuración de LVS-NAT con Keepalived está documentado en el Anexo E.

HAProxy

En la arquitectura de alta disponibilidad de Moodle, HAProxy cumple dos roles fundamentales: primero, en términos de disponibilidad, proporciona un balanceo de carga eficiente para el acceso a cada una de las instancias de los contenedores de Moodle, asegurando una distribución equitativa de las solicitudes. Segundo, en el ámbito de la seguridad, actúa como un filtro efectivo para prevenir ataques de Denegación de Servicio (DDoS). La configuración de HAProxy se encuentran en el Anexo F.

Moodle

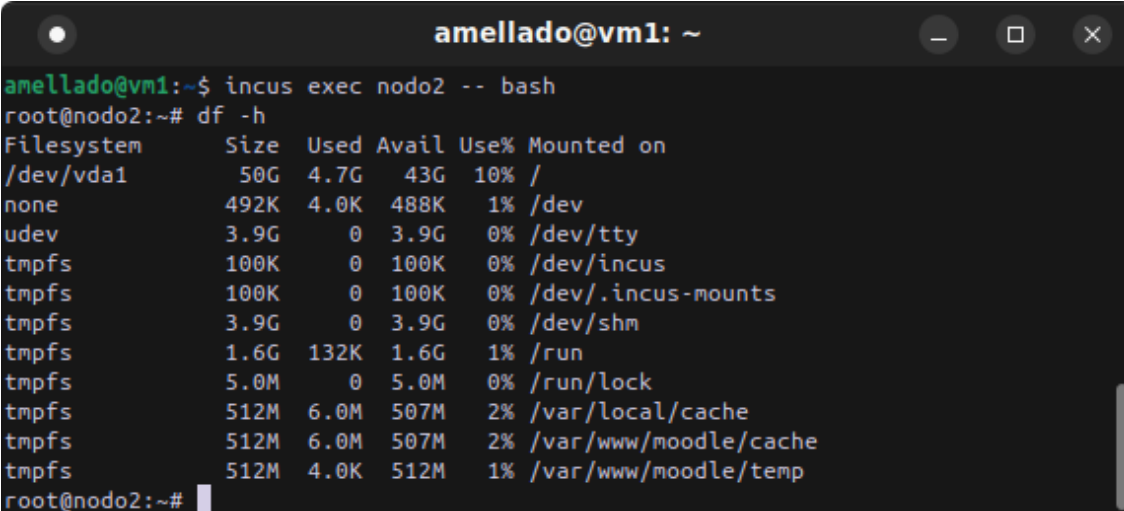
El proceso de implementación del contenedor Moodle se realizó en varias iteraciones buscando la mejor combinación entre aislamiento, rendimiento, escalado horizontal y consistencia. Había claridad en ubicar Moodle detrás de un Proxy, también el hecho de que un contenedor sería lo más flexible y rápido de instanciar. Pero dada las características estructurales que Moodle requiere, hubo que realizar bastantes ajustes para lograr este objetivo. La primera estructura fue la propuesta en el diseño, que solo consideraba el directorio compartido moodledata en cada uno de los contenedores instanciados. Debido a que el almacenamiento de alta disponibilidad utilizado es Ceph, se consideró como opción usar un directorio compartido en Ceph. Por tanto, el directorio moodledata debía ser montado asociado a un sistema de archivos CephFS, inicialmente este montaje se logró con Ceph-Fuse, pero fue descartado porque no permitía un aislamiento adecuado a nivel de servicios, dejando expuesto todo el sistema de archivos compartido. Como uno de los objetivos estaba centrado en la seguridad, el contenedor Moodle fue aislado en una red exclusiva para contenedores, en este caso, al pasar por un router para llegar a Ceph, el acceso al directorio moodledata no superaba los 36 MB/s (medida con dd escribiendo un archivo de 1GB), retardando los despliegues y dejando la respuesta del servicio muy lenta, generando además, una serie de inconsistencias que no permitían el uso de Moodle en distintos contenedores. Para mejorar el acceso al moodledata se agregó una VLAN adicional a los nodos del clúster Ceph que fuera parte de la VLAN de máquinas virtuales y se

agrego a los contenedores una interfaz correspondiente a esa VLAN, logrando así incrementar la tasa de transferencia a moodledta a 70 MB/s. No obstante, posteriores pruebas de rendimiento, cotejadas con experiencias de implementación en nube pública indicaban que una mejor opción era usar un servicio NFS. Como la plataforma de almacenamiento es Ceph, este servicios se habilito mediante NFS-Ganesha. Esta última opción también fue descartada ya que NFS-Ganesha no permitió el uso del atributo GID del espacio de disco compartido, lo que generó problemas en el momento de recuperar cursos en Moodle. Finalmente se optó por usar un sistema de sincronización de los directorios: web de Moodle, y, el directorio de datos y caché moodledata.

Uno de los desafíos significativos en la implementación de Moodle en un entorno de alta disponibilidad es la gestión del caché. Dado que el ambiente es compartido, el caché debe sincronizarse a través de la red con el repositorio común (moodledata), lo que puede estar limitado por las tasas de transferencia de la red. Inicialmente, se exploró la mejora de esta limitación mediante el uso de Redis-Sentinel. Sin embargo, la solución final adoptada fue la creación de un sistema de caché en RAM, permitiendo la sincronización de los cachés de cada instancia de manera eficiente.

El uso de caché mejoró significativamente el rendimiento de Moodle en entornos de alta disponibilidad. Sin embargo, persistían problemas de inconsistencia, atribuibles a dos factores principales: el manejo de sesiones a través de los proxies y la actualización del repositorio de extensiones de Moodle. Estos problemas se resolvieron incorporando peers en HAProxy y compartiendo el directorio completo del proyecto donde se aloja Moodle.

En la configuración de Moodle dentro de contenedores, se empleó un enfoque que difiere de la práctica común pero ofrece un nivel significativo de aislamiento y organización. Comúnmente, en entornos de contenedores, se utiliza el directorio '/var/www' para alojar sitios web; sin embargo, para esta implementación, se optó por un método que implica la creación de un usuario específico para el sitio. Este usuario llamado "moodle", tiene su directorio "/home/moodle", es creado sin contraseña y está asignado únicamente al grupo www-data, al cual se le otorgan todos los permisos necesarios, proporcionando así un manejo más seguro y aislado del entorno.



```
amellado@vm1: ~
amellado@vm1:~$ incus exec nodo2 -- bash
root@nodo2:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1        50G  4.7G   43G  10% /
none             492K  4.0K  488K   1% /dev
udev            3.9G    0  3.9G   0% /dev/tty
tmpfs           100K    0  100K   0% /dev/incus
tmpfs           100K    0  100K   0% /dev/.incus-mounts
tmpfs           3.9G    0  3.9G   0% /dev/shm
tmpfs           1.6G  132K   1.6G   1% /run
tmpfs           5.0M    0   5.0M   0% /run/lock
tmpfs           512M  6.0M  507M   2% /var/local/cache
tmpfs           512M  6.0M  507M   2% /var/www/moodle/cache
tmpfs           512M  4.0K  512M   1% /var/www/moodle/temp
root@nodo2:~#
```

Imagen 4: Caché en RAM del contenedor nodo2.

En un esquema de alta disponibilidad en Moodle las distintas instancias deben contar con entorno que comparta los datos y configuración. Como se ha mencionado, inicialmente se realizaron pruebas usando un directorio común montado con ceph-fuse y/o NFS-Ganesha sobre Ceph. Con estas configuraciones no se obtuvieron resultados satisfactorios, encontrando problemas de consistencia, bloqueos en la escritura de datos y niveles de aislamiento de red. Como el directorio en que residirán los directorios y archivos de Moodle se ubico en '/home/moodle/' (sin privilegios), se optó por usar la herramienta Syncting para sincronizar subdirectorios los esenciales: 'www' para albergar la aplicación web de Moodle y 'moodledata' para los archivos de usuarios, cursos y caché, esta sincronización se realizó en tres instancias de contenedores del servicio Moodle. Este diseño, detallado en la Figura 7, no solo facilita la gestión y el despliegue eficiente de múltiples instancias, sino que también mejora el aislamiento, otorgando seguridad a los datos y la aplicación.

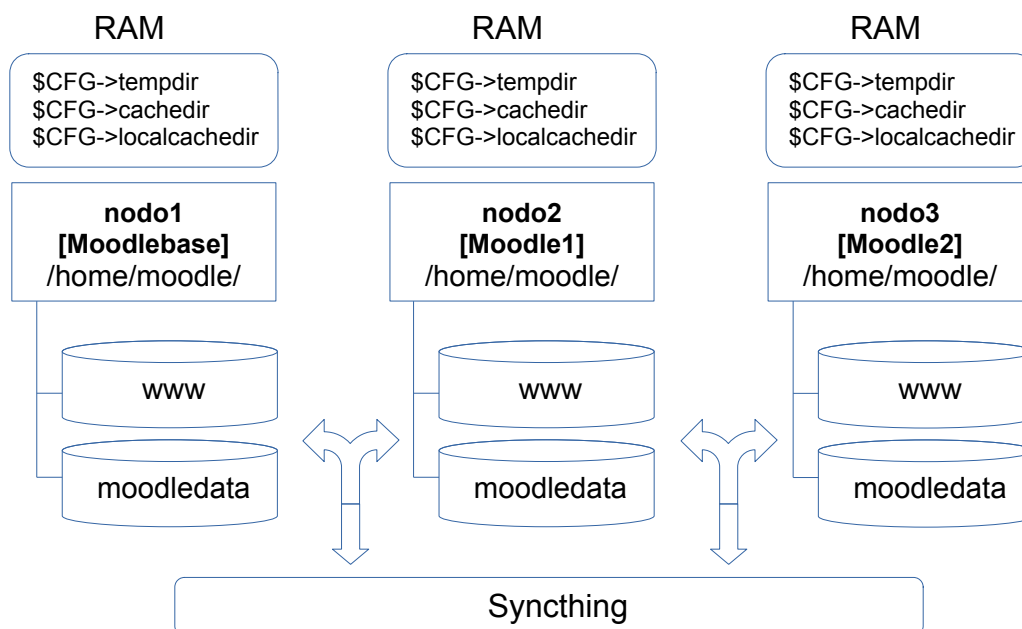


Figura 7: Directorios Moodle

En la parte superior de la Figura 7, se observan las distintas cachés definidas con el propósito de mejorar el rendimiento de Moodle, estableciendo tres niveles de caché: El nivel local, dado por \$CFG->localcache, asignado al directorio /var/local/cache; \$CFG->cachedir, asignado al directorio /var/www/moodle/cache; y \$CFG->tempdir, asignado al directorio /var/www/moodle/temp. Estas configuraciones de caché fueron extraídas del ejemplo de configuración de Moodle, config-dist.php, y están evidenciadas en la captura de terminal de la Imagen 4.

La sincronización de las cachés se realiza mediante un script ejecutado a cada contenedor en diferentes tiempos definidos en el cron del sistema operativo:

Script de sincronización de cachés /usr/local/bin/synccachemoodle.sh :

```
#!/bin/bash

# Directorios a sincronizar
MOODLE_DIR="/home/users/moodle"
DIR_ORIGEN_CACHE="/var/www/moodle/cache"
DIR_ORIGEN_TEMP="/var/www/moodle/temp"
DIR_DESTINO_CACHE="$MOODLE_DIR/moodledata/cache"
DIR_DESTINO_TEMP="$MOODLE_DIR/moodledata/temp"

# Sincronizar directorios bidireccionalmente
unison "$DIR_ORIGEN_CACHE" "$DIR_DESTINO_CACHE" -auto -batch
unison "$DIR_ORIGEN_TEMP" "$DIR_DESTINO_TEMP" -auto -batch
```

Crontab -e, en cada contenedor:

Moodledata:

```
0,7,14,20,28,35,42,48,55 * * * * /usr/local/bin/synccachemoodle.sh
```

Moodle1:

```
2,9,16,22,30,37,44,51,57 * * * * /usr/local/bin/synccachemoodle.sh
```

Moodle2:

```
5,11,19,25,33,40,47,54,59 * * * * /usr/local/bin/synccachemoodle.sh
```

La diferencias en los tiempos de sincronización se realiza para evitar posibles solapamiento en los procesos de lectura y escritura de la caché.

En la parte inferior de la Figura 7 se observa Syncting, aplicación instalada en cada uno de los contenedores de Moodle (nodo1, nodo2 nodo3). Syncting es una aplicación de sincronización continua de archivos que sincroniza archivos entre dos o más ordenadores en tiempo real, de forma segura y protegida [24]. Aunque Syncting no fue diseñada para ambientes de alta disponibilidad, las pruebas realizadas han sido satisfactorias, observando las replicas en los nodos se realizan correctamente.

Galera MariaDB

Al igual que para OpenNebula, para Moodle se implementó un Clúster Galera MariaDB. Dado que se realizaron distintas configuraciones de prueba. Para la comunicación de Galera MariaDB con Moodle, inicialmente se creó una red privada que asegure niveles de aislamiento que otorguen mayores niveles de seguridad a la implementación. Sin embargo, finalmente se optó por acercar Galera MariaDB a los mismos nodos de servicio Web de Moodle para una conexión más segura y directa (Ver Figura 8). La configuración de un Clúster MariaDB esta en detalle en el Anexo C.

Segmentación en redes de aislamiento

En la Figura 8 se presenta la estructura de comunicación entre servicios, organizada en capas de red que establecen distintos niveles de aislamiento. Desde el punto de vista de la seguridad, se asume que cualquier servicio conectado directamente a una red pública (Internet) es potencialmente inseguro. El acceso al servicio realiza mediante la IP virtual definida en LVS (Linux Virtual Service), exponiendo solamente los puertos 80 y 443. Conforme los accesos se alejan o descienden de nivel, la red incrementa progresivamente sus niveles de seguridad, asegurando que las capas más internas estén más protegidas. En consecuencia, la red que establece la conectividad entre el LVS-NAT y el HAProxy se ha representado de color amarillo, dado que, al estar detrás del NAT, incrementa su nivel de seguridad. Por otro lado, la conectividad entre HAProxy y Moodle se ubica en una red exclusiva para contenedores. Finalmente, el clúster Galera MariaDB se desplegó en el mismo contenedor Moodle, la conectividad por tanto se asegura por medio de la loopback (127.0.0.1) que además mejora el nivel acoplamiento evitando los accesos a la red.

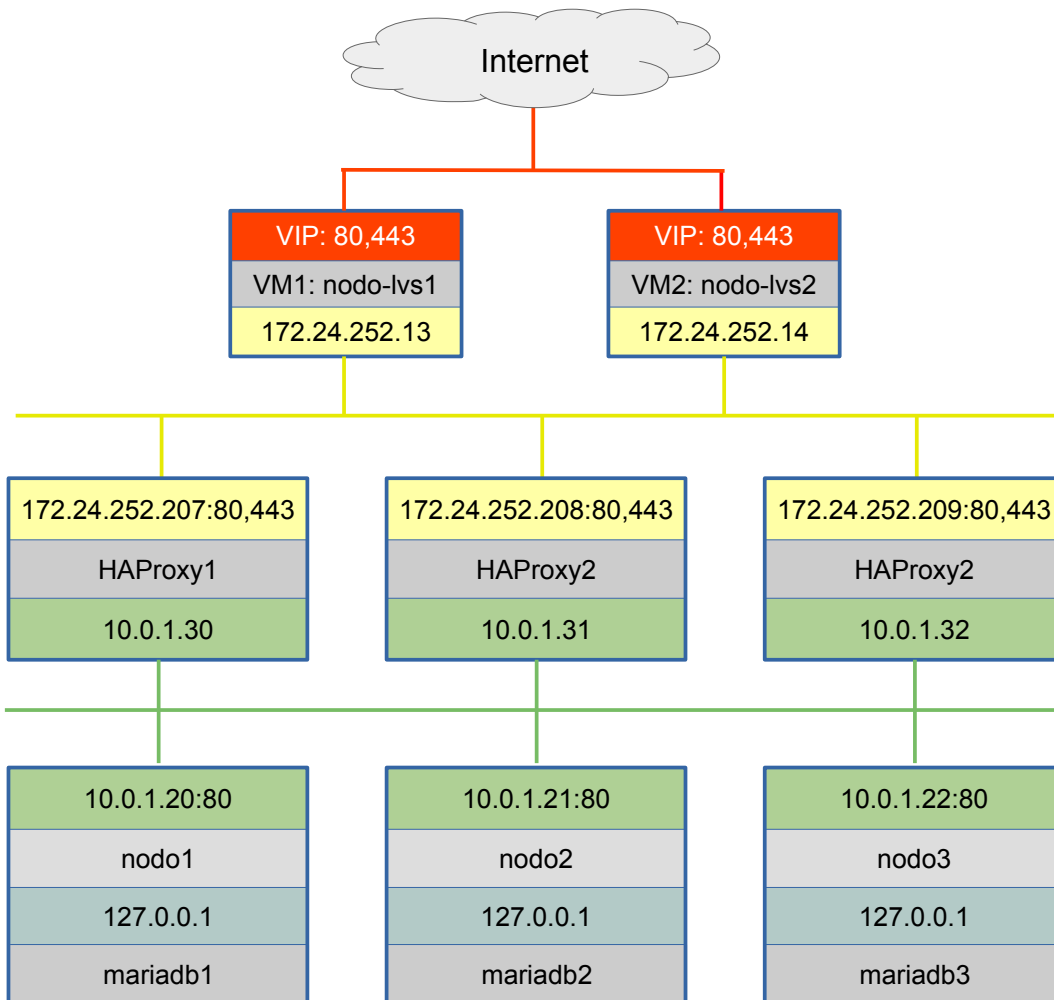


Figura 8: Esquema de red por capas de aislamiento

Para más detalle de la implementación de Moodle HA, ver los scripts de despliegue en el Anexo H.

3. La seguridad en la infraestructura instalada

La seguridad de todo el sistema integrado será abordada desde la perspectiva del triángulo de la ciberseguridad, también conocido como la triada CIA [22]. En este contexto, el término 'sistema' se refiere específicamente a la implementación de OpenNebula como Infraestructura como Servicio (IaaS), el clúster Incus como Plataforma como Servicio (PaaS), y Moodle como un servicio de prueba en un entorno de alta disponibilidad. Esta sección describirá los aspectos de la implementación que están orientados a salvaguardar la confidencialidad, la integridad y la disponibilidad de estos elementos, que incluyen todos los componentes involucrados en el almacenamiento, procesamiento y transmisión de los datos.

Confidencialidad mediante control de acceso y red aislada

En la infraestructura de nivel físico, especialmente en los servidores Bare Metal, se utiliza Canonical Maas para el despliegue de los sistemas operativos base. De manera predeterminada, Maas permite el acceso a los sistemas desplegados solo desde la interfaz de línea de comandos (CLI) alojada en la misma máquina donde reside la aplicación. Aunque el servicio SSH está habilitado en los sistemas operativos de las máquinas físicas desplegadas, el acceso está restringido a la autenticación mediante un par de claves pública-privada, configurado durante el proceso de despliegue. Como se mencionó anteriormente, cada máquina física está equipada con dos interfaces de red: una dedicada a la red de gestión de infraestructura 'física BMaaS' y otra conectada a un puerto troncal del switch de red. Esta configuración permite habilitar las VLANs necesarias desde un nivel superior, según las necesidades de los administradores de sistemas. La red de gestión esta idealmente aislada, con mínima o ninguna interacción con otras redes, y restringida exclusivamente al personal de gestión de infraestructura TI.

Estas medidas de seguridad están diseñadas para proteger contra la exposición no autorizada de datos y sistemas en las capas más bajas de la infraestructura, proporcionando así una base sólida y segura para los niveles superiores.

Asegurando el nivel de almacenamiento compartido

En general, cualquier arquitectura de servicios en la nube o infraestructura de IaaS orientada a la alta disponibilidad requiere de un sistema de almacenamiento compartido [27][28]. Para garantizar la disponibilidad de dicho almacén, el diseño del servicio de almacenamiento debe incorporar explícitamente este requisito. En este proyecto, el clúster de almacenamiento Ceph se ha configurado para reforzar tanto la seguridad como la disponibilidad. Las medidas de seguridad integradas en Ceph, como las llaves de usuario que controlan el acceso a los pools, se complementan con configuraciones adicionales implementadas durante el despliegue para optimizar la protección y el rendimiento.

Como se mencionó en la sección de implementación de infraestructura, el proceso de despliegue del almacenamiento compartido de alta disponibilidad en Ceph incluyó la habilitación del intercambio de claves pública-privada utilizando el algoritmo ed25519, que ofrece mejoras significativas en seguridad mediante el uso de curvas elípticas. Además, es importante destacar que el clúster de almacenamiento hereda las medidas de seguridad descritas anteriormente, ya que se instala en los mismos servidores que actúan como nodos de OpenNebula KVM (ver Figura 5).

Seguridad en OpenNebula

Desde el punto de vista de la disponibilidad, todo el despliegue de OpenNebula fue concebido para operar en alta disponibilidad, las principales características implementadas son: se utiliza una base de datos Galera MariaDB; la interfaz de gestión Web Sunstone está replicada en tres servidores y se accede mediante una IP flotante; el sistema cuenta con tres nodos de OpenNebula KVM y almacenamiento compartido gestionado por Ceph. Además, la red de gestión es exclusiva y los accesos están restringidos mediante pares de llaves pública-privada.

En la configuración de la interfaz de gestión basada en web de OpenNebula, que integra Sunstone con FireEdge, se observan mecanismos de seguridad para prevenir ataques de tipo CSRF (Cross-Site Request Forgery), diseñados para bloquear la suplantación de accesos por sitios previamente autenticados. Sin embargo, el acceso predeterminado tanto a Sunstone como a FireEdge utiliza el protocolo HTTP, lo que no proporciona suficiente confidencialidad en la transmisión de datos.

Para evitar la exposición de la red subyacente, compuesta por la red de gestión de servidores físicos (MaaS) y la red de máquinas virtuales (IaaS), colectivamente denominada RedCore, se ha implementado un aislamiento efectivo. Desde el cortafuegos corporativo (pfsense) se bloquea todo acceso IP no autorizado a RedCore, asegurando su completo aislamiento. En la Imagen 5, correspondiente a reglas de filtrado en un cortafuego PfSense, se observa que desde la red de administradores de sistema, solo se habilita el acceso SSH para efectos de gestión de servidores y todo el tráfico a RedCore queda bloqueado.

<input type="checkbox"/>	✓	0/289 KiB	IPv4 TCP	*	*	RedCore	22 (SSH)	*	none	
<input type="checkbox"/>	✗	0/7.12 MiB	IPv4 TCP	LAN1 subnets	*	RedCore	*	*	none	

Imagen 5: Reglas de filtrado PfSense, acceso desde LAN1.

Para hacer efectivo el aislamiento de la RedCore, en todas las demás interfaces correspondientes a distintas VLANs se aplica el filtro que bloquea el acceso. En la Imagen 6, se muestra que es la primera la regla, precediendo a todas las demás reglas.

Rules (Drag to Change Order)											
<input type="checkbox"/>	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input type="checkbox"/>	✗	0/5 KiB	IPv4*	*	*	RedCore	*		none		

Imagen 6: Regla de filtrado, acceso desde otras interfaces.

Para acceder a Sunstone y FireEdge, se ha implementado un proxy en una máquina virtual con dos interfaces: una dedicada a la red de usuarios (administradores del sistema) y otra a la red de gestión. Este diseño de proxy funciona como una barrera crucial, manteniendo la red de gestión oculta y segura de accesos no autorizados, y garantizando que solo los usuarios autenticados y autorizados puedan interactuar con los sistemas críticos.

A continuación se presenta un extracto esencial de la configuración en el archivo `/etc/haproxy/haproxy.cfg` describiendo del rol de cada línea:

- **'acl host_sunstone hdr(host) -i one.inf.uct.cl'** - Esta línea define el nombre del host que responderá al servicio Web Sunstone.
- **'acl is_csrf token path_reg ^\?csrf token=.*\$'** - Esta regla maneja el acceso al token de autenticación para Fireedge, usado para proveer consola en las máquinas virtuales (NoVNC).
- **'acl is_fireedge path_beg /fireedge'** - Identifica el acceso a la interfaz de gestión de Web Fireedge.
- **'http-request add-header Access-Control-Allow-Origin * if is_csrf token OR is_fireedge'** - Permite todas las fuentes para solicitudes con tokens CSRF o accesos a Fireedge.
- **'http-request set-header Access-Control-Allow-Origin https://one.inf.uct.cl if is_csrf token OR is_fireedge'** - Asegura que solo `https://one.inf.uct.cl` pueda realizar solicitudes CSRF a los servicios de Sunstone y FireEdge.
- **'http-request add-header Access-Control-Allow-Methods "GET, POST, OPTIONS" if is_csrf token OR is_fireedge'** - Especifica los métodos HTTP permitidos para estas solicitudes.
- **'http-request add-header Access-Control-Allow-Headers "Origin, X-Requested-With, Content-Type, Accept, Authorization" if is_csrf token OR is_fireedge'** - Define los encabezados permitidos para las solicitudes CORS.

Adicionalmente, es necesario cambiar la ruta pública a Fireedge para que coincida con el nombre de host del servicio Sunstone definido en HAProxy al final del archivo `/etc/one/sunstone-server.conf` :

```
:public_fireedge_endpoint: https://one.inf.uct.cl
```

Para ver la configuración completa de HAProxy ir al Anexo G.

Segmentación de redes con niveles de seguridad

La Figura 9 ilustra la estructura de la red segmentada por niveles de seguridad. En el nivel superior, representado con color rojo, se encuentra el segmento público, que se asume con una seguridad muy baja o nula. Este segmento incluye el router que conecta a Internet y un cortafuego etiquetado como 'FW'. Además, se encuentran las máquinas virtuales con Linux Virtual Server (LVS), LVS1 y LVS2, cada una con dos interfaces de red. La primera interfaz de cada LVS está conectada directamente al segmento público, mientras que la segunda interfaz conecta estos servidores con los servidores proxy en el nivel intermedio, específicamente haproxy1, haproxy2 y haproxy3. En este esquema, cada servidor proxy cuenta con dos interfaces de red: la primera interfaz de cada proxy está conectada a la red asociada con los LVS, mientras que la segunda interfaz se conecta a un nivel inferior que no tiene conectividad directa a Internet. El gateway para estos servidores proxy es la IP virtual proporcionada por la configuración de LVS, y su conectividad está limitada exclusivamente a atender solicitudes provenientes de los LVS.

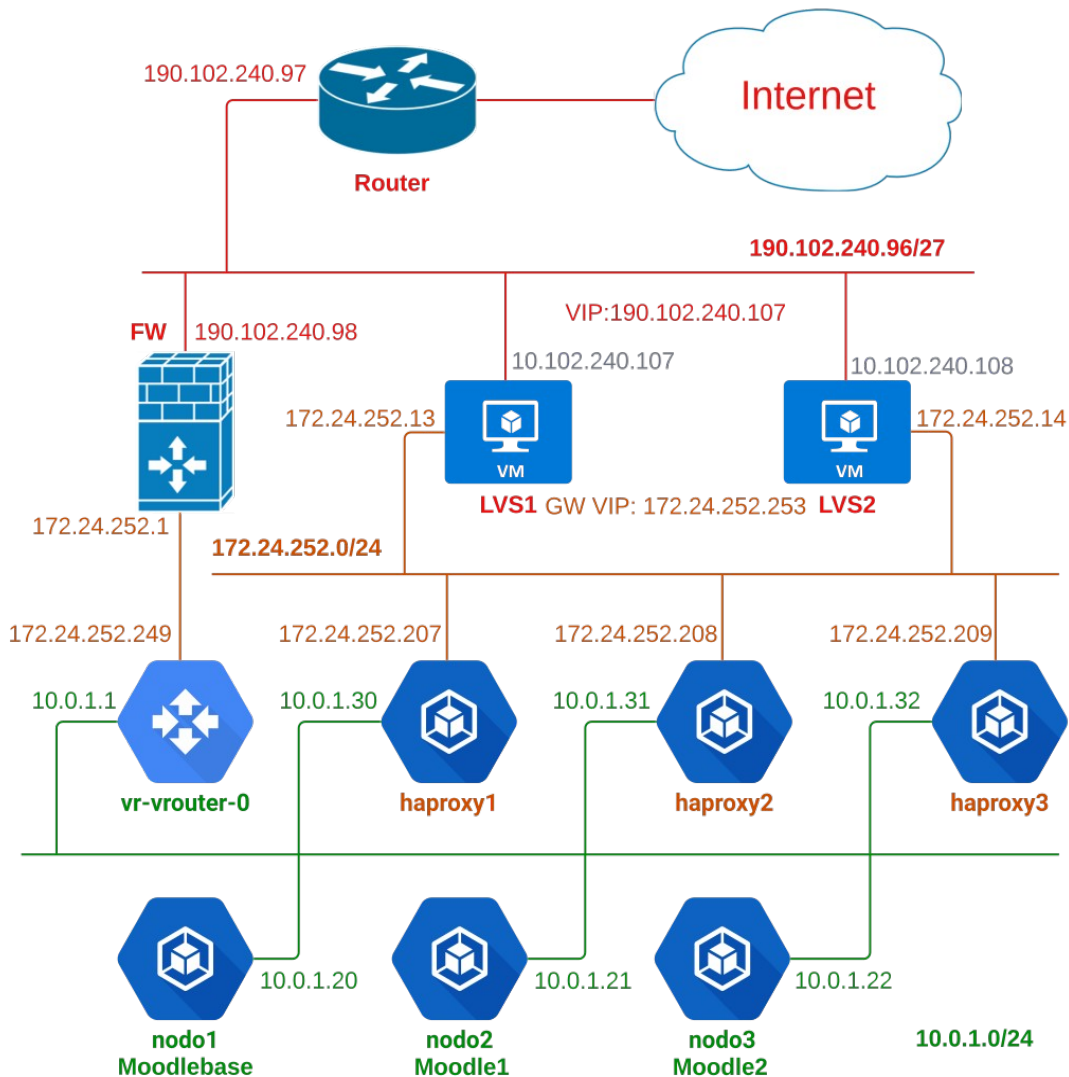


Figura 9: Arquitectura de redes aisladas.

Figura 9. Arquitectura de redes aisladas

La capa más baja, representada en verde, está compuesta por contenedores, con la excepción de su respectivo Gateway, implementado mediante un router virtual de OpenNebula denominado vr-vrouter-0. Este router realiza una función de NAT, actuando como filtro de tráfico hacia y desde la red de contenedores. Al enmascarar las direcciones IP internas, el vr-vrouter-0 facilita la conectividad hacia una red externa de manera transparente, permitiendo el flujo de tráfico necesario mientras mantiene la privacidad y seguridad de las redes internas. Los contenedores conectados a esta red incluyen los servidores proxies con su segunda interfaz y los nodos de servicios de Moodle.

Adicionalmente, en vr-vroute-0 se establecieron reglas de traducción de puertos para alcanzar y/o gestionar la GUI de cada servicio Synchting en los nodos del servicio Moodle. En la Figura 10 puede apreciar el acceso a la gestión de Synchting que se establece desde un red autorizada.

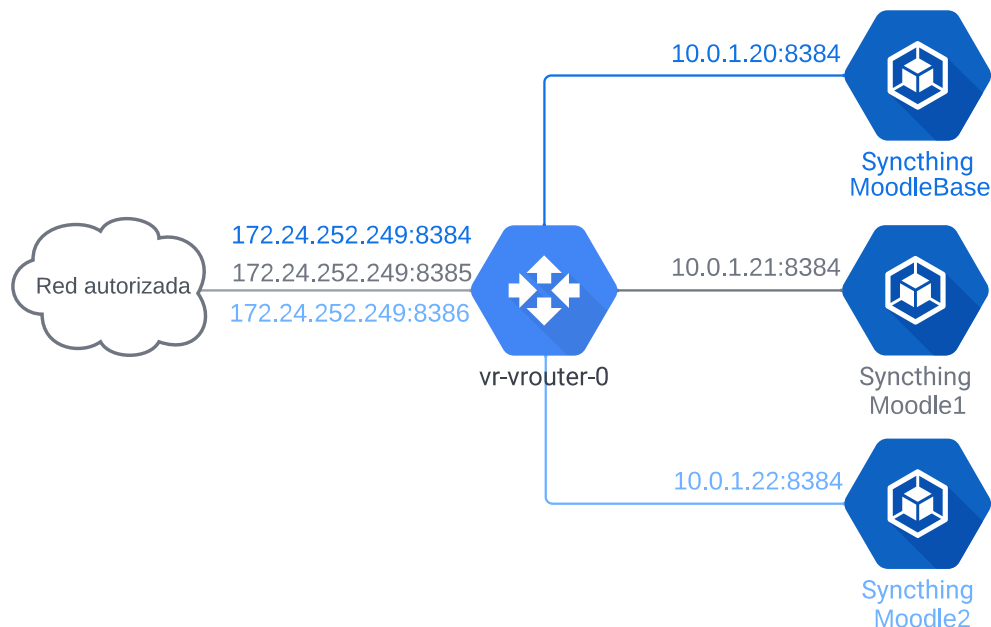


Figura 10: Acceso a la GUI de Synchting desde una red autorizada.

La reglas aplicadas en el Router vr-vrouter-0 para realizar DNAT son:

```
$ sudo iptables -t nat -A PREROUTING -p tcp -s 192.168.2.0/24 \  
-d 172.24.252.249 --dport 8384 -j DNAT --to-destination 10.0.1.20:8384  
$ sudo iptables -t nat -A PREROUTING -p tcp -s 192.168.2.0/24 \  
-d 172.24.252.249 --dport 8385 -j DNAT --to-destination 10.0.1.21:8384  
$ sudo iptables -t nat -A PREROUTING -p tcp -s 192.168.2.0/24 \  
-d 172.24.252.249 --dport 8386 -j DNAT --to-destination 10.0.1.22:8384
```


En donde:

- El parámetro “-t nat” indica que la regla se aplica a la tabla NAT.
- La cadena PREROUTING se utiliza para modificar los datagramas IP entrantes, traduciendo las direcciones de destino de paquetes que provienen de Internet a direcciones dentro de la red interna
- El argumento “-p tcp” especifica que la regla se aplica únicamente al protocolo TCP. La especificación “-s 192.168.2.0/24” identifica la red autorizada desde la cual se pueden originar los paquetes, mientras que “-d 172.24.252.249” designa la dirección de destino expuesta externamente, con el puerto de destino especificado por “--dport 8384”. Esta configuración se utiliza para controlar y limitar el acceso a un recurso específico que, aunque expuesto, necesita una gestión cuidadosa del tráfico entrante.
- La acción “-j DNAT --to-destination 10.0.1.20:8384” redirige los paquetes a una dirección específica dentro de la red aislada. Esta parte de la regla asegura que todo el tráfico destinado al servicio expuesto sea correctamente encaminado hacia el servidor interno adecuado, preservando así la seguridad de la red interna y garantizando que sólo el tráfico legítimo y autorizado pueda alcanzar el recurso crítico."

Posteriormente se encontró que no era adecuado dejar esta red expuesta, por tanto el acceso migró al uso de proxy (HAProxy) como intermediario, los permisos de accesos se otorgaron solo a las IP de los administradores de sistema.

Integridad y disponibilidad a nivel de clúster de base datos

El uso de un clúster de bases de datos como Galera MariaDB introduce características clave que mejoran tanto la continuidad operativa como la integridad de los datos:

Integridad: Galera MariaDB implementa la replicación sincrónica, donde los datos se replican en todos los nodos simultáneamente. Esto requiere que cada transacción sea confirmada por todos los nodos antes de considerarse completa. Además, Galera incluye mecanismos de prevención de conflictos de escritura que certifican cada transacción contra las transacciones concurrentes antes de su aplicación. Estas medidas eliminan las discrepancias entre nodos y aseguran que las operaciones de escritura no interfieran unas con otras, preservando así la integridad de los datos.

Disponibilidad: El clúster de Galera destaca por su alta disponibilidad, gracias a su capacidad para gestionar fallos de nodos eficazmente. Si un nodo falla, los otros nodos del clúster continúan operando sin interrupciones. Cuando un nodo afectado se recupera, puede reintegrarse al clúster y sincronizarse automáticamente con los demás nodos, restableciendo su estado para mantener la coherencia de los datos. Además, el clúster facilita la adición y eliminación de nodos sin tiempo de inactividad, lo cual aporta una significativa flexibilidad operativa y minimiza las interrupciones en el servicio.

Galera Clúster es una solución robusta para aplicaciones que necesitan alta disponibilidad y una integridad de datos rigurosa. Su replicación sincrónica y la

capacidad de gestionar fallos de nodos aseguran que las operaciones de base de datos se mantengan continuas y confiables, haciendo de Galera Clúster una elección ideal para entornos donde la precisión de los datos y la minimización de interrupciones son esenciales [25][26].

Niveles de aislamiento mediante el uso de contenedores

Los niveles de aislamiento de servicios en contenedores Incus, o en un clúster Incus, mejoran la confidencialidad y disponibilidad de los datos:

Confidencialidad: Dado que los servicios se implementan de manera dedicada y la administración de cada contenedor se realiza a través del CLI de Incus, no es necesario habilitar servicios como SSH dentro del contenedor. Esto reduce la superficie de ataque y mejora la confidencialidad, ya que el acceso se controla estrictamente a través de comandos ejecutados desde un nodo del clúster. Los contenedores además proporcionan un entorno aislado que limita el acceso a los recursos del sistema operativo subyacente. Si el servicio de un contenedor se viera comprometido, habrían dificultades para acceder a otros contenedores o el host. En específico el uso de los grupos de control (cgroups) de los contenedores en Debian aseguran el aislamiento al separar los recursos de sistema (CPU, Memoria, I/O) entre contenedores.

Disponibilidad: La rápida instanciación de contenedores y la estructura del clúster Incus, que permite migrar contenedores de un nodo a otro con facilidad, aseguran una alta disponibilidad. Esto es crucial para mantener los servicios operativos incluso en caso de fallo de hardware o mantenimiento programado.

Asegurar la disponibilidad mediante Proxies

El uso de proxies en las distintas capas de arquitecturas de alta disponibilidad se presentan más bien como un mecanismo para lograr balanceo de carga y disponibilidad. Como se ha mencionado anteriormente el uso de HAProxy y/o los proxies en general se ven más bien como un dispositivo de software que provee conectividad en la “frontera de dos redes”, exponiendo y haciendo visible solo la componente pública o externa de la red, dejando oculto lo que se desea proteger, atendiendo el principio de confidencialidad, es el caso del HAProxy usado para acceder a OpenNebula Sunstone y Fireedge.

Los proxies proveen además de mecanismos de análisis ya que se encuentran en el medio de una conexión, registrando y contabilizando las conexiones hacia el servicio que se desea proteger. Los problemas más frecuentes encontrados en los sitios Web es el ataque de Denegación de Servicios (DoS) o Denegación de Servicios Distribuidos (DDoS). En el caso del servicio instalado en alta disponibilidad (Moodle), adicionalmente se le ha agregado al HAProxy un sistema de contabilidad de conexiones desde una IP para evitar así posibles ataques DDOS. A continuación un extracto de los configuración para evitar este tipo de ataques describiendo el rol de cada línea:

- **'stick-table type ip size 100k expire 30s store conn_cur,http_req_rate(10s)'**
– Se define una tabla de almacenamiento 'stick-table' de tipo IP, la tabla puede almacenar hasta 100.000 entradas, la entradas expirarán en 30 segundos. La tabla almacena dos valores: el número de actual de conexiones 'conn_cur' y la tasa de solicitudes HTTP en los últimos 10 segundos.
- **'acl too_many_connections sc1_conn_cur gt 50'** – Se define otra ACL llamada 'too_many_conections', en esta ACL se activa el número actual de conexiones 'sc1conn_cur' mayor que 50.
- **'acl too_many_requests sc1_http_req_rate gt 100'** – Se define una llamada 'to_many_request', esta ACL se activa si la tasa de solicitudes HTTP ('sc1_http_req_rate') es mayor que 100 solicitudes en 10 segundos.
- **'tcp-request connection reject if too_many_connections'** – Rechaza las conexión TCP si la ACL 'too_many_connections0' está activa, o sea, si el número actual de conexiones es mayor que 50, la conexión será rechazada.
- **'http-request set-header X-Forwarded-Proto https if { ssl_fc }'** - Añade o establece el encabezado HTTP 'X-Forwarded-For' a 'https' si la conexión está sobre SSL/TLS ('ssl_fc').|
- **'http-request set-header X-Forwarded-For %[src]'** - Añade o establece el encabezado HTTP X-Forwarded-For con la dirección IP del cliente (%[src]).
- **'http-request deny if too_many_requests'** - Rechaza la solicitud HTTP si la ACL 'too_many_requests' está activa, es decir, si la tasa de solicitudes HTTP es mayor que 100 en los últimos 10 segundos, la solicitud será rechazada.

4. Pruebas y Resultados

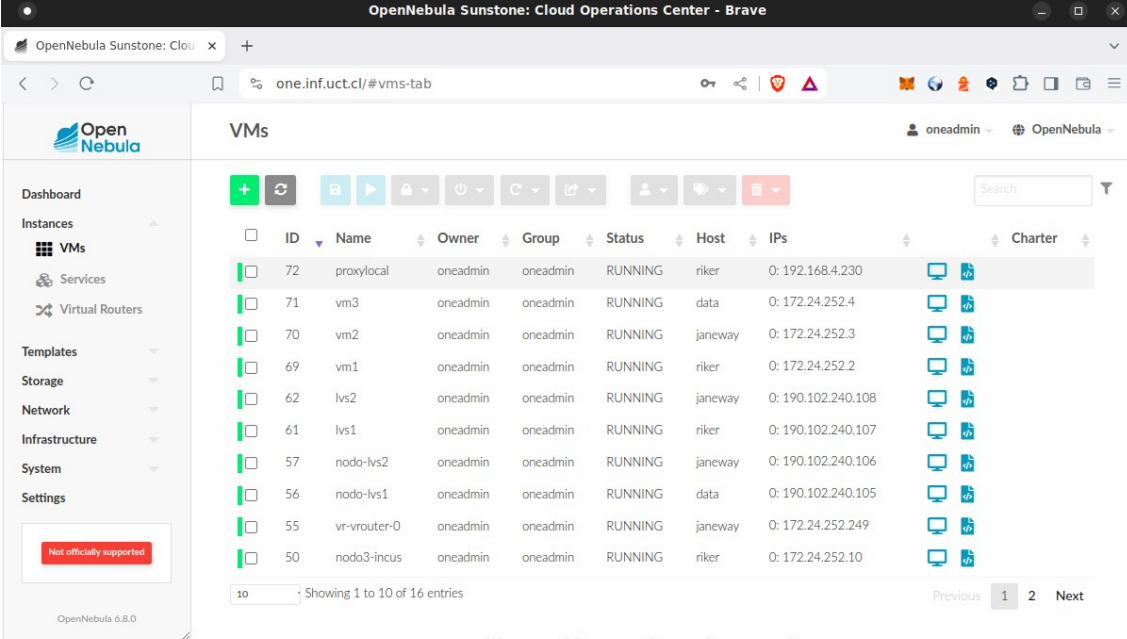
Siguiendo la metodología de desarrollo ágil, las pruebas se realizaron de manera progresiva, coincidiendo con los distintos hitos del proyecto, lo que resultó en una plataforma de alta disponibilidad comprobada a través del servicio Moodle. Es importante destacar que esta plataforma también puede adaptarse a otros tipos de servicios que requieran niveles similares de disponibilidad. Las pruebas se estructuraron en varios frentes:

- **Funcionalidad:** Verificar que el servicio implementado funcione correctamente bajo condiciones normales de uso.
- **Rendimiento:** Asegurar que el servicio implementado ofrezca un rendimiento satisfactorio bajo diferentes cargas de trabajo.
- **Alta Disponibilidad:** Confirmar que el servicio implementado cumpla con los requisitos de alta disponibilidad establecidos.
- **Seguridad:** Comprobar que toda la infraestructura implementada mantenga altos niveles de seguridad en cada una de sus capas y componentes.

Funcionalidad

Las pruebas de funcionalidad comenzaron tras la implementación de OpenNebula, aprovechando que la plataforma Canonical MaaS ya estaba en funcionamiento. Una vez implementado el clúster de OpenNebula, se exploraron las funcionalidades que ofrece esta plataforma, incluyendo: el uso de Ceph como almacenamiento, la creación de redes virtuales, la generación de plantillas para máquinas virtuales, y el análisis del comportamiento de las máquinas virtuales bajo diversas configuraciones de red. Inicialmente, OpenNebula fue probado en un ambiente simulado, lo que facilitó su implementación y posterior operación en un entorno de producción sin mayores inconvenientes.

En la Imagen 7, se observa OpenNebula Sunstone presentado las instancias de las máquinas virtuales utilizadas en las distintas fases de la implementación y pruebas en capa PaaS. En la lista la columna host indica el hipervisor en donde se está ejecutando máquina virtual, en el proceso de prueba se comprobó que las máquinas son creadas de acuerdo al método round-robin que va balanceando la carga entre los distintos hosts.



The screenshot shows the OpenNebula Sunstone interface in a Brave browser window. The page title is "OpenNebula Sunstone: Cloud Operations Center - Brave". The browser address bar shows "one.inf.uct.cl/#vms-tab". The interface includes a sidebar with navigation options: Dashboard, Instances (VMs, Services, Virtual Routers), Templates, Storage, Network, Infrastructure, System, and Settings. A red warning box states "Not officially supported". The main content area displays a table of VMs with columns for ID, Name, Owner, Group, Status, Host, IPs, and Charter. Below the table, it shows "Showing 1 to 10 of 16 entries" and a status summary: "16 TOTAL 13 ACTIVE 3 OFF 0 PENDING 0 FAILED".

ID	Name	Owner	Group	Status	Host	IPs	Charter
72	proxylocal	oneadmin	oneadmin	RUNNING	riker	0: 192.168.4.230	
71	vm3	oneadmin	oneadmin	RUNNING	data	0: 172.24.252.4	
70	vm2	oneadmin	oneadmin	RUNNING	janeway	0: 172.24.252.3	
69	vm1	oneadmin	oneadmin	RUNNING	riker	0: 172.24.252.2	
62	lvs2	oneadmin	oneadmin	RUNNING	janeway	0: 190.102.240.108	
61	lvs1	oneadmin	oneadmin	RUNNING	riker	0: 190.102.240.107	
57	nodo-lvs2	oneadmin	oneadmin	RUNNING	janeway	0: 190.102.240.106	
56	nodo-lvs1	oneadmin	oneadmin	RUNNING	data	0: 190.102.240.105	
55	vr-router-0	oneadmin	oneadmin	RUNNING	janeway	0: 172.24.252.249	
50	nodo3-incus	oneadmin	oneadmin	RUNNING	riker	0: 172.24.252.10	

Imagen 7: Vista de OpenNebula en producción.

La capa de plataforma (PaaS) se construyó inicialmente con un clúster de contenedores LXD, que luego para mayor transparencia y compatibilidad se migró a un clúster Incus. En la Imagen 8, se muestra la lista de contenedores usados para la implementación del servicio de prueba en alta disponibilidad (Moodle). Al igual que OpenNebula, el método de asignación de anfitriones de contención es round-robin, por tanto cada una de las instancias de los proxies vistos y los nodos de instancias de Moodle están balanceados en entre el total de nodos que conforman el clúster.

Una vez probada y validada la funcionalidad de la capa PaaS, se procedió a realizar la implementación del servicio Moodle en alta disponibilidad. Moodle como plataforma no fue construido con tecnologías para este fin, y en el proceso, la estructura fue cambiando progresivamente hasta lograr el objetivo trazado, hecho que se puede constatar en las diferencias progresivas observadas en las secciones de diseño e implementación de este documento. Las instancias de Moodle están implementadas en los contenedores nodo1, nodo2 y nodo3, el acceso a Moodle se realiza mediante los proxies que están constantemente evaluando la vida del servicio Web en donde se aloja y que en caso que un nodo falle no será considerado por el proxy seleccionando otro nodo para la consulta.

```

amellado@vm1: ~
amellado@vm1:~$ incus list
+-----+-----+-----+-----+-----+-----+-----+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS | LOCATION |
+-----+-----+-----+-----+-----+-----+-----+
| haproxy1 | RUNNING | 172.24.252.207 (eth0) | | CONTAINER | 0 | vm1 |
| | | 10.0.1.30 (eth1) | | | | |
+-----+-----+-----+-----+-----+-----+-----+
| haproxy2 | RUNNING | 172.24.252.208 (eth0) | | CONTAINER | 0 | vm2 |
| | | 10.0.1.31 (eth1) | | | | |
+-----+-----+-----+-----+-----+-----+-----+
| haproxy3 | RUNNING | 172.24.252.209 (eth0) | | CONTAINER | 0 | vm3 |
| | | 10.0.1.32 (eth1) | | | | |
+-----+-----+-----+-----+-----+-----+-----+
| nodo1 | RUNNING | 10.0.1.20 (eth0) | | CONTAINER | 0 | vm1 |
+-----+-----+-----+-----+-----+-----+-----+
| nodo2 | RUNNING | 10.0.1.21 (eth0) | | CONTAINER | 0 | vm2 |
+-----+-----+-----+-----+-----+-----+-----+
| nodo3 | RUNNING | 10.0.1.22 (eth0) | | CONTAINER | 0 | vm3 |
+-----+-----+-----+-----+-----+-----+-----+
amellado@vm1:~$

```

Imagen 8: Vista de la lista de contenedores.

Uno de los problemas funcionales de Moodle en alta disponibilidad encontrados, fue la restauración de cursos exportados desde otras implementaciones de Moodle. Problema que fue superado eliminando el directorio común y creando un sistema de sincronización de datos mediante Syncthing. En la Imagen 9 se puede apreciar el funcionamiento de Moodle con dos cursos que fueron restaurados.

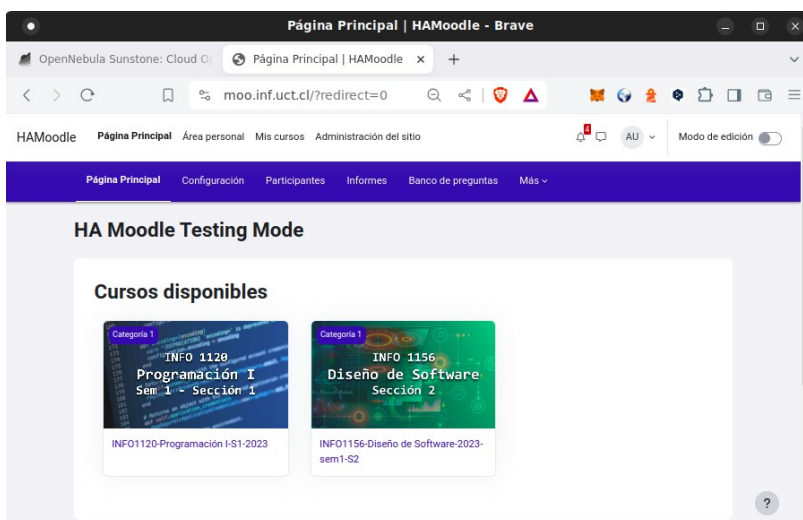


Imagen 9: Vista de dos cursos restaurados.

Rendimiento

El rendimiento fue uno de los problemas que requirió un mayor nivel de atención. Toda arquitectura de clúster compuesta por computadores distribuidos, es un sistemas débilmente acoplado. Por tanto, es evidente un retardo significativo en proceso de lectura/escritura a disco, sobre todo cuando el almacén de OpenNebula está en un clúster Ceph sobre una red de 1 Gbps. En el caso de Moodle, el principal problema radicaba en las caché de disco, que fue superado con el uso de caché en RAM. Igualando o replicando las caché de cada contenedor mediante mediante la sincronización de discos. La Imagen 10 muestra el rendimiento optimizado de Moodle obtenido con el plugin Benchmark.

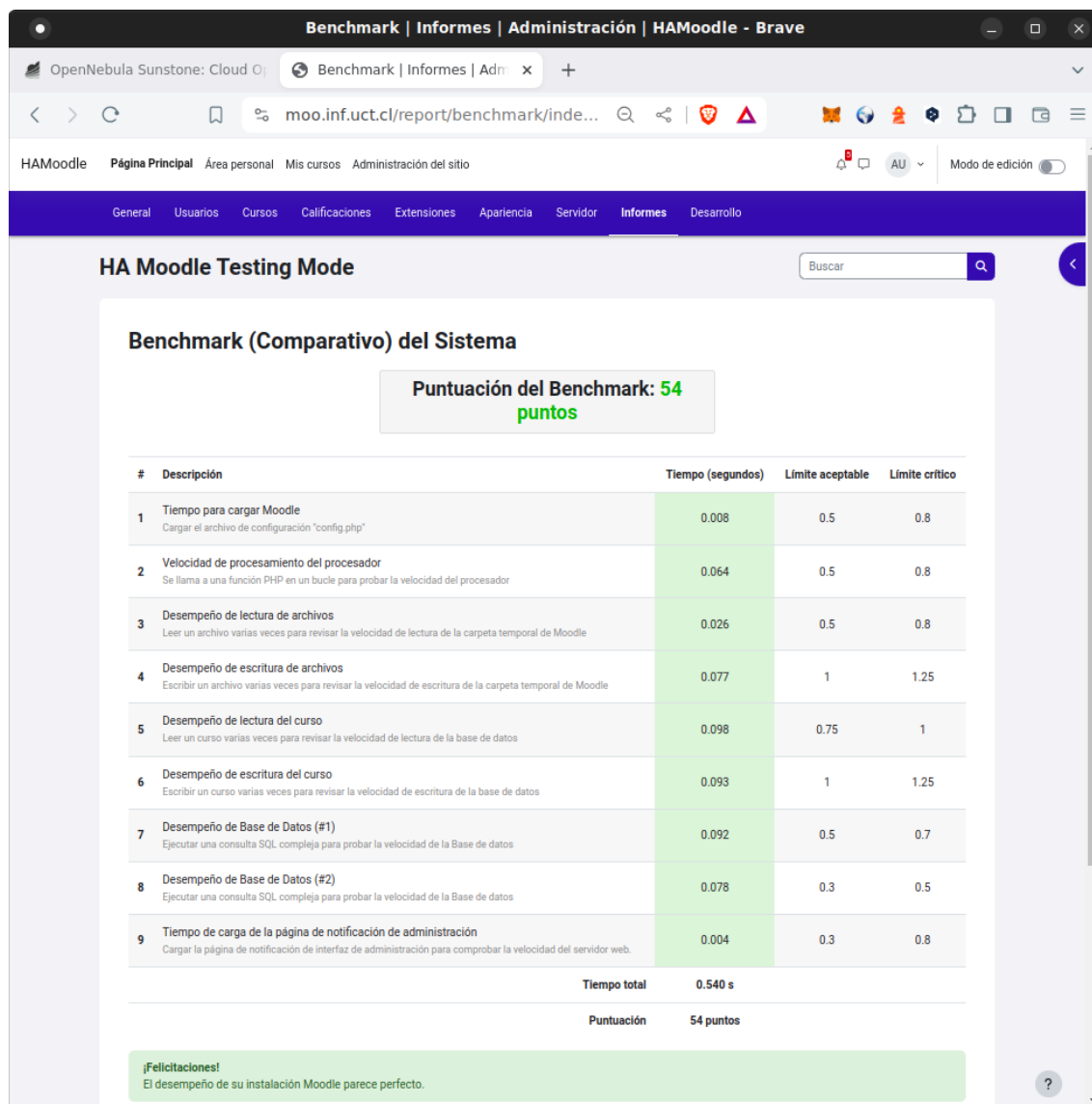


Imagen 10: Resultados del benchmark realizado en Moodle.

En este caso, las últimas pruebas se realizaron después de aplicar reglas en el cortafuego para bloquear todo el tráfico a puertos innecesarios, permitiendo inicialmente el acceso hacia internet a los puertos TCP 21, 22, 53, 80 y 443. Lo que en los dos días siguientes, creo un problema de sincronización en Ceph, que requería sincronizar su clocks con servidores NTP presentes en Internet. Este problema degrado la consistencia de los pools de Ceph y como efecto domino degrado el rendimiento de Moodle. Se habilito por tanto el puerto UDP 123 para permitir desde los

nodos del clúster Ceph acceso a Internet, debiendo resincronizar los monitores de Ceph de forma manual para recuperar su salud (HEALT_OK). Sin embargo, la degradación de Moodle continuaba, detectando que el plugin Benchmark de Moodle, intentaba un acceso UDP al puerto 53 del DNS local. Una vez implementada la regla, se recuperó nuevamente el rendimiento de Moodle observado en la captura de pantalla de la Imagen 10

Alta Disponibilidad

Las pruebas de alta disponibilidad del servicio Moodle debieron ser abordadas de acuerdo a cada una de las capas que conforman el sistema como un todo. El enfoque usado fue desde el nivel de acceso del servicio expuesto en internet hasta el nivel físico de servidores. La primera prueba se realizó apagando la **máquina virtual 'lvs1'** y comprobando que la máquina 'lvs2' asumiera su rol. En la Imagen 11 se pueden apreciar las direcciones IP virtuales pública y privada que asume 'lvs2' para tomar el control del NAT. Siendo la IP 190.102.240.107, la asignada a moo.inf.uct.cl en la zona de dominio.

```

amellado@lvs2: ~
amellado@lvs2:~$ ip addr show eth0 && date
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:00:be:66:f0:6c brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    altname ens3
    inet 10.102.240.108/27 brd 10.102.240.127 scope global eth0
        valid_lft forever preferred_lft forever
    inet 190.102.240.107/27 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::beff:fe66:f06c/64 scope link
        valid_lft forever preferred_lft forever
Fri May 31 14:31:57 UTC 2024
amellado@lvs2:~$ ip addr show eth1
4: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:00:ac:18:fc:0e brd ff:ff:ff:ff:ff:ff
    altname enp0s4
    altname ens4
    inet 172.24.252.14/24 brd 172.24.252.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet 172.24.252.253/24 scope global secondary eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::acff:fe18:fc0e/64 scope link
        valid_lft forever preferred_lft forever
amellado@lvs2:~$

```

Imagen 11: VIP asumidas por keepalived en 'lvs2'.

En la Imagen 12 se muestra la lista máquinas virtuales en donde 'lvs1' está apagada, luego un llamada curl al sitio web de Moodle obteniendo la cookie de sesión.

```

ubuntu@spock: ~
oneadmin@spock:~$ onevm list -f NAME~lvs && date


| ID | USER     | GROUP    | NAME      | STAT | CPU | MEM   | HOST    | TIME      |
|----|----------|----------|-----------|------|-----|-------|---------|-----------|
| 62 | oneadmin | oneadmin | lvs2      | runn | 0.2 | 1024M | janeway | 8d 18h56  |
| 61 | oneadmin | oneadmin | lvs1      | poff | 0.2 | 1024M | riker   | 8d 18h56  |
| 57 | oneadmin | oneadmin | nodo-lvs2 | runn | 0.2 | 1024M | janeway | 24d 11h24 |
| 56 | oneadmin | oneadmin | nodo-lvs1 | runn | 0.2 | 1024M | data    | 24d 11h24 |


Fri May 31 14:32:02 UTC 2024
oneadmin@spock:~$ curl -I -s https://moo.inf.uct.cl | grep "set-cookie" && date
set-cookie: MoodleSession=7lrmrfujskgs3532krfirk1k8f; path=/; secure; HttpOnly
Fri May 31 14:32:05 UTC 2024
oneadmin@spock:~$

```

Imagen 12: Lista de máquinas virtuales LVS y acceso a Moodle.

En la **capa de contenedores (PaaS)** se realizó el mismo procedimiento apagando una máquina virtual y verificando que el servicio Moodle respondiera correctamente. En la Imagen 13 se muestra que todos los contenedores de la máquina 'vm1' están en estado de error.

```

amellado@vm2: ~
amellado@vm2:~$ incus list && date
+-----+-----+-----+-----+-----+-----+-----+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS | LOCATION |
+-----+-----+-----+-----+-----+-----+-----+
| haproxy1 | ERROR | | | CONTAINER | 0 | vm1 |
+-----+-----+-----+-----+-----+-----+-----+
| haproxy2 | RUNNING | 172.24.252.208 (eth0) | | CONTAINER | 0 | vm2 |
| | | 10.0.1.31 (eth1) | | | | |
+-----+-----+-----+-----+-----+-----+-----+
| haproxy3 | RUNNING | 172.24.252.209 (eth0) | | CONTAINER | 0 | vm3 |
| | | 10.0.1.32 (eth1) | | | | |
+-----+-----+-----+-----+-----+-----+-----+
| nodo1 | ERROR | | | CONTAINER | 0 | vm1 |
+-----+-----+-----+-----+-----+-----+-----+
| nodo2 | RUNNING | 10.0.1.21 (eth0) | | CONTAINER | 0 | vm2 |
+-----+-----+-----+-----+-----+-----+-----+
| nodo3 | RUNNING | 10.0.1.22 (eth0) | | CONTAINER | 0 | vm3 |
+-----+-----+-----+-----+-----+-----+-----+
Fri May 31 15:13:09 UTC 2024
amellado@vm2:~$

```

Imagen 13: Lista de contenedores del Clúster Incus.

La Imagen 14. presenta la lista de máquinas virtuales en donde la máquina virtual vm1 está en estado de error, luego una llamada curl al sitio web de Moodle obteniendo la cookie de sesión, comprobando así que el sitio está activo.

```

ubuntu@spock: ~
oneadmin@spock:~$ onevm list -f NAME~vm && date
ID USER GROUP NAME STAT CPU MEM HOST TIME
71 oneadmin oneadmin vm3 runn 0.3 8G data 7d 21h41
70 oneadmin oneadmin vm2 runn 0.3 8G janeway 7d 21h41
69 oneadmin oneadmin vm1 poff 0.3 8G riker 7d 21h41
Fri May 31 15:13:11 UTC 2024
oneadmin@spock:~$ curl -I -s https://moo.inf.uct.cl | grep "set-cookie" && date
set-cookie: MoodleSession=38ho7d272rlkmu0f5t7coo8c6g; path=/; secure; HttpOnly
Fri May 31 15:13:13 UTC 2024
oneadmin@spock:~$

```

Imagen 14: Estado de las máquinas virtuales y acceso a Moodle.

Los comandos de verificación se han combinado con el comando 'date' para reflejar que todas las capturas de pantalla se han realizado en el mismo momento en que se ha probado la continuidad del servicio.

Capa MaaS

En el nivel físico de servidores, desde Canonical MaaS se procedió a apagar el servidor Janeway, esta prueba es más completa que las anteriores ya que la pérdida de una máquina puede afectar a todos los sistemas presentes en la capas o niveles superiores. Esta máquina es parte del clúster Ceph, por tanto, la primera alarma de falla estará dada por una advertencia en clúster de almacenamiento en donde dos de ocho OSDs no estarán presentes. La Imagen 15 muestra una advertencia en el estado de salud del clúster Ceph en donde el servidor Janeway no está presente.


```

root@archier: /home/ubuntu
root@archier:/home/ubuntu# ceph health && date
HEALTH_WARN 1 hosts fail cephadm check; 1/4 mons down, quorum riker,archier,data
Fri May 31 16:37:33 UTC 2024
root@archier:/home/ubuntu#

```

Imagen 15: Advertencia de salud de Ceph con un nodo apagado.

En la Imagen 16, se observa que los contenedores haproxy2 y nodo2 se ejecutan en la máquina virtual 'vm2' están en estado de error.

```

amellado@vm1: ~
amellado@vm1:~$ incus list && date
+-----+-----+-----+-----+-----+-----+-----+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS | LOCATION |
+-----+-----+-----+-----+-----+-----+-----+
| haproxy1 | RUNNING | 172.24.252.207 (eth0) | | CONTAINER | 0 | vm1 |
| | | 10.0.1.30 (eth1) | | | | |
+-----+-----+-----+-----+-----+-----+-----+
| haproxy2 | ERROR | | | CONTAINER | 0 | vm2 |
+-----+-----+-----+-----+-----+-----+-----+
| haproxy3 | RUNNING | 172.24.252.209 (eth0) | | CONTAINER | 0 | vm3 |
| | | 10.0.1.32 (eth1) | | | | |
+-----+-----+-----+-----+-----+-----+-----+
| nodo1 | RUNNING | 10.0.1.20 (eth0) | | CONTAINER | 0 | vm1 |
+-----+-----+-----+-----+-----+-----+-----+
| nodo2 | ERROR | | | CONTAINER | 0 | vm2 |
+-----+-----+-----+-----+-----+-----+-----+
| nodo3 | RUNNING | 10.0.1.22 (eth0) | | CONTAINER | 0 | vm3 |
+-----+-----+-----+-----+-----+-----+-----+
Fri May 31 16:37:34 UTC 2024
amellado@vm1:~$

```

Imagen 16: Contenedores del Clúster Incus.

La Imagen 17 muestra secuencialmente el estado de los hosts de OpenNebula, en donde se observa que el servidor Janeway está en estado de error, lo que por efecto domino crea un fallo en la máquina virtual 'vm2', que a su vez presenta un error en los contenedores haproxy2 y nodo2, vistos en la Imagen 16, en el último comando se verifica al acceso a la cookie de sesión de Moodle como activo, comprobando la alta disponibilidad de todo el sistema.

```

ubuntu@spock: ~
oneadmin@spock:~$ onehost list && date
ID NAME CLUSTER TVM ALLOCATED_CPU ALLOCATED_MEM STAT
2 janeway default 0 0 / 1600 (0%) 0K / 251.2G (0%) err
1 data default 9 500 / 2400 (20%) 60G / 251.2G (23%) on
0 riker default 7 290 / 2400 (12%) 31.5G / 47.1G (66%) on
Fri May 31 16:37:37 UTC 2024
oneadmin@spock:~$ onevm list -f NAME~vm && date
ID USER GROUP NAME STAT CPU MEM HOST TIME
71 oneadmin oneadmin vm3 runn 0.3 8G data 7d 23h05
70 oneadmin oneadmin vm2 fail 0.3 8G data 7d 23h05
69 oneadmin oneadmin vm1 runn 0.3 8G riker 7d 23h05
Fri May 31 16:37:41 UTC 2024
oneadmin@spock:~$ curl -I -s https://moo.inf.uct.cl | grep "set-cookie" && date
set-cookie: MoodleSession=lb4842or099jat3gkhna444lib; path=/; secure; HttpOnly
Fri May 31 16:37:44 UTC 2024
oneadmin@spock:~$

```

Imagen 17: Hosts y máquinas virtuales, respuesta del servicio Moodle.

Estas pruebas son concluyentes demostrando la efectividad del servicio e nube de alta disponibilidad.

Cabe señalar que en el proceso de verificación de la continuidad del servicio Moodle, el clúster OpenNebula en el proceso de recuperación de 'vm2' migro la máquina virtual a el hipervisor del host "data". El router virtual vr-vroute-0 quedo en estado de fallo y degradó significativamente el rendimiento de Moodle. Como solución de este problema se concluyo que el vr-router-0 debía también contar con una replica usando la misma técnica de keepalived que en caso de los LVS.

Seguridad

Las pruebas de seguridad se realizan con tres enfoques principales:

- Exploración de la exposición de las direcciones IP de las redes de gestión y máquinas virtuales.
- Asunción de una posible filtración de red IP oculta.
- Exposición en la capa de servicios.

La Imagen 18, corresponde al terminal de un computador en un laboratorio destinado a estudiantes. Esta, muestra la síntesis de varias pruebas realizadas secuencialmente, empezando con la exploración de direcciones IP activas. Utilizando el comando `fping`, se identifican direcciones IP 'vivas' en la red 192.168.6.0/24, detectándose 31 direcciones que responden a paquetes ICMP. Esta prueba demuestra la conectividad existente y la viabilidad de realizar exploraciones a este nivel.

La segunda red a explorar es la de acceso. Aunque un atacante podría acceder directamente a través de la consola, también es posible el acceso desde Internet mediante un servidor SSH público. Este servidor es accesible gracias a un DNAT y es utilizado por profesores y estudiantes. Al estar un usuario autenticado en esta red (192.168.4.0/24 con 19 direcciones IP activas), queda expuesta intencionalmente como parte de una estrategia de riesgo controlado. Sin embargo, esta exposición no compromete la integridad de la RedCore, la cual permanece efectivamente oculta y segura frente a accesos no autorizados.

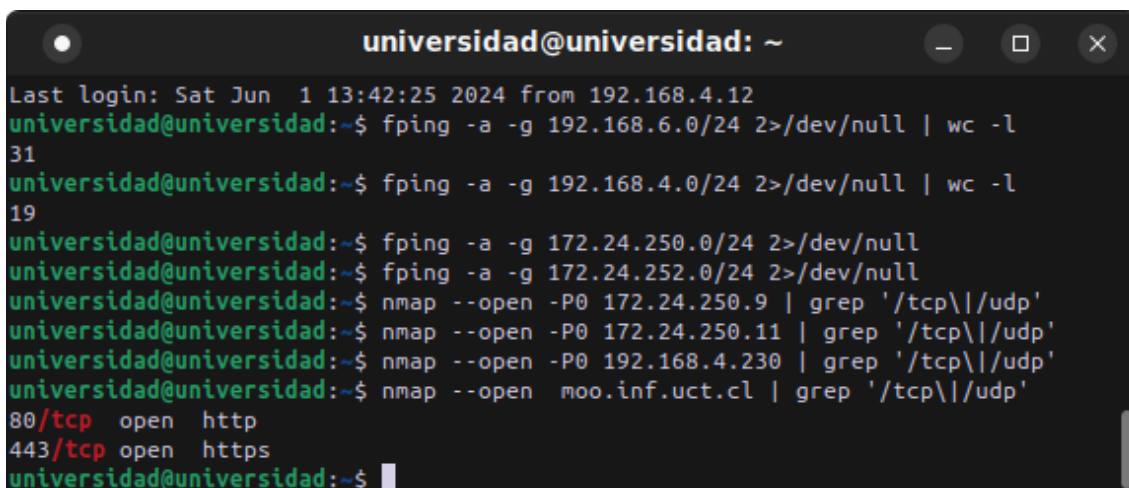
Las exploraciones subsecuentes con `fping` se dirigen a la red denominada RedCore, asignada en el cortafuegos, que comprende los segmentos de gestión y máquinas virtuales, específicamente las subredes 172.24.250.0/24 y 172.24.250.252/24. En ambos casos, no se reciben respuestas a los paquetes echo, lo que indica que estas máquinas están efectivamente ocultas y fuera del alcance de detección directa.

Posteriormente, se procede con pruebas de mapeo de puertos utilizando `nmap` con las opciones `--open` para mostrar únicamente puertos activos y `-P0` para omitir el descubrimiento host previo, dado que el ping está bloqueado. El primer mapeo de puertos se efectúa en la máquina líder que hospeda servicios clave de OpenNebula, incluyendo Sunstone y FireEdge, en los puertos 9869 y 2616 respectivamente. Como se esperaba, no se obtiene respuesta alguna de la IP 172.24.252.9, confirmando la seguridad de la red.

El segundo mapeo se realiza al servidor del dashboard de Ceph, en la IP 172.24.252.1, que también funciona como nodo seguidor del líder de OpenNebula. En este caso, también resulta en ausencia de puertos abiertos visibles de esta IP.

El tercer mapeo se dirige al proxy 192.168.4.230, que facilita la gestión web de diversas plataformas del clúster en la nube, incluyendo MaaS, Ceph, OpenNebula y Synchting, constatando que este acceso está adecuadamente protegido.

Finalmente, se verifica el servicio de Moodle para confirmar su correcta accesibilidad desde Internet, lo que también valida la efectividad de las medidas de seguridad observadas en los mapeos anteriores que no revelaron puertos abiertos.



```
universidad@universidad: ~
Last login: Sat Jun 1 13:42:25 2024 from 192.168.4.12
universidad@universidad:~$ fping -a -g 192.168.6.0/24 2>/dev/null | wc -l
31
universidad@universidad:~$ fping -a -g 192.168.4.0/24 2>/dev/null | wc -l
19
universidad@universidad:~$ fping -a -g 172.24.250.0/24 2>/dev/null
universidad@universidad:~$ fping -a -g 172.24.252.0/24 2>/dev/null
universidad@universidad:~$ nmap --open -P0 172.24.250.9 | grep '/tcp\|/udp'
universidad@universidad:~$ nmap --open -P0 172.24.250.11 | grep '/tcp\|/udp'
universidad@universidad:~$ nmap --open -P0 192.168.4.230 | grep '/tcp\|/udp'
80/tcp open http
443/tcp open https
universidad@universidad:~$
```

Imagen 18: Secuencia de comando de exploración de IPs y puertos.

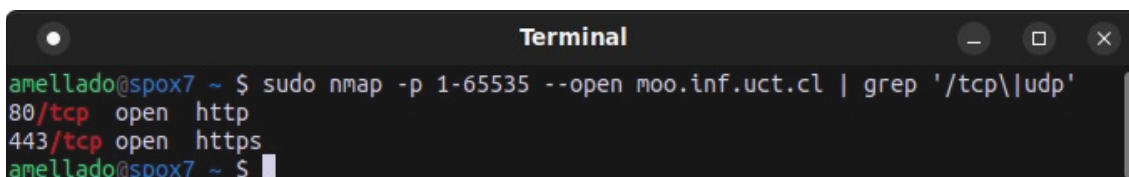
En una segunda prueba, realizada desde la red 192.168.4.0/24 utilizada por los administradores de sistema, los resultados de exploración de direcciones IP vivas en un segmento fueron consistentes con los observados en la red RedCore. La Imagen 19, muestra el resultado de explorar segmentos IP y puertos en el computador de un administrador. En esta exploración, los accesos a la red RedCore están restringidos exclusivamente al servicio SSH. En cuanto al proxy, se detectaron puertos abiertos para los servicios SSH, HTTP (que solo redirige a HTTPS), HTTPS (Sunstone/Fireedge), el puerto 5240 para el servicio MaaS, los puertos 8383, 8385 y 9386 para Synchting, y el puerto 8443 para el dashboard de Ceph. Estos resultados confirman que se logra un nivel de seguridad adecuado sin comprometer el acceso a la gestión de los distintos componentes del clúster de servicios en la nube privada.



```
amellado@epu: ~
amellado@epu:~$ fping -a -g 172.24.250.0/24 2>/dev/null
amellado@epu:~$ fping -a -g 172.24.252.0/24 2>/dev/null
amellado@epu:~$ nmap -p 1-65535 --open 192.168.4.230 | grep '/tcp\|/udp'
22/tcp open ssh
80/tcp open http
443/tcp open https
5240/tcp open unknown
8384/tcp open marathontp
8385/tcp open unknown
8386/tcp open unknown
8443/tcp open https-alt
amellado@epu:~$
```

Imagen 19: Exploración desde la red de administradores.

La última prueba de exposición de puertos se realizó desde un computador conectado a Internet, utilizando Starlink como proveedor de servicios de Internet (ISP). La Imagen 20 corresponde a una exploración exhaustiva de puertos, desde el 1 hasta el 65535. Los resultados muestran que solo están expuestos los puertos correspondientes a los servicios HTTP y HTTPS, es decir, los puertos 80 y 443, respectivamente. Cabe destacar que cualquier petición de conexión al puerto 80 es automáticamente redirigida al puerto 443. Esta configuración asegura que la superficie de ataque se limite exclusivamente al servicio HTTPS.

A terminal window titled "Terminal" with standard macOS window controls. The prompt is "amellado@sbox7 ~". The command entered is "sudo nmap -p 1-65535 --open moo.inf.uct.cl | grep '/tcp\\|udp'". The output shows two lines: "80/tcp open http" and "443/tcp open https". The prompt "amellado@sbox7 ~ \$" is visible at the bottom.

```
amellado@sbox7 ~ $ sudo nmap -p 1-65535 --open moo.inf.uct.cl | grep '/tcp\\|udp'
80/tcp open http
443/tcp open https
amellado@sbox7 ~ $
```

Imagen 20: Exploración de puertos de moo.inf.uct.cl desde Internet.

Es fundamental aclarar que este proyecto se centra en implementar una infraestructura de servicios en nube segura, específicamente en los distintos niveles que conforman las capas de IaaS y PaaS. Aunque esta infraestructura soporta la ejecución de aplicaciones como Moodle, la seguridad de Moodle, en su calidad de aplicación de servicios en nube, se maneja de forma independiente y está fuera del alcance directo de este proyecto. Por tanto, se recomienda gestionar la seguridad a nivel de aplicación mediante estrategias y controles específicos que complementen las medidas de seguridad implementadas en la infraestructura propuesta.

5. Conclusiones y trabajos futuros

El objetivo inicial de este proyecto fue establecer una infraestructura de servicios en la nube que fuera segura, sostenible y económicamente escalable, apta para implementarse en cualquier tipo de empresa u organización. Al concluir el proyecto, es ambiguo determinar si se ha cumplido completamente este objetivo. El hardware utilizado en la plataforma de producción consistió mayoritariamente en servidores reciclados, incluyendo un par de servidores nuevos, el uso de seis servidores en total sugiere que no todas las organizaciones podrían adoptar esta implementación, siendo una barrera de entrada contar con el hardware que soporte la infraestructura propuesta. Aunque las pruebas se realizaron en un ambiente simulado con tres nodos en un computador portátil, serían necesarias más pruebas para obtener conclusiones definitivas.

En cuanto a la escalabilidad, se observa que, desde el punto de vista de los recursos de software y con una configuración adecuada, existen todas las condiciones necesarias para que la infraestructura sea económicamente escalable. Sin embargo, la sostenibilidad presenta un desafío: la implementación de todos los elementos requeridos para la plataforma en la nube conlleva una curva de aprendizaje considerable, incluso cuando se optó por tecnologías que se percibían como más accesibles. Las horas invertidas en este proyecto resultaron ser subestimadas. Para que un proyecto de esta naturaleza sea tanto sostenible como económicamente escalable, es recomendable contar con un equipo de trabajo bien preparado, dado que estas dos variables son interdependientes; no es posible lograr escalabilidad sin sostenibilidad, o viceversa.

Aunque parezca contradictorio, los objetivos parciales se desarrollaron completamente. El primer objetivo, que consistía en el análisis, resultó ser el más fácil de abordar debido a la abundancia y variedad de tecnologías de software libre disponibles para la implementación de infraestructura de nube privada, a pesar de que la documentación no siempre es clara.

El mínimo funcional para la producción de Moodle como servicio no es concluyente, ya que el servicio implementado aún es una prueba de concepto. Se espera que durante el cambio de semestre entre en una fase de marcha blanca para su prueba y eventual puesta en producción. Sin embargo, la implementación realizada con OpenNebula y Ceph revela que son una plataforma lista para producción, donde la alta disponibilidad depende en gran medida de los niveles superiores. Aunque a nivel de IaaS es una plataforma de alta disponibilidad, las limitaciones inherentes a este nivel se hacen evidentes: cuando se implementa alta disponibilidad solo en la capa IaaS, el tiempo de recuperación de una máquina virtual en otro hipervisor es perceptible para el usuario. Esto resalta la importancia de la capa PaaS para mitigar este impacto.

Los aspectos para proveer una plataforma segura se desarrollaron desde el principio del proyecto, incrementando progresivamente su presencia en el diseño y la implementación. En la última etapa del proyecto, las pruebas se dedicaron exclusivamente a identificar y solucionar problemas de seguridad, ajustando la configuración para mitigar los riesgos encontrados.

La metodología ágil fue un factor principal en el desarrollo y logro del proyecto. La cantidad de errores de diseño que se ajustaron durante la implementación se fueron subsanando a medida que se conformaba la infraestructura en cada una de sus capas y niveles. La planificación fue adecuada, ya que se contaba con una idea general de

las fases e hitos que debía seguir el proyecto, aunque en aspectos específicos algunos elementos eran territorio desconocido.

Dado que el proyecto es una implementación de plataforma, no es posible evaluar los impactos éticos-sociales, de sostenibilidad y de diversidad de manera directa. Sin embargo, se estima que el impacto ético-social dependerá en gran medida de factores que van más allá de lo estrictamente técnico y están más relacionados con los valores de quienes toman las decisiones en las organizaciones o empresas que requieren servicios en la nube. El dilema en este caso está en optar por una solución de una gran compañía de servicios de nube pública, que en la mayoría de los casos es extranjera, o por una solución privada (on-premise) o de una pequeña compañía local que puede utilizar la misma implementación desarrollada en este proyecto. Esta última opción acerca la nube al cliente, creando puestos de trabajo para técnicos especializados locales. Además, se considera que una opción local, ya sea privada o pública, puede ser más amigable con el medioambiente si existen leyes locales que protejan el entorno, creando así un mejor equilibrio entre la infraestructura de servicios en la nube y la sostenibilidad.

En el análisis técnico se exploraron distintas alternativas. Como se mencionó en la sección de análisis, se realizó una simulación de las plataformas OpenStack y OpenNebula, optando por esta última debido a su menor curva de aprendizaje. Inicialmente, el uso de OpenNebula fue complicado, especialmente al no lograr implementar Cloud-init, usado principalmente con KVM y OpenStack. Sin embargo, con el tiempo se fue familiarizando con su lógica de funcionamiento y facilidad para configurar máquinas virtuales. Se observó que OpenNebula debería mejorar las opciones de configuración de red de las máquinas virtuales para hacerlas más amigables en el uso de contenedores. Un punto a favor es la facilidad con que las interfaces de red de los nodos KVM interactúan con OpenvSwitch, LinuxBridge y VXLAN, configurándose desde el CLI o Sunstone (esto es mucho mejor que Neutron de OpenStack).

Un aspecto destacado fue la integración de OpenNebula con Ceph como medio de almacenamiento común. Esta configuración es esencial en cualquier sistema de nube de alta disponibilidad o virtualización, permitiendo que las imágenes de discos de las máquinas virtuales sean accesibles desde los hipervisores y posibilitando su migración en vivo. La configuración fue transparente, lo que permitió explorar otros servicios y clientes de Ceph, como Ceph-fuse y NFS-Ganesha. Sin embargo, estos últimos fueron implementados y descartados por no cumplir con los requisitos de seguridad y rendimiento.

En la capa de plataforma (PaaS), la implementación inicial se realizó utilizando un clúster LXD sobre nodos Debian 12. Debido a que linuxcontainers.org eliminó los repositorios de imágenes para LXD, fue necesario cambiar a los repositorios de Canonical. Por este motivo, en la siguiente iteración de implementación, se cambió a un clúster Incus para conservar el soporte de la comunidad LinuxContainers. Este cambio asegura que la plataforma mantenga el soporte y la flexibilidad necesarios para adaptarse a futuros desarrollos.

Todos los clústeres de alta disponibilidad generalmente tienen los mismos requisitos y patrones de estructura, siendo el mínimo de 3 nodos. Tecnologías como Galera MariaDB, la base de OpenNebula con Sunstone, OpenNebula KVM y Ceph, requieren al menos 3 nodos. Las IPs flotantes o virtuales son típicamente configuradas para mantener el acceso a los servicios a través de un canal disponible, y los proxies son fundamentales para balancear la carga. En este contexto, el concepto de 'clúster' es

primario, donde una estructura de alta disponibilidad es esencialmente la suma de múltiples clústeres integrados.

Debido a ciertas particularidades que posee Moodle, se concluye que Moodle no fue construido como un sistema de alta disponibilidad. De hecho, existe escasa información desde la fuente oficial para su implementación con este requisito, limitándose principalmente a recomendaciones de configuración para un ambiente de alta disponibilidad. Por este motivo, Moodle, como caso de prueba, fue el componente que generó más problemas, requiriendo una inversión significativa de tiempo con múltiples enfoques para lograr disponibilidad, funcionalidad y rendimiento.

Cuando Moodle no cuenta con un DNS, el rendimiento se degrada significativamente. Por lo tanto, es necesario usar dos routers virtuales en hipervisores distintos para mantener el rendimiento en caso de fallos. Dado que la red de contenedores está aislada y se gestiona a nivel de virtualización, la máquina virtual que contiene el router puede quedar fuera de servicio si su hipervisor falla.

El enfoque utilizado para abordar la seguridad una vez implementada la plataforma, se desarrolló de acuerdo a la triada de ciberseguridad, considerando la confidencialidad, integridad y disponibilidad en el almacenamiento, procesamiento y transmisión de información. Sin duda, la fortaleza de este proyecto está reflejada en el principio de disponibilidad. No obstante, la confidencialidad se expresa claramente en los niveles de aislamiento proporcionados a las redes y los contenedores, que protegen tanto la plataforma base subyacente como los niveles de virtualización y contención. Los cortafuegos, routers virtuales, proxies, pares de llaves públicas-privadas y la criptografía son componentes esenciales para el resguardo de este principio. Finalmente, aunque el principio de integridad no se observa claramente, el uso de Ceph como primera capa de almacenamiento y Galera MariaDB como clúster de base de datos proporciona los niveles necesarios que ayudan a garantizar la consistencia e integridad de los datos.

Debido a la magnitud y complejidad del proyecto, no fue posible implementar herramientas de monitoreo en esta fase, lo cual se considera una tarea pendiente para trabajos futuros. Tampoco fue posible automatizar los respaldos de las máquinas virtuales, una funcionalidad reservada solo para la edición Enterprise de OpenNebula. Se realizaron pruebas de respaldos manuales extrayendo las imágenes de las máquinas virtuales desde Ceph, en combinación con el uso de plantillas. La automatización de los respaldos quedará como tarea pendiente. En cuanto a la monitorización, se llevaron a cabo evaluaciones preliminares de herramientas destacadas como Prometheus y Grafana para el monitoreo detallado del rendimiento del clúster, así como Wazuh para la supervisión integral de la seguridad. Las limitaciones de tiempo y espacio impidieron su integración en esta etapa, pero su implementación futura promete mejorar significativamente las capacidades de monitoreo y seguridad de la plataforma.

6. Glosario

BMaas (Bare Metal as a Service): Servicio que ofrece acceso directo a hardware físico sin virtualización, ideal para aplicaciones que requieren alto rendimiento.

Backend: Parte de una aplicación que se ejecuta en el servidor, que provee funcionalidad y soporte al frontend, manejando el procesamiento de datos y las interacciones con la base de datos.

Ceph: Sistema de almacenamiento distribuido de alta disponibilidad diseñado para proporcionar un rendimiento, confiabilidad y escalabilidad.

Ceph-fuse: Cliente que permite montar un sistema de archivos Ceph en el espacio de usuario.

CLI (Command Line Interface): Interfaz de línea de comandos que permite a los usuarios interactuar con el sistema operativo o las aplicaciones mediante el ingreso de texto.

Clúster: Conjunto de servidores integrados que actúan como si fueran una única entidad para proporcionar alta disponibilidad y balanceo de carga.

CSRF (Cross-Site Request Forgery): Ataque que engaña al navegador del usuario para que realice acciones no autorizadas en un sitio web en el que está autenticado.

Dashboard: Panel de control que proporciona una visualización gráfica del estado y el rendimiento del sistemas o aplicaciones.

DDoS (Distributed Denial of Service): Ataque que intenta hacer que un servicio o recurso sea inaccesible para los usuarios legítimos saturando el objetivo con tráfico de múltiples fuentes.

DoS (Denial of Service): Ataque que tiene como objetivo interrumpir el servicio de un recurso conectado a internet, haciéndolo inaccesible a los usuarios.

DNAT (Destination Network Address Translation): Técnica de red que se utiliza para redirigir el tráfico entrante a diferentes direcciones IP y puertos.

DNS (Domain Name System): Sistema que traduce nombres de dominio a direcciones IP, permitiendo la localización de servidores y dispositivos dentro de redes globales.

Frontend: Parte de una aplicación que interactúa directamente con el usuario, generalmente compuesta por interfaces visuales por medio de un navegador o la interfaz de un dispositivo.

GUI (Graphical User Interface): Interfaz gráfica de usuario que permite la interacción con dispositivos digitales a través de elementos como ventanas, iconos y menús.

HA (High Availability): Diseño de sistemas que asegura un nivel acordado de operatividad operacional durante un periodo evaluado.

HAProxy: Software de balanceo de carga y proxy para aplicaciones TCP y HTTP que distribuye las solicitudes entrantes a través de múltiples servidores.

HTTP (Hypertext Transfer Protocol): Protocolo utilizado para transferir datos en la World Wide Web.

HTTPS (Hypertext Transfer Protocol Secure): Es una versión segura de HTTP que utiliza cifrado SSL/TLS para proteger la transmisión de datos.

IaaS (Infrastructure as a Service): Modelo de infraestructura como servicio en la nube que proporciona recursos informáticos virtualizados a través de internet.

ICMP (Internet Control Message Protocol): Protocolo utilizado para enviar mensajes de error y operativos indicando, por ejemplo, que un servicio o host solicitado no está disponible o no es accesible.

Incus: Plataforma que facilita la implementación y administración de aplicaciones en contenedores.

IP (Internet Protocol): Protocolo Internet de comunicación principal para la transmisión de paquetes de datos a través de redes de computadoras.

Keepalived: Software de enrutamiento que proporciona servicios simples de alta disponibilidad (HA) para sistemas Linux.

KVM (Kernel-based Virtual Machine): Máquina virtual basada en kernel que provee virtualización completa en sistemas operativos Linux.

LMS (Learning Management System): Software diseñado para crear, gestionar y distribuir contenido educativo digital.

LXC (Linux Containers): Tecnología de virtualización a nivel de sistema operativo que permite la ejecución de múltiples entornos Linux aislados en un solo host de control.

LXD: Gestor de contenedores basado en LXC, proporcionando una experiencia de máquina virtual a través de contenedores Linux.

LVS (Linux Virtual Server): Solución de balanceo de carga de alto rendimiento que opera en el nivel de transporte, permitiendo la distribución eficiente de solicitudes a múltiples servidores backend.

MaaS (Metal as a Service): Servicio que permite tratar la infraestructura física como una nube, aprovisionando y gestionando dinámicamente servidores físicos.

Moodle: Plataforma de aprendizaje en línea, libre y de código abierto, diseñada para proporcionar a los educadores, administradores y estudiantes un sistema integrado, robusto y seguro para crear ambientes de aprendizaje personalizados.

NAT (Network Address Translation): Método que permite modificar las direcciones de red en los encabezados de los paquetes IP mientras están en tránsito a través de un dispositivo de enrutamiento.

NFS (Network File System): Protocolo que permite a un usuario en un cliente acceder a archivos a través de una red de la misma manera que se accede a los discos duros locales.

Nginx: Servidor web que también se utiliza como proxy inverso, balanceador de carga y proxy de correo HTTP.

NTP (Network Time Protocol): Protocolo diseñado para sincronizar los relojes de los sistemas informáticos a través de redes de paquetes de datos con latencia variable.

PaaS (Platform as a Service): Modelo de servicio en la nube que proporciona una plataforma y un entorno para permitir a los desarrolladores construir aplicaciones y servicios a través de internet.

OSD (Object Storage Device): Componente de almacenamiento usado en sistemas distribuidos para manejar datos y metadatos de objetos.

QEMU: Emulador de máquina genérico y de código abierto que permite a los usuarios ejecutar programas de un sistema operativo en otro.

RBD (RADOS Block Device): Sistema de almacenamiento distribuido para bloques que almacena datos en discos virtuales.

SaaS (Software as a Service): Modelo que permite acceder y usar software a través de internet sin necesidad de instalarlo localmente, gestionado por un proveedor externo.

SSH (Secure Shell): Protocolo de red que permite la comunicación segura y cifrada entre dos dispositivos, usado generalmente para terminales remotos y para transferencia de archivos.

TCP (Transmission Control Protocol): Protocolo de control de la transmisión que proporciona entrega ordenada y confiable de datos entre aplicaciones en Internet.

UDP (User Datagram Protocol): Protocolo de datagrama de usuario, permite el envío de mensajes ligeros sin establecer una conexión previa y con menos confiabilidad.

UUID (Universally Unique Identifier): Identificador único universal estándar usado para generar identificadores únicos garantizados en sistemas distribuidos, sin necesidad de un punto central de coordinación.

VLAN (Virtual Local Area Network): Red de área local virtual, tecnología que permite segmentar una red física en múltiples redes lógicas independientes, mejorando la administración y la seguridad.

VXLAN (Virtual Extensible LAN): Virtual LAN extensible, tecnología de red que permite la creación de redes lógicas superpuestas sobre infraestructuras de red física, utilizando encapsulación para escalar y aislar aplicaciones en entornos de nube.

7. Bibliografía

- [0] Este documento ha contado con la asistencia de ChatGPT, un modelo de lenguaje desarrollado por OpenAI, para la revisión y redacción de ciertos fragmentos.
- [1] Effects of cloud market concentration – Carnegie cloud governance toolkit. (s/f). Carnegieendowment.org. Recuperado el 10 de marzo de 2024, de <https://cloud.carnegieendowment.org/cloud-governance-issues/effects-of-cloud-market-concentration/>
- [2] Arthur, W. B. (1996). Increasing Returns and the New World of Business. Harvard Business Review, 74(4), 100-109.
- [3] Commission approves up to €1.2 billion of State aid by seven Member States for an Important Project of Common European Interest in cloud and edge computing technologies. (s/f). Shaping Europe's Digital Future. Recuperado el 7 de marzo de 2024, de <https://digital-strategy.ec.europa.eu/en/news/commission-approves-eu12-billion-state-aid-seven-member-states-important-project-common-european>
- [4] ¿Qué es la metodología Kanban? (2018, octubre 11). Deloitte Spain. <https://www2.deloitte.com/es/es/pages/technology/articles/que-es-metodologia-kanban.html>
- [5] Pursell, S. (2021, diciembre 1). Metodología Agile: qué es y cómo aplicarla a tu proyecto. Hubspot.es. <https://blog.hubspot.es/marketing/metodologia-agile>
- [6] OpenStack, C. L. A. (s/f). *Software*. OpenStack. Recuperado el 11 de marzo de 2024, de <https://www.openstack.org/software/>
- [7] About OpenNebula. (2023, mayo 1). OpenNebula – Open Source Cloud & Edge Computing Platform; OpenNebula (test). <https://opennebula.io/about-opennebula>
- [8] *Apache CloudStack*. (s/f). Apache.org. Recuperado el 11 de marzo de 2024, de <https://cloudstack.apache.org/>
- [9] Bedi, P., Deep, B., Kumar, P., & Sarna, P. (2018). Comparative study of OpenNebula, CloudStack, Eucalyptus and OpenStack. International journal of distributed and cloud computing, 6(01), 37-42.
- [10] Pollock, M. (s/f). Cloudstack vs OpenStack: Which is Right For You? Liquid Web. Recuperado el 9 de abril de 2024, de <https://www.liquidweb.com/kb/cloudstack-vs-openstack/>
- [11] Shankar. (2023, diciembre 10). OpenNebula vs OpenStack: Cloud management platforms. Innovation Insider - My Technology Blog. <https://innovationinsider.info/opennebula-vs-openstack/>

- [12] OpenNebula Front-end HA — OpenNebula 6.8.2 documentation. (s/f). Opennebula.io. Recuperado el 1 de abril de 2024, de https://docs.opennebula.io/6.8/installation_and_configuration/ha/frontend_ha.html
- [13] OpenInfra, C. L. A. (s/f). Install MAAS — charm-deployment-guide 0.0.1.dev519 documentation. Openstack.org. Recuperado el 1 de abril de 2024, de <https://docs.openstack.org/project-deploy-guide/charm-deployment-guide/latest/install-maas.html>
- [14] Sinhoreli, M. (2022, marzo 21). CloudStack vs. OpenStack Comparison - What you need to know before choosing a cloud management system. ShapeBlue; The CloudStack Company. <https://www.shapeblue.com/cloudstack-vs-openstack-comparison/>
- [15] Daffara, C. (2014, octubre 2). Comparing OpenNebula and OpenStack: Two different views on the cloud - OpenNebula – open source cloud & edge computing platform. OpenNebula – Open Source Cloud & Edge Computing Platform. <https://opennebula.io/blog/experiences/comparing-opennebula-and-openstack-two-different-views-on-the-cloud/>
- [16] Installing ceph — ceph documentation. (s/f). Ceph.com. Recuperado el 30 de abril de 2024, de <https://docs.ceph.com/en/latest/install/>
- [17] OpenNebula Front-end HA — OpenNebula 6.8.2 documentation. (s/f). Opennebula.io. Recuperado el 8 de abril de 2024, de https://docs.opennebula.io/6.8/installation_and_configuration/ha/frontend_ha.html
- [18] OpenNebula repositories — OpenNebula 6.8.3 documentation. (s/f). Opennebula.io. Recuperado el 25 de abril de 2024, de https://docs.opennebula.io/6.8/installation_and_configuration/frontend_installation/opennebula_repository_configuration.html
- [19] (S/f-a). Computingforgeeks.com. Recuperado el 30 de abril de 2024, de <https://computingforgeeks.com/install-opennebula-front-end-on-ubuntu/>
- [20] (S/f-b). Computingforgeeks.com. Recuperado el 30 de abril de 2024, de <https://computingforgeeks.com/install-and-configure-opennebula-kvm-node-on-ubuntu/>
- [21] Server clustering improvements proposal - MoodleDocs. (s/f). Moodle.org. Recuperado el 20 de mayo de 2024, de https://docs.moodle.org/dev/Server_clustering_improvements_proposal
- [22] Hashemi-Pour, C., & Chai, W. (2023, diciembre 21). CIA triad (confidentiality, integrity and availability). WhatIs; TechTarget. <https://www.techtarget.com/whatis/definition/Confidentiality-integrity-and-availability-CIA>
- [23] Prakasa, J. E. W., Hanani, A., Hariri, F. R., & Utama, S. N. (2024). Improving Moodle Performance Using HAProxy and MariaDB Galera Cluster. *Journal of Applied Information System and Management*.

- [24] Syncthing. (2019, septiembre 5). Syncthing.net. <https://syncthing.net/>
- [25] Baszczyj, S. (s/f). *MariaDB Galera Cluster guide*. United States. Recuperado el 27 de mayo de 2024, de <https://www.insentragroup.com/us/insights/geek-speak/modern-workplace/a-step-by-step-guide-to-mariadb-galera-cluster-installation-and-configuration/>
- [26] Overview of Galera cluster — Galera cluster documentation. (s/f). Galeracluster.com. Recuperado el 27 de mayo de 2024, de <https://galeracluster.com/library/documentation/overview.html>
- [27] Pacheco, M. (2023, mayo 10). *Private Cloud Architecture: A complete overview*. TierPoint, LLC; TierPoint. <https://www.tierpoint.com/blog/private-cloud-architecture/>
- [28] Perry, Y., & Manager, T. C. (2020, noviembre 18). *High availability cluster: Concepts and architecture*. Netapp.com; Netapp. <https://bluexp.netapp.com/blog/cvo-blg-high-availability-cluster-concepts-and-architecture>
- [29] *Virtual Machines High Availability — OpenNebula 6.8.3 documentation*. (s/f). Opennebula.io. Recuperado el 27 de mayo de 2024, de https://docs.opennebula.io/6.8/installation_and_configuration/ha/vm_ha.html

8. Anexos

Anexo A: Estandarización del despliegue mediante Canonical MaaS

Estructura estándar de almacenamiento y particionado

Como se muestra en la Figura 11, se ha definido una configuración de particionado estándar para cada servidor dentro del clúster. Esta estructura de particionado se diseñó para mitigar las sobrecargas en los sistemas de archivos y para proporcionar una partición de SWAP que mejora el rendimiento de los servidores con menor cantidad de RAM. Además, cuatro de los servidores están equipados con dos discos adicionales cada uno, que se mantienen sin configurar y sin formato. Estos discos se utilizan exclusivamente para el clúster de almacenamiento distribuido con Ceph, lo que permite una gestión de datos más eficiente y robusta al proporcionar un almacenamiento dedicado y escalable para las necesidades del clúster. Este enfoque no solo optimiza el manejo de recursos sino que también contribuye a la estabilidad del sistema al distribuir la carga de forma más equitativa entre las unidades de almacenamiento.

Filesystems

NAME	SIZE	FILESYSTEM	MOUNT POINT	MOUNT OPTIONS	ACTIONS
sda-part1	536.87 MB	fat32	/boot/efi		▼
vgroot-lv0	19.99 GB	ext4	/		▼
vgroot-lv1	15.99 GB	swap	none		▼
vgroot-lv2	49.99 GB	ext4	/var		▼
vgroot-lv3	29.99 GB	ext4	/usr		▼
vgroot-lv4	99.99 GB	ext4	/home		▼
vgroot-lv5	263.01 GB	ext4	/opt		▼

Available disks and partitions

<input type="checkbox"/>	NAME SERIAL	MODEL FIRMWARE	BOOT	SIZE	TYPE NUMA NODE	HEALTH TAGS	ACTIONS
<input type="checkbox"/>	sdb 6d4ae520a...	PERC H700 2.10	×	3.84 TB Free: 3.84 TB	Physical 0	? Unknown ssd	▼
<input type="checkbox"/>	sdc 6d4ae520a...	PERC H700 2.10	×	3.84 TB Free: 3.84 TB	Physical 0	? Unknown ssd	▼

Imagen 21: Vista del Storage estándar en Canonical MaaS (Anexo A).

Estructura estándar de interfaces de red

Con el fin de asegurar una gestión transparente y eficiente, así como el aislamiento necesario en la red, se ha establecido que cada servidor cuente con al menos dos interfaces de red en configuración de puente (bridge):

1. **Interfaz br0**: Configurada como un bridge para la red de gestión, facilitando el control administrativo y el mantenimiento del clúster sin interferir con el tráfico de usuario.
2. **Interfaz br-ex**: Configurada con Open vSwitch, destinada a gestionar las VLAN que se utilizan o pueden ser creadas en OpenNebula, proporcionando así flexibilidad y expansión de red sin comprometer la seguridad ni el rendimiento.

Esta configuración de red no solo promueve la segregación y optimización del tráfico según los requisitos operacionales y de seguridad, sino que también permite una escalabilidad eficiente y controlada dentro del ambiente de nube.

<input type="checkbox"/> NAME MAC	PXE	LINK/INTERFACE SPEED	TYPE NUMA NODE	FABRIC VLAN	SUBNET NAME	IP ADDRESS STATUS	ACTIO...
<input type="checkbox"/> br-ex d4:ae:52:7f:88:ae			Open vSwitch 0	fabric-0 untagged	Unconfigured	Unconfigured	▼
eno2 d4:ae:52:7f:88:ae		1 Gbps/1 Gbps	Open vSwitch 0				
<input type="checkbox"/> br0 d4:ae:52:7f:88:ad	✓		Bridge 0	fabric-1 untagged	172.24.250.0... 172.24.250.0/24	172.24.250....	▼
eno1 d4:ae:52:7f:88:ad		1 Gbps/1 Gbps	Bridged phy... 0				

Imagen 22: Configuración de red estándar en Canonical Maas (Anexo A).

Anexo B: Implementación de Clúster de Almacenamiento con Ceph

Para la implementación de Ceph como infraestructura de almacenamiento de alta disponibilidad se han utilizado 4 servidores con 3 discos cada uno: 1 para el sistema operativo y 2 para los OSDs.

1. Configuración Inicial del Software en los Servidores

Siguiendo las directrices del manual de instalación de Ceph, se procedió a configurar cada uno de los servidores con las herramientas necesarias para el despliegue eficiente del cluster. Se instaló Docker para facilitar el manejo y la ejecución de los servicios en contenedores, lo cual simplifica la escalabilidad y la gestión de las aplicaciones. Además, se implementó Chrony para asegurar una sincronización precisa del tiempo entre todos los servidores del cluster. La correcta sincronización del tiempo es crucial en Ceph para mantener la coherencia de los datos y evitar conflictos que podrían surgir debido a discrepancias temporales entre las operaciones de los nodos. La instalación de estos componentes en Ubuntu Server se realiza mediante el siguiente comando:

```
$ sudo apt install docker chrony
```

Este paso inicial prepara el entorno de los servidores para un funcionamiento óptimo y coherente del cluster, estableciendo una base sólida para las operaciones de almacenamiento distribuido y la alta disponibilidad de los datos.

2. Configuración de Autenticación SSH vía llave pública

Para facilitar una comunicación segura y eficiente entre los nodos del cluster, es esencial configurar la autenticación sin contraseña para el usuario root. Este proceso comienza generando un par de llaves (pública y privada) en el nodo principal, lo que permite al usuario root acceder a los otros nodos sin necesidad de ingresar una contraseña cada vez. Esto es particularmente útil para la administración automatizada y las operaciones de mantenimiento entre los nodos del cluster.

El siguiente comando en el nodo principal se usa para generar el par de llaves utilizando el algoritmo Ed25519:

```
$ ssh-keygen -t ed25519 -b 2048
```

Este comando crea un par de llaves en el directorio `/root/.ssh/` del nodo principal. La llave pública (`id_ed25519.pub`) debe ser distribuida a cada uno de los otros nodos para configurar el acceso sin contraseña.

Posteriormente, se copia la llave pública al archivo `authorized_keys` de los otros servidores, lo que permite que el nodo principal se autentique transparentemente en los otros nodos.

Por lo general, el par de llaves se crea utilizando los algoritmos RSA o DSA. Sin embargo, en este caso se ha optado por Ed25519 debido a que es una firma digital basada en curvas elípticas. Este algoritmo ofrece mayor seguridad con claves de

menor tamaño, facilitando autenticaciones más rápidas y eficientes. Adicionalmente, Ed25519 es resistente a ataques de canal lateral, lo que incrementa la seguridad en entornos vulnerables.

3. Instalación del Administrador Cephadm

Para instalar y configurar el cluster de Ceph utilizando la última versión denominada "reef", primero se requiere la descarga del administrador de instalación, cephadm. Este proceso comienza con la descarga del script desde el repositorio oficial en GitHub. Se otorgan permisos de ejecución al archivo descargado y luego se utiliza para agregar el repositorio de Ceph correspondiente a la versión "reef".

Ejecute los siguientes comandos en el nodo principal:

Descarga del script cephadm:

```
$ curl --silent --remote-name -location \
https://github.com/ceph/ceph/raw/quincy/src/cephadm/cephadm
```

Otorgar permisos de ejecución al script:

```
$ chmod +x cephadm
```

Agregar el repositorio de Ceph para la versión "reef":

```
$ sudo ./cephadm add-repo --release reef
```

Estos pasos preparan el sistema para la posterior instalación de Ceph, asegurando que se utilice la versión más reciente y estable disponible.

4. Configuración de la Tabla de Hosts en Cada Servidor del Cluster

Se recomienda, aunque opcionalmente, editar el archivo `/etc/hosts` en cada uno de los servidores que conforman el cluster de Ceph. Este paso involucra agregar la dirección IP y el nombre asignado a cada nodo dentro del cluster, lo que facilita la comunicación y la resolución de nombres entre los nodos. Al configurar adecuadamente este archivo, se mejora la eficiencia de las conexiones internas y se minimizan los posibles problemas de comunicación que podrían surgir por resoluciones de nombres inadecuadas. A continuación se muestra un ejemplo de cómo configurar el archivo `/etc/hosts`:

```
172.24.250.11 cephnodo1
172.24.250.12 cephnodo2
172.24.250.13 cephnodo3
172.24.250.14 cephnodo4
```

5. Instalación del Cluster con Cephadm

Inicie la instalación de Ceph en el cluster ejecutando el siguiente comando con el script de Python cephadm:

```
$ sudo ./cephadm install
```

Este comando prepara cada nodo asegurando que todos los componentes necesarios de Ceph estén instalados correctamente.

6. Inicialización del Cluster con Bootstrap

En el nodo principal, se aplica el siguiente comando para inicializar el cluster de Ceph:

```
$ sudo ./cephadm bootstrap --mon-ip 172.24.250.11
```

Este comando configura el nodo principal como el monitor (mon) del cluster, utilizando la dirección IP 172.24.25.11. El proceso de bootstrap establece todos los componentes necesarios del cluster, incluyendo la configuración inicial, las claves de autenticación, y el despliegue de los servicios esenciales de Ceph. Adicionalmente, este proceso genera las credenciales de acceso al dashboard de Ceph, que permiten una gestión visual y facilitan el monitoreo del cluster. Estas credenciales son cruciales para la administración segura y eficiente del cluster.

7. Instalación de Componentes de Ceph en los Demás Nodos

En cada uno de los nodos secundarios del cluster, se debe instalar el paquete ceph-common. Este paquete contiene las utilidades comunes necesarias para la operación y el mantenimiento del cluster de Ceph. Para realizar esta instalación, se ejecuta el siguiente comando:

```
$ sudo apt install ceph-common
```

Este paso asegura que todos los nodos tengan las herramientas necesarias para integrarse y funcionar adecuadamente dentro del cluster.

8. Distribución de Llaves Públicas de Ceph a los Nodos Secundarios

Una vez configurado el nodo principal, es necesario distribuir las llaves públicas de Ceph a cada uno de los nodos secundarios del cluster. Esto permite una autenticación segura y sin contraseña entre el nodo principal y los demás nodos, facilitando la administración y la comunicación segura dentro del cluster. Para copiar las llaves públicas, se utilizan los siguientes comandos, ejecutados desde el nodo principal:

```
$ ssh-copy-id -f -i /etc/ceph/ceph.pub root@cephnodo1  
$ ssh-copy-id -f -i /etc/ceph/ceph.pub root@cephnodo2  
$ ssh-copy-id -f -i /etc/ceph/ceph.pub root@cephnodo3
```

Estos comandos utilizan la herramienta ssh-copy-id para facilitar la copia de la llave pública almacenada en /etc/ceph/ceph.pub a los archivos authorized_keys de los usuarios root en cada nodo, asegurando que las comunicaciones entre el nodo principal y los nodos secundarios sean fluidas y seguras.

9. Creación de Object Storage Daemons (OSDs)

Para la configuración de almacenamiento en el cluster de Ceph, es esencial crear y configurar los Object Storage Daemons (OSDs) en cada nodo. Los OSDs son responsables de almacenar datos de manera distribuida y proporcionan redundancia y recuperación de datos dentro del cluster. Se utilizan los siguientes comandos para agregar los OSDs en los discos designados de cada nodo:

```
$ ceph orch daemon add osd cephnodo1:/dev/vdb
$ ceph orch daemon add osd cephnodo1:/dev/vdc
$ ceph orch daemon add osd cephnodo2:/dev/vdb
$ ceph orch daemon add osd cephnodo2:/dev/vdc
$ ceph orch daemon add osd cephnodo3:/dev/vdb
$ ceph orch daemon add osd cephnodo3:/dev/vdc
$ ceph orch daemon add osd cephnodo4:/dev/vdb
$ ceph orch daemon add osd cephnodo4:/dev/vdc
```

Cada comando especifica la creación de un OSD en un disco particular de un nodo específico. Este paso es crucial para asegurar que el cluster tenga suficiente capacidad y redundancia de almacenamiento para manejar las operaciones de datos eficientemente.

10. Acceso al Dashboard de Ceph

Una vez finalizada la instalación y configuración del cluster, se puede acceder al dashboard de Ceph para monitorear y administrar el cluster. El dashboard proporciona una interfaz gráfica de usuario accesible a través del navegador. Para acceder al dashboard, se debe utilizar la siguiente URL:

<https://<IP>:8433>

Como el nodo principal tiene la dirección IP '172.24.250.11', el dashboard se puede visualizar en:

<https://172.24.250.11:8433>

Este acceso permite a los administradores supervisar el estado y el rendimiento del cluster de manera conveniente y eficiente.

Anexo C: Implementación de Galera MariaDB para Alta Disponibilidad

El paquete de instalación mariadb-server en Ubuntu 22.04 contiene las herramientas necesarias para configurar Galera MariaDB, lo que facilita su despliegue en cada uno de los nodos del clúster. La instalación se realiza mediante el siguiente comando:

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

Configuración de Seguridad para MariaDB:

Posteriormente, se inicia la configuración de seguridad del servidor de base de datos con el siguiente comando:

```
$ sudo mysql_secure_installation
```

Configuración del Clúster de Galera en los Nodos:

Para configurar el clúster de Galera, se debe ajustar el archivo `/etc/mysql/mariadb.conf.d/60-galera.cnf` en cada nodo participante. Un ejemplo de configuración para el primer nodo podría ser:

```
[mysqld]
bind-address=<0.0.0.0 | IP nododb1> # Ejemplo para el nododb 1
default_storage_engine=InnoDB
binlog_format=row
innodb_autoinc_lock_mode=2

# Configuración del cluster Galera
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so
wsrep_cluster_address="gcomm://<IP nododb1>,<IP nododb2>,<IP nododb3>"
wsrep_cluster_name="mariadb-galera-cluster"
wsrep_sst_method=rsync

# Configuración del nodo del Cluster
wsrep_node_address="<IP nododb1>" # Ejemplo para el nododb1
wsrep_node_name="nododb1"
```

Sincronización y Verificación del Clúster:

Para sincronizar el clúster, es necesario detener el servicio de MariaDB en todos los nodos, iniciar Galera en el primer nodo, y luego reiniciar MariaDB en los demás nodos:

```
$ sudo systemctl stop mariadb
$ sudo galera_new_cluster # Solo en el primer nodo
$ sudo systemctl start mariadb # Se omite en el primer nodo
```

Para verificar que el clúster de MariaDB está funcionando correctamente, se puede ejecutar:

```
$ mysql -u root -p -e "SHOW STATUS LIKE 'wsrep_cluster_size'"
```

Salida esperada:

```
Enter password:
+-----+-----+
| Variable_name | Value |
+-----+-----+
| wsrep_cluster_size | 4 |
+-----+-----+
```

Anexo D: Vista de Sunstone en un navegador

Captura de pantalla que muestra Sunstone en un navegador Web.

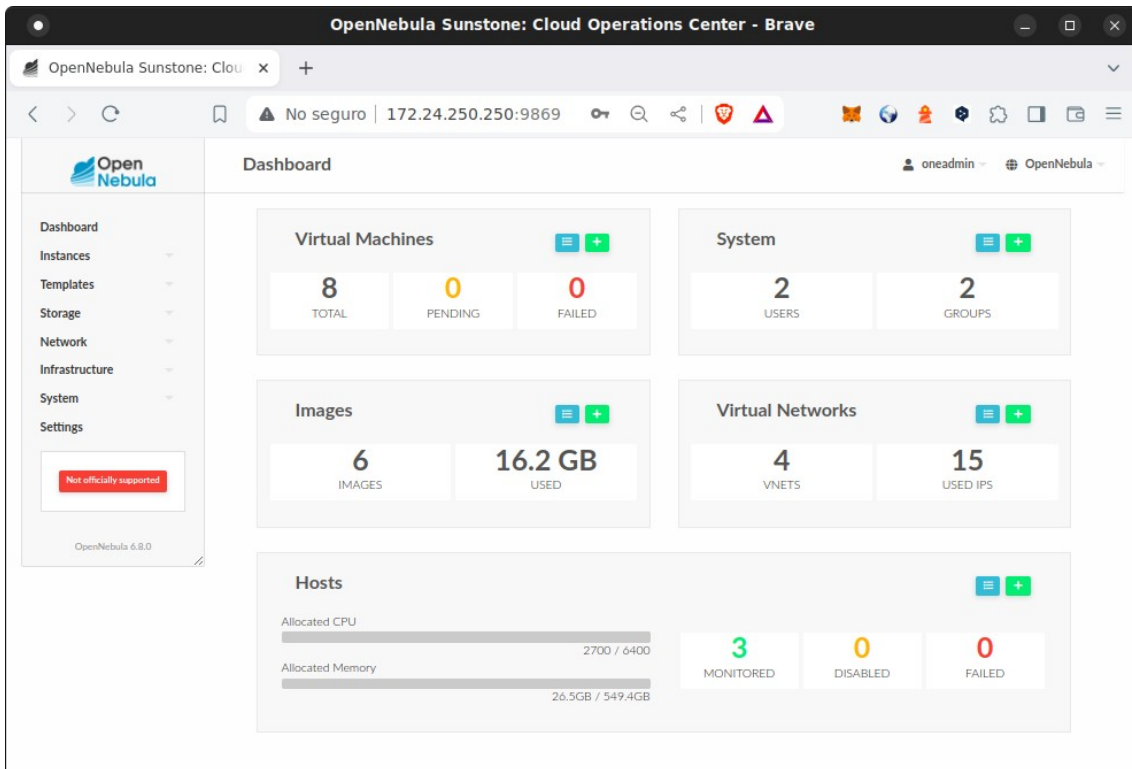


Imagen 23: Vista de Sunstone en una navegador Web (Anexo D).

Anexo E: Configuración VLS-NAT con Keepalived

Configuración VLS-NAT con Keepalived, archivo /etc/keepalived/keepalived.conf:

MASTER	BACKUP
<pre>global_defs { router_id LVS_MAIN } vrrp_instance VI_1 { state MASTER interface eth0 virtual_router_id 51 priority 100 advert_int 1 authentication { auth_type PASS auth_pass 42 } virtual_ipaddress { 190.102.240.107/27 } track_interface { eth0 } virtual_routes { 0.0.0.0/0 via 190.102.240.97 dev eth0 } } vrrp_instance VI_GATEWAY { state MASTER interface eth1 virtual_router_id 52 priority 200 advert_int 1 authentication { auth_type PASS auth_pass 62 } virtual_ipaddress { 172.24.252.253/24 } } virtual_server 190.102.240.107 80 { delay_loop 5 lb_algo rr lb_kind NAT persistence_timeout 0 protocol TCP real_server 172.24.252.207 80 { weight 1 TCP_CHECK { connect_timeout 3 nb_get_retry 1 delay_before_retry 3 connect_port 80 } } real_server 172.24.252.208 80 { weight 1 } }</pre>	<pre>global_defs { router_id LVS_MAIN } vrrp_instance VI_1 { state MASTER interface eth0 virtual_router_id 51 priority 100 advert_int 1 authentication { auth_type PASS auth_pass 42 } virtual_ipaddress { 190.102.240.107/27 } track_interface { eth0 } virtual_routes { 0.0.0.0/0 via 190.102.240.97 dev eth0 } } vrrp_instance VI_GATEWAY { state MASTER interface eth1 virtual_router_id 52 priority 200 advert_int 1 authentication { auth_type PASS auth_pass 62 } virtual_ipaddress { 172.24.252.253/24 } } virtual_server 190.102.240.107 80 { delay_loop 5 lb_algo rr lb_kind NAT persistence_timeout 0 protocol TCP real_server 172.24.252.207 80 { weight 1 TCP_CHECK { connect_timeout 3 nb_get_retry 1 delay_before_retry 3 connect_port 80 } } real_server 172.24.252.208 80 { weight 1 } }</pre>

<pre> TCP_CHECK { connect_timeout 3 nb_get_retry 1 delay_before_retry 3 connect_port 80 } } real_server 172.24.252.209 80 { weight 1 TCP_CHECK { connect_timeout 3 nb_get_retry 1 delay_before_retry 3 connect_port 80 } } virtual_server 190.102.240.107 443 { delay_loop 5 lb_algo rr lb_kind NAT persistence_timeout 0 protocol TCP real_server 172.24.252.207 443 { weight 1 TCP_CHECK { connect_timeout 3 nb_get_retry 1 delay_before_retry 3 connect_port 443 } } real_server 172.24.252.208 443 { weight 1 TCP_CHECK { connect_timeout 3 nb_get_retry 1 delay_before_retry 3 connect_port 443 } } real_server 172.24.252.209 443 { weight 1 TCP_CHECK { connect_timeout 3 nb_get_retry 1 delay_before_retry 3 connect_port 443 } } } </pre>	<pre> TCP_CHECK { connect_timeout 3 nb_get_retry 1 delay_before_retry 3 connect_port 80 } } real_server 172.24.252.209 80 { weight 1 TCP_CHECK { connect_timeout 3 nb_get_retry 1 delay_before_retry 3 connect_port 80 } } virtual_server 190.102.240.107 443 { delay_loop 5 lb_algo rr lb_kind NAT persistence_timeout 0 protocol TCP real_server 172.24.252.207 443 { weight 1 TCP_CHECK { connect_timeout 3 nb_get_retry 1 delay_before_retry 3 connect_port 443 } } real_server 172.24.252.208 443 { weight 1 TCP_CHECK { connect_timeout 3 nb_get_retry 1 delay_before_retry 3 connect_port 443 } } real_server 172.24.252.209 443 { weight 1 TCP_CHECK { connect_timeout 3 nb_get_retry 1 delay_before_retry 3 connect_port 443 } } } </pre>
---	---

Anexo F: Configuración de HAProxy para acceso a Moodle

Sección de configuración de HAProxy para acceso a los servidores Moodle en los contenedores haproxy1, haproxy2 y haproxy3. Archivo /etc/haproxy/haproxy.cfg:

```
peers moodle
  peer haproxy1 10.0.1.30:10000
  peer haproxy2 10.0.1.31:10000
  peer haproxy3 10.0.1.32:10000

frontend http_front
  bind :80
  mode http
  redirect scheme https code 301 if !{ ssl_fc }
  default_backend https_back

frontend https_front
  bind :443 ssl crt /etc/ssl/inf.uct.cl.pem
  mode http

  stick-table type ip size 100k expire 30s store conn_cur,http_req_rate(10s)

  acl too_many_connections sc1_conn_cur gt 50
  acl too_many_requests sc1_http_req_rate gt 100

  tcp-request connection reject if too_many_connections

  http-request set-header X-Forwarded-Proto https if { ssl_fc }
  http-request set-header X-Forwarded-For %[src]
  http-request deny if too_many_requests

  default_backend https_back

backend https_back
  mode http
  balance roundrobin
  stick-table type ip size 200k expire 30m peers moodle
  stick on src
  option httpchk GET /healthcheck
  http-check expect status 200
  timeout check 500ms
  server server1 10.0.1.20:80 check inter 2s fall 1 rise 2 weight 1 backup
  server server2 10.0.1.21:80 check inter 2s fall 1 rise 2 weight 3
  server server3 10.0.1.22:80 check inter 2s fall 1 rise 2 weight 3
```

Esta configuración se realiza en el proceso de despliegue de contenedores y corresponde al archivo haproxy_template.cfg referenciado en el Anexo H.

Anexo G: Configuración de HAProxy para gestionar Sunstone y Fireedge

Sección de configuración de HAProxy para acceso a las Interfaces de gestión basada en Web de OpenNebula, Sunstone y Fireedge. Archivo /etc/haproxy/haproxy.cfg:

```
frontend http_frontend
    bind *:80
    mode http
    redirect scheme https if ![ ssl_fc ]

frontend https_frontend
    bind *:443 ssl crt /etc/ssl/inf.uct.cl.pem
    mode http

    acl host_sunstone hdr(host) -i one.inf.uct.cl
    acl is_csrf_token path_reg ^/\?csrf_token=.*$
    acl is_fireedge path_beg /fireedge

    http-request add-header Access-Control-Allow-Origin * if is_csrf_token OR
is_fireedge
    http-request set-header Access-Control-Allow-Origin https://one.inf.uct.cl
if is_csrf_token OR is_fireedge
    http-request add-header Access-Control-Allow-Methods "GET, POST, OPTIONS"
if is_csrf_token OR is_fireedge
    http-request add-header Access-Control-Allow-Headers "Origin, X-Requested-
With, Content-Type, Accept, Authorization" if is_csrf_token OR is_fireedge
    use_backend fireedge_backend if is_csrf_token
    use_backend fireedge_backend if is_fireedge
    default_backend sunstone_backend

backend sunstone_backend
    mode http
    option forwardfor
    http-request set-header X-Real-IP %[src]
    http-request set-header Host %[req.hdr(Host)]
    http-request set-header X-Forwarded-For %[src]
    http-request set-header X-Forwarded-Proto https
    server sunstone_server1 172.24.250.250:9869 check

backend fireedge_backend
    mode http
    option forwardfor
    option http-server-close
    http-request set-header X-Real-IP %[src]
    http-request set-header Host %[req.hdr(Host)]
    http-request set-header X-Forwarded-For %[src]
    http-request set-header X-Forwarded-Proto https
    http-request set-header Upgrade %[req.hdr(Upgrade)]
    http-request set-header Connection "upgrade"
    http-response add-header Access-Control-Allow-Origin *
    http-response add-header Access-Control-Allow-Methods "GET, POST, OPTIONS"
    http-response add-header Access-Control-Allow-Headers "Origin, X-
Requested-With, Content-Type, Accept, Authorization"
    server fireedge_server1 172.24.250.250:2616 check
```

Anexo H: Scripts de despliegue

Para desplegar le proyecto se utilizaron secuencialmente los siguientes scripts:

```
1 ./haproxy_deploy.sh
2 ./nodos_deploy.sh nodo1 10.0.1.20
3 ./nodos_deploy.sh nodo2 10.0.1.21
4 ./nodos_deploy.sh nodo3 10.0.1.22
5 ./galera_deploy.sh
6 ./galera_check.sh nodo1
7 ./setramcache.sh nodo1
8 ./setramcache.sh nodo2
9 ./setramcache.sh nodo3
10 ./database.sh
11 ./webserver_deploy.sh nodo1
12 ./replica-webserver_deploy.sh nodo2
13 ./replica-webserver_deploy.sh nodo3
```

1. Despliegue de los proxies:

```
$ cat haproxy_deploy.sh
1 #!/bin/bash
2
3 CONTAINER1="haproxy1"
4 CONTAINER2="haproxy2"
5 CONTAINER3="haproxy3"
6 IP_MV1="172.24.252.207"
7 IP_MV2="172.24.252.208"
8 IP_MV3="172.24.252.209"
9 IP_GW="172.24.252.1"
10 IP_C01="10.0.1.30"
11 IP_C02="10.0.1.31"
12 IP_C03="10.0.1.33"
13
14 ./debian_haproxy.sh $CONTAINER1 $IP_MV1 $IP_C01 $IP_GW
15 ./debian_haproxy.sh $CONTAINER2 $IP_MV2 $IP_C02 $IP_GW
16 ./debian_haproxy.sh $CONTAINER3 $IP_MV3 $IP_C03 $IP_GW
```

```
$ cat debian_haproxy.sh
1 #!/bin/bash
2
3 CONTAINER=$1
4 IMAGE="debian/12"
5 PROFILE="proxies"
6 IP_ETH0=$2
7 IP_ETH1=$3
8 IP_GW=$4
9
10 incus launch images:$IMAGE $CONTAINER --profile $PROFILE
11
12 incus exec $CONTAINER -- bash -c "cat > /etc/systemd/network/eth0.network <<EOF
13 [Match]
14 Name=eth0
15
16 [Network]
17 Address=$IP_ETH0/24
18 Gateway=$IP_GW
19 DNS=192.168.4.6
20 EOF"
21
22 incus exec $CONTAINER -- bash -c "cat > /etc/systemd/network/eth1.network <<EOF
23 [Match]
24 Name=eth1
25
26 [Network]
```

```

27 Address=$IP_ETH1/24
28 EOF"
29
30 incus exec $CONTAINER -- systemctl restart systemd-networkd
31 incus exec $CONTAINER -- apt update
32 incus exec $CONTAINER -- apt install -y haproxy
33 # Copiar certificado
34 incus exec $CONTAINER -- mkdir /etc/ssl
35 incus file push inf.uct.cl.pem $CONTAINER/etc/ssl/
36
37 # Configuración de proxies
38 incus file push haproxy_template.cfg $CONTAINER/tmp/
39 incus exec $CONTAINER -- bash -c "cat /tmp/haproxy_template.cfg >>
/etc/haproxy/haproxy.cfg"
40 incus exec $CONTAINER -- rm /tmp/haproxy_template.cfg
41 incus exec $CONTAINER -- systemctl restart haproxy

```

2. Despliegue de los nodos Web donde será implementado Moodle

```

$ cat nodos_deploy.sh
1 #!/bin/bash
2
3 # Variables
4 CONTAINER_NAME=$1
5 IMAGE="debian/12"
6 STORAGE_POOL="local"
7 PROFILE="redlocal"
8 IP_ETH0=$2
9
10 # Crear el contenedor con el perfil específico y configuración de red
11 incus launch images:$IMAGE $CONTAINER_NAME --profile $PROFILE
12
13 incus exec $CONTAINER_NAME -- bash -c "cat > /etc/systemd/network/eth0.network <<EOF
14 [Match]
15 Name=eth0
16
17 [Network]
18 Address=$IP_ETH0/24
19 Gateway=10.0.1.1
20 DNS=192.168.4.6
21 EOF"
22
23
24 incus exec $CONTAINER_NAME -- bash -c "mv /etc/hosts /etc/hosts.old"
25 incus file push $HOME/workspace/hosts $CONTAINER_NAME/etc/
26
27 sleep 2
28
29 incus exec $CONTAINER_NAME -- apt update
30 incus exec $CONTAINER_NAME -- apt install -y syncthing mariadb-server
31
32 incus exec $CONTAINER_NAME -- bash -c "adduser --ingroup www-data --disabled-
password --gecos \"\" moodle"
33
34 incus file push syncthing.service $CONTAINER_NAME/etc/systemd/system/
35 incus exec $CONTAINER_NAME -- bash -c "sed -i 's/mouse=a/mouse= a/'
/usr/share/vim/vim90/defaults.vim"
36
37 incus exec $CONTAINER_NAME -- systemctl enable syncthing
38 incus exec $CONTAINER_NAME -- systemctl start syncthing
39
40 sleep 5
41
42 incus exec $CONTAINER_NAME -- bash -c "sed -i 's/<gui enabled=\"true\" tls=\"false\"
debugging=\"false\"/><gui enabled=\"true\" tls=\"true\" debugging=\"false\">/'
/home/moodle/.config/syncthing/config.xml"
43 incus exec $CONTAINER_NAME -- bash -c "sed -i
's/<address>127.0.0.1:8384</address>/<address>0.0.0.0:8384</address>/'
/home/moodle/.config/syncthing/config.xml"
44 incus exec $CONTAINER_NAME -- bash -c "sed -i
'</address>0.0.0.0:8384</address>a\\\\\\t<user>admin</user>/'
/home/moodle/.config/syncthing/config.xml"
45 incus exec $CONTAINER_NAME -- bash -c "sed -i '/<user>admin</user>a\\\\\\t<password>
$2a\\$10\\$CI1IX44t.a0LQF208AiZkesSF4BgL7dSyI4SKASWBxw\\CUANstW0a</password>/'

```

```

/home/moodle/.config/syncthing/config.xml"
46 incus exec $CONTAINER_NAME -- systemctl restart syncthing
47
48 # Definir contraseña y detener mariadb para configurar galera
49 incus exec $CONTAINER_NAME -- mysqladmin -u root -h localhost password "secret"
50 incus exec $CONTAINER_NAME -- systemctl stop mariadb
51
52 # Directorios para montajes
53 incus exec $CONTAINER_NAME -- mkdir -p /var/local/cache
54 incus exec $CONTAINER_NAME -- mkdir -p /var/www/moodle/cache
55 incus exec $CONTAINER_NAME -- mkdir -p /var/www/moodle/temp

```

3. Despliegue de Galera MariaDB en los nodos Web de Moodle

```

$ cat galera_deploy.sh
1 #!/bin/bash
2
3 CONTAINER1="nodo1"
4 CONTAINER2="nodo2"
5 CONTAINER3="nodo3"
6
7
8 incus file push $HOME/workspace/galera/mariadb1/60-galera.cnf
$CONTAINER1/etc/mysql/mariadb.conf.d/
9 incus file push $HOME/workspace/galera/mariadb2/60-galera.cnf
$CONTAINER2/etc/mysql/mariadb.conf.d/
10 incus file push $HOME/workspace/galera/mariadb3/60-galera.cnf
$CONTAINER3/etc/mysql/mariadb.conf.d/
11
12 incus exec $CONTAINER1 -- galera_new_cluster
13 sleep 5
14 incus exec $CONTAINER2 -- systemctl start mariadb
15 incus exec $CONTAINER3 -- systemctl start mariadb

```

4. Comprobar que Galera MariaDB funciona correctamente

```

$ cat galera_check.sh
1 #!/bin/bash
2
3 incus exec $1 - mysql --execute "SHOW GLOBAL STATUS LIKE
'wsrep_cluster_size';"

```

5.1. Configurar caché en RAM

```

$ cat setramcache.sh
1 #!/bin/bash
2
3 # Variables
4 CONTAINER_NAME=$1
5
6 # Directorios para montajes
7 incus exec $CONTAINER_NAME -- mkdir -p /var/local/cache
8 incus exec $CONTAINER_NAME -- mkdir -p /var/www/moodle/cache
9 incus exec $CONTAINER_NAME -- mkdir -p /var/www/moodle/temp
10
11 # Copiar script de montaje
12 incus file push mount_moodle_ram.sh $CONTAINER_NAME/usr/local/bin/
13 incus exec $CONTAINER_NAME -- chmod +x /usr/local/bin/mount_moodle_ram.sh
14
15 incus exec $CONTAINER_NAME -- bash -c "cat >
/etc/systemd/system/moodlemount.service <<EOF

```

```

16 [Unit]
17 Description=Mount User moodle and RAM
18 After=network.target
19
20 [Service]
21 Type=oneshot
22 ExecStart=/usr/local/bin/mount_moodle_ram.sh
23 ExecStop=/usr/local/bin/umount_moodle_ram.sh
24 RemainAfterExit=yes
25
26 [Install]
27 WantedBy=multi-user.target
28 EOF"
29
30 incus exec $CONTAINER_NAME -- systemctl enable moodlemount.service
31 incus exec $CONTAINER_NAME -- systemctl start moodlemount.service

```

5.2. Montaje de los directorios de caché en RAM

```

$ cat mount_moodle_ram.sh
1 #!/bin/bash
2
3 # Montaje en RAM
4 /usr/bin/mount -t tmpfs -o size=512M tmpfs /var/local/cache
5 /usr/bin/mount -t tmpfs -o size=512M tmpfs /var/www/moodle/cache
6 /usr/bin/mount -t tmpfs -o size=512M tmpfs /var/www/moodle/temp
7
8 echo "Montajes finalizado..."

```

5.3. Desmontaje de directorios de caché en RAM

```

$ cat umount_moodle_ram.sh
1 #!/bin/bash
2
3 echo "Desmontando caches..."
4 umount /var/local/cache
5 umount /var/www/moodle/cache
6 umount /var/www/moodle/temp
7 echo "Caches desmontadas."

```

6. Crear la base de datos para Moodle

```

$ cat database.sh
1 #!/bin/bash
2
3 CONTAINER="nodo1"
4 DATABASE="moodle"
5 USERNAME="moodle"
6 PASSWORD="secretM00dle"
7
8 incus exec $CONTAINER -- bash -c "mysql --execute \"create database
$DATABASE\""
9 incus exec $CONTAINER -- bash -c "mysql --execute \"grant all privileges
on $DATABASE.* to '$USERNAME'@'127.0.0.1' identified by '$PASSWORD'\""
10 incus file push moodle_template.sql $CONTAINER/root/
11 incus exec $CONTAINER -- bash -c "mysql $DATABASE <
/root/moodle_template.sql"

```

7. Desplegar el entorno web para Moodle en el primer nodo

```
$ cat webserver_deploy.sh
```

```
1  #!/bin/bash
2
3  CONTAINER_NAME=$1
4  CONTAINER_DATABASE="nodo1"
5  MOODLE_DIR="/home/moodle"
6  MOODLE_TEMPLATE="moodle_template.tar.gz"
7  MOODLE_DATA="moodledata.tar.gz"
8  MOODLE_SYNCACHE="syncachemoodle.sh"
9
10
11 incus exec $CONTAINER_NAME -- apt update
12 incus exec $CONTAINER_NAME -- apt install nginx php php-fpm php-mysql php-mbstring
php-curl php-intl php-tokenizer php-xmlrpc php-soap php-zip php-gd php-json php-xml php-
redis php-pear php-dev cron unison -y
13
14 incus exec $CONTAINER_NAME -- apt remove apache2 -y
15
16 incus exec $CONTAINER_NAME -- bash -c "sed -i 's/upload_max_filesize =
./upload_max_filesize = 128M/' /etc/php/8.2/fpm/php.ini"
17 incus exec $CONTAINER_NAME -- bash -c "sed -i 's/post_max_size = ./post_max_size =
128M/' /etc/php/8.2/fpm/php.ini"
18 incus exec $CONTAINER_NAME -- bash -c "sed -i 's/;max_input_vars = ./max_input_vars
= 10000/' /etc/php/8.2/fpm/php.ini"
19 incus exec $CONTAINER_NAME -- bash -c "sed -i 's/max_execution_time =
./max_execution_time = 600/' /etc/php/8.2/fpm/php.ini"
20 incus exec $CONTAINER_NAME -- bash -c "sed -i '21i          client_max_body_size
128M;' /etc/nginx/nginx.conf"
21
22 incus file push site-moodle.conf $CONTAINER_NAME/etc/nginx/sites-available/
23 incus exec $CONTAINER_NAME -- ln -s /etc/nginx/sites-available/site-moodle.conf
/etc/nginx/sites-enabled/
24 incus exec $CONTAINER_NAME -- systemctl restart nginx
25
26 incus exec $CONTAINER_NAME -- chmod 770 $MOODLE_DIR
27 incus exec $CONTAINER_NAME -- bash -c "echo 'umask 007' >> $MOODLE_DIR/.bashrc"
28
29 incus file push $MOODLE_TEMPLATE $CONTAINER_NAME/root/
30 incus file push $MOODLE_DATA $CONTAINER_NAME/root/
31 incus exec $CONTAINER_NAME -- /usr/bin/tar xzfv /root/$MOODLE_TEMPLATE -C
$MOODLE_DIR
32 incus exec $CONTAINER_NAME -- /usr/bin/tar xzfv /root/$MOODLE_DATA -C $MOODLE_DIR
33
34 incus exec $CONTAINER_NAME -- chown -R moodle:www-data $MOODLE_DIR
35 incus exec $CONTAINER_NAME -- chmod -R ug+rw $MOODLE_DIR/www
36 incus exec $CONTAINER_NAME -- chmod -R 777 $MOODLE_DIR/moodledata
37
38 # Configuración de caches en RAM
39 incus exec $CONTAINER_NAME -- bash -c "cp -R $MOODLE_DIR/moodledata/temp/*
/var/www/moodle/temp/"
40 incus exec $CONTAINER_NAME -- bash -c "cp -R $MOODLE_DIR/moodledata/cache/*
/var/www/moodle/temp/"
41 incus exec $CONTAINER_NAME -- bash -c "cp -R $MOODLE_DIR/moodledata/localcache/*
/var/local/cache/"
42
43 incus exec $CONTAINER_NAME -- chmod -R 777 /var/www/moodle
44 incus exec $CONTAINER_NAME -- chmod -R 777 /var/local/cache
45
46 incus file push $MOODLE_SYNCACHE $CONTAINER_NAME/usr/local/bin/
47 incus exec $CONTAINER_NAME -- chmod +x /usr/local/bin/$MOODLE_SYNCACHE
48 incus exec $CONTAINER_NAME -- bash -c "(crontab -l 2>/dev/null;
echo \"0,7,14,21,28,35,42,49,56 * * * * /usr/local/bin/$MOODLE_SYNCACHE\") | crontab
-"
```

7. Desplegar el entorno web para Moodle en los demás nodos

```
$ cat replica-webserver_deploy.sh
1  #!/bin/bash
2
3  CONTAINER_NAME=$1
4  MOODLE_DIR="/home/moodle"
5  MOODLE_SYNCACHE="syncachemoodle.sh"
6
7
8  incus exec $CONTAINER_NAME -- apt update
9  incus exec $CONTAINER_NAME -- apt install nginx php php-fpm php-mysql php-mbstring
php-curl php-intl php-tokenizer php-xmlrpc php-soap php-zip php-gd php-json php-xml
php-redis php-pear php-dev cron unison -y
10
11 incus exec $CONTAINER_NAME -- apt remove apache2 -y
12
13 incus exec $CONTAINER_NAME -- bash -c "sed -i 's/upload_max_filesize =
./upload_max_filesize = 128M/' /etc/php/8.2/fpm/php.ini"
14 incus exec $CONTAINER_NAME -- bash -c "sed -i 's/post_max_size = ./post_max_size =
128M/' /etc/php/8.2/fpm/php.ini"
15 incus exec $CONTAINER_NAME -- bash -c "sed -i 's/max_execution_time =
./max_execution_time = 600/' /etc/php/8.2/fpm/php.ini"
16 incus exec $CONTAINER_NAME -- bash -c "sed -i 's;/max_input_vars = ./max_input_vars
= 10000/' /etc/php/8.2/fpm/php.ini"
17 incus exec $CONTAINER_NAME -- bash -c "sed -i '21i client_max_body_size 128M;'
/etc/nginx/nginx.conf"
18
19 incus file push site-moodle.conf $CONTAINER_NAME/etc/nginx/sites-available/
20 incus exec $CONTAINER_NAME -- ln -s /etc/nginx/sites-available/site-moodle.conf
/etc/nginx/sites-enabled/
21 incus exec $CONTAINER_NAME -- systemctl restart nginx
22 incus exec $CONTAINER_NAME -- systemctl restart php8.2-fpm
23
24 incus file push $MOODLE_SYNCACHE $CONTAINER_NAME/usr/local/bin/
25 incus exec $CONTAINER_NAME -- chmod +x /usr/local/bin/$MOODLE_SYNCACHE
26 incus exec $CONTAINER_NAME -- bash -c "(crontab -l 2>/dev/null;
echo \"4,10,16,22,28,34,40,46,52 * * * * /usr/local/bin/$MOODLE_SYNCACHE\") | crontab
-"
27 incus exec $CONTAINER_NAME -- bash -c "(crontab -l 2>/dev/null;
echo \"5,12,19,26,33,40,47,54,59 * * * * /usr/local/bin/$MOODLE_SYNCACHE\") | crontab
-"
28
29 # Configuración de caches en RAM
30 incus exec $CONTAINER_NAME -- bash -c "cp -R $MOODLE_DIR/moodledata/temp/*
/var/www/moodle/temp/"
31 incus exec $CONTAINER_NAME -- bash -c "cp -R $MOODLE_DIR/moodledata/cache/*
/var/www/moodle/temp/"
32 incus exec $CONTAINER_NAME -- bash -c "cp -R $MOODLE_DIR/moodledata/localcache/*
/var/local/cache/"
33
34 incus exec $CONTAINER_NAME -- chmod -R 777 /var/www/moodle
35 incus exec $CONTAINER_NAME -- chmod -R 777 /var/local/cache
36
37 incus exec $CONTAINER_NAME -- chown -R moodle:www-data $MOODLE_DIR
38 incus exec $CONTAINER_NAME -- chmod -R g+w $MOODLE_DIR/www
39 incus exec $CONTAINER_NAME -- chmod -R 777 $MOODLE_DIR/moodledata
```