ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA EN SISTEMAS

DESARROLLO DE UN PROTOTIPO DE CLOUD COMPUTING UTILIZANDO OPENSTACK

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

JOSÉ IGNACIO GARCÍA SILVA
jose.garcia01@epn.edu.ec

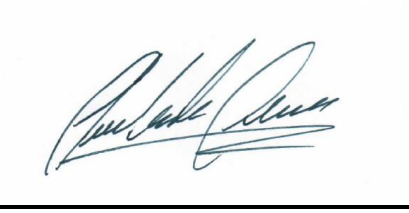
Director: Ing. Roberto Omar Andrade Paredes
roberto.andrade@epn.edu.ec

Codirector: PhD. Sang Guun Yoo
sang.yoo@epn.edu.ec

Quito, julio 2022

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por José Ignacio García Silva, bajo mi supervisión.



Ing. Roberto Omar Andrade Paredes

DIRECTOR

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por José Ignacio García Silva, bajo mi supervisión.

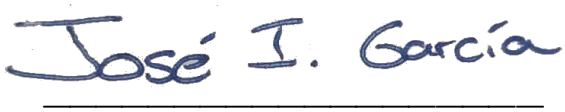
PhD. Sang Guun Yoo

CODIRECTOR DEL PROYECTO

DECLARACIÓN

Yo José Ignacio García Silva, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.



José Ignacio García Silva

AGRADECIMIENTOS

A todos los que han formado parte de esta historia que llamamos vida.
Agradecimientos especiales para:

Miembros del Smartlab: Por todos esos momentos que nos arrebató la pandemia.

Roberto Andrade: Por la confianza y paciencia que se requirió para terminar este trabajo.

Sang Yoo: Por ser fuente de inspiración para ser mejores profesionales y personas.

Carolina Narváez: Por darle un giro a mi vida de 180 grados.

Carlos Hernández: Por el apoyo al inicio de esta carrera.

Johanna Cerezo: Por tenderme la mano en situaciones críticas.

Jose Murillo: Por todas las infiltraciones al Smartlab y favores para terminar este trabajo.

Diego Portero: Por animarme a realizar este trabajo cuando ya lo daba por perdido.

Alberto García: Por ayudarme a enfrentar mis miedos.

Nancy Álvarez: Por el cariño y la preocupación que recibí.

Andrea Villafuerte: Por mostrarme que todos tenemos nuestras batallas.

Alejandra Villafuerte: Por recordarme algo muy importante.

Hanny Reinthaller: Por depositar confianza en mí desde el primer día.

Alejandra Corvalán: Por el apoyo y todos esos momentos que me hicieron más fuerte.

Juan D. Guevara: Por ser un hermano para mí.

Kevin Salazar y Martín Salazar: Por ser mis hermanos mayores.

Abel Almeida: Por siempre estar al tanto a pesar del tiempo y la distancia.

Juan M. García: Por incentivarme a ser una mejor persona cada día.

Pedro Borja: Por darme el apoyo incondicional en el momento más crítico de mi vida.

Mauricio Villafuerte: Por todos los debates constructivos que me hicieron más humilde.

Sylvia García: Por brindarme la experiencia de un hogar que jamás tuve.

Margarita García: Por todo el apoyo que recibí a lo largo de la carrera.

Fabian Ledesma: Por contribuir a mi desarrollo personal.

Marcia Ledesma: Por inspirarme a tener un corazón más grande. Y mil cosas más.

Marcia Silva: Por hacerme más empático y haberlo entregado todo desde el día 0.

Juan García: Por todo padre, jamás hubiera llegado aquí sin ti.

DEDICATORIA

Dedicado a las personas que conocí y llegué a valorar cuando ya no estaban aquí.

José García, Gonzalo García, Alicia Córdoba, Beatriz Romero y Adolfo Ledesma

ÍNDICE DE CONTENIDOS

1. CAPÍTULO PRIMERO: INTRODUCCIÓN

Los espacios especializados en infraestructura como servicio (IaaS), también llamados servicios de infraestructura en la nube son una forma de "Cloud Computing", dichos espacios constituyen un entorno en el que se almacenan datos y a su vez entrega otras herramientas de desarrollo que permiten crear y administrar redes de información, dando como resultado final un espacio con todas las herramientas, todo esto a través de las tecnologías de información (TI), que son todos los recursos tanto materiales como de red que permiten al usuario la creación y manipulación de información digital. Actualmente cada persona con acceso a internet tiene contacto con las nubes, y, por ende, hace uso de las TI a pesar de no saber su verdadero potencial, en la actualidad constituyen un recurso sumamente útil y hasta cierto grado, imprescindible, especialmente para aquellos que se dedican a profesiones relacionadas a la informática y las redes, ya que más allá del almacenamiento, son herramientas que otorgan capacidad de procesamiento y desarrollo, todo en una red virtualizada. (Kumar, Pramod, & Acken, 2020)

Como miembros de la comunidad estudiantil de la Escuela Politécnica Nacional, y a su vez parte del sistema educativo público del Ecuador, se puede palpar el déficit de herramientas con el que cuentan los estudiantes a nivel de todo el país, esto se debe a que en la mayoría de las instituciones educativas ecuatorianas, tanto profesores como estudiantes no cuentan con un espacio especializado en Infraestructura como Servicio (IaaS) que les permita implementar proyectos personales o académicos, lo que dificulta enormemente el proceso de plasmar ideas que pueden ser útiles para toda la comunidad, la carencia de herramientas como IaaS limita enormemente el potencial de la comunidad educativa, principalmente cuando se trata de los proyectos de titulación, ya que no se cuenta con la tecnología de la información adecuada. (Mendoza, 2020)

La mayoría de los computadores personales en la actualidad cuentan con tecnología para optimizar la virtualización, sin embargo, no son flexibles al momento de escalar recursos de procesamiento y memoria volátil que un proyecto puede demandar en su desarrollo; por otro lado, pese a que la EPN provee recursos computacionales a los profesores, el proceso para adquirirlos es tortuoso, conlleva trámites y papeleo, lo cual impone una barrera que puede ser abolida con el desarrollo de un prototipo de cloud computing, dicho proyecto puede ayudar no solo a los profesores, sino a todo aquel que requiera herramientas para la generación de proyectos, de esta forma, no solo un grupo selecto se beneficia, sino toda la comunidad educativa, cortando las limitaciones y fomentando la investigación. (EPN, 2010)

1.1. Planteamiento del Problema

Desde el inicio de la pandemia de COVID-19 causado por el SARSCoV2, el COE nacional al igual que muchas otras instituciones a nivel mundial, impuso restricciones a múltiples actividades laborales y educativas que impliquen la presencia de una cercanía interpersonal, y, por lo tanto, las instituciones educativas se vieron obligadas a impedir momentáneamente el ingreso de los estudiantes a sus instalaciones y posteriormente a reducirlo, incluso cuando existen ciertas esperanzas de normalidad por la vacunación, es evidente que la forma de trabajar, estudiar y relacionarse no será la misma nunca más; (COE, 2020) esto priva a los miembros de la comunidad educativa de la Escuela Politécnica Nacional a acceder a los recursos computacionales y de red que se encuentran de forma física en la institución, por lo que muchos proyectos que estaban realizándose fueron forzosamente pausados a pesar de la insistencia de los colaboradores, de estos problemas nace la idea de desarrollar un prototipo de Cloud computing que brinde recursos computacionales a los profesores y estudiantes de la Facultad de Ingeniería en Sistemas, incluso fuera del campus universitario, con una capacidad suficiente para que quienes conforman la comunidad puedan tener acceso equitativo a los recursos necesarios sin necesidad de exponerse. (COE, 2020)

Este prototipo brindará el espacio y las herramientas de red necesarias para poder desarrollar proyectos académicos, personales y universitarios, garantizando un aprendizaje significativo, con una retribución al apoyo brindado por la universidad, que a su vez asegura una IaaS a los miembros de la comunidad politécnica que sea gratuito, sin trabas legales, y de calidad.

1.2. Justificación Teórica:

"La computación en nube es un modelo que permite el acceso ubicuo, cómodo y bajo demanda a un conjunto compartido de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente aprovisionados y liberados con un mínimo esfuerzo de gestión o interacción por parte del proveedor de servicios." (Mell & Grance, 2011)

Existen 3 tipos de proveedores de computación en la nube:

- Nube Pública: Ofrece sus servicios a todos los clientes que deseen el servicio sin distinción, los usuarios pueden acceder al servicio desde Internet, sin

embargo, tienen la peculiaridad de un espacio de almacenamiento limitado. (Mell & Grance, 2011)

- Nube Privada: Los servicios que ofrece están limitados a un grupo que, en la mayoría de los casos pertenece a la misma organización, utiliza su propia red y no necesariamente debe ser accesible desde Internet, sino desde el hardware de dicha organización, aunque se pueden crear desde centros de datos alquilados. (Mell & Grance, 2011)
- Nube Híbrida: Utiliza una nube privada, con la ventaja de poder expandirse con infraestructura de la Nube Pública, para ofrecer mayor servicio. (Mell & Grance, 2011)

Las 3 nubes públicas más grandes del mundo que ofrecen sus servicios bajo demanda son Amazon Web Services, Microsoft Azure y Google Cloud (Statista, 2022), ofreciendo servicios de: computo, almacenamiento, bases de datos, analíticas, red, herramientas de gestión, herramientas de desarrollo, IoT, entre otros. Además, estos proveedores permiten formar una nube híbrida con la implementación de una nube privada de cualquier organización. (Srinivasan & Quadir, 2015)

El nivel de servicios que se puede obtener de los modelos de cloud computing está dividido principalmente en: Software como Servicio (SaaS), Plataforma como Servicio (PaaS) e Infraestructura como Servicio (IaaS). (Mell & Grance, 2011) La infraestructura como Servicio (IaaS), es la entrega de recursos computacionales (procesamiento, memoria, almacenamiento y red), y el software asociado (sistemas operativos, tecnología de virtualización, sistemas de archivos) como servicio. (Sunilkumar Manvi, 2021)

Las plataformas de IaaS de código abierto más populares son Openstack, Eucalyptus, Apache CloudStack, Apache Mesos y OpenNebula (Jigsaw, 2021); siendo OpenStack la plataforma open source más implementada del mundo, esta plataforma está conformado por distintas herramientas denominadas proyectos, los cuales permiten al usuario encargarse de los servicios del cloud computing (Openinfra, The Most Widely Deployed Open Source Cloud Software in the World, 2021). Siendo estas plataformas las más populares en lo que respecta a open source, nos permiten implementar su modelo de forma gratuita y modificarlo según las necesidades del usuario, adaptándose perfectamente a los objetivos de este proyecto.

1.3. Justificación Metodológica:

OpenStack es un proyecto muy extenso que está dividido en componentes. El uso de esta metodología facilita su desarrollo, configuración, manejo y mantenimiento; y, debido a su amplitud, se hace necesario descomponer a estos componentes en pequeñas unidades de trabajo para poder utilizarlos de manera ordenada y eficiente. (Openstack, OpenStack Services, 2021)

Ya que algunos componentes dependen de otros para operar normalmente, se debe establecer orden y prioridad para la instalación de estos, y a su vez priorizar a las unidades de trabajo de cada uno de ellos para lograr un funcionamiento y ensamblaje óptimo de todas las partes. (Openstack, OpenStack Services, 2021)

Para un proyecto se necesita una metodología que permita planificar y visualizar el cumplimiento de objetivos y el surgimiento de nuevas tareas, la metodología escogida es Kanban; Kanban es una metodología ágil y flexible que ofrece un marco de trabajo, en donde se puede clasificar y dar prioridad a las unidades de trabajo que se vayan agregando y utiliza una pizarra como herramienta, que consta de varias columnas en donde las unidades de trabajo van fluyendo de izquierda a derecha, tomando en cuenta la prioridad de arriba hacia abajo. (Benson, 2011) Además, permite que se puedan agregar más columnas dependiendo de las necesidades se vayan presentando, de ellas, las más importantes son:

- Backlog: Utilizada para establecer cualquier idea relacionada con el proyecto que puede o no ser implementada. (Benson, 2011)
- Listo: En esta columna se agregan las unidades de trabajo realizables y plausibles que agregarán valor al proyecto. (Benson, 2011)
- Doing: Aquí se establece el trabajo que se está realizando tomando en cuenta de no superar el límite de tareas en proceso (WIP) descrito más adelante. (Benson, 2011)
- Hecho: Se colocan todas las unidades de trabajo finalizadas. (Benson, 2011)

La metodología ágil Kanban se rige por tres reglas:

- Visualizar el flujo de trabajo: El objetivo de esta regla es definir las unidades de trabajo de forma clara; estas unidades de trabajo son puestas en un tablero en donde se puede visualizar fácilmente todo el trabajo del proyecto, tomando en cuenta las prioridades de cada unidad, las ideas que se pueden implementar y

los objetivos de desarrollo que ya fueron cumplidos, reflejando el desarrollo del proyecto a tiempo real. (Benson, 2011)

- Determinar el límite del WIP (Work In Progress): Permite limitar el número de unidades de trabajo facilitando el desarrollo del proyecto de manera eficiente, ya que se adapta a la capacidad de trabajo de cada persona; También es posible detectar cuellos de botella o unidades de trabajo que toman más tiempo del estimado, permitiendo llegar a soluciones rápidamente. (Benson, 2011)
- Controlar el tiempo que se va a tomar el completar una actividad (Lead time): Se lo obtiene desde la creación de la unidad de trabajo hasta su entrega final. Con esto se pueden obtener métricas del proyecto, que ayudan a aplicar la mejora continua. (Benson, 2011)

En cuanto al Technology Acceptance Model, está basado en la premisa de que la percepción de la utilizad y la percepción de la facilidad de uso de un sistema son determinantes directas para un usuario del sistema, debido a esto, no se puede dejar de lado las características de diseño, ya que a pesar de que son consideradas una variable externa que no define en sí la utilidad de un sistema, el hecho de que se perciba más sencillo y más útil permite al usuario aumentar su productividad, y esto es, en si, la prioridad de aquellos que se dedican a crear programas de primera interacción con el usuario, representa una relación directa de causalidad para que el usuario elija o no un sistema. (Davis, 1985)

1.4. Justificación Práctica:

El prototipo de cloud computing permitirá a los estudiantes y profesores de la FIS a acceder a los recursos computacionales sin trámites y papeleos excesivos para poder aprovisionar máquinas virtuales con varios sistemas operativos. Algunas máquinas virtuales podrán ser pre-configuradas antes de su lanzamiento, con parámetros que el usuario necesite para agilizar el proceso de aprovisionamiento, por otro lado, tendrán la opción de utilizar tecnologías de contenerización como Docker. (Docker, 2022)

Los profesores en las instituciones educativas tendrán la posibilidad de virtualizar ambientes aislados que realizarán para desarrollar actividades y facilitar el entendimiento de la materia impartida en clase. Además, Los proyectos que se implementan en el campus de la universidad también podrán ser alojados y contar con el beneficio de menor latencia de red por estar en la misma a pocos saltos de la red local donde se implementen.

Los estudiantes que tienen la necesidad de utilizar una máquina virtual como servidor para los proyectos, podrán también aprovisionarse de recursos computacionales temporalmente, siendo esto muy útil en caso de que el proyecto requiera utilizar infraestructura, dispositivos o tecnología adicional de los laboratorios que solamente son accesibles dentro de la red de la universidad.

1.5. Objetivos

1.5.1. Objetivo General:

Desarrollar un prototipo de cloud computing utilizando OpenStack para brindar IaaS, utilizando las guías de diseño de Arquitectura, instalación e imagen de máquina virtual que proporciona OpenStack.

1.5.2. Objetivos Específicos

1. Adaptar las guías proporcionadas por OpenStack a las necesidades y realidad del proyecto. (Openstack, Installation Guide, 2021), (Openstack, OpenStack Architecture Design Guide, 2021), (Openstack, OpenStack Virtual Machine Image Guide, 2021)

2. Seleccionar correctamente el sistema operativo y los componentes de OpenStack que conformarán el prototipo de cloud computing.

3. Instalar, configurar e integrar los componentes de OpenStack.

4. Configurar el prototipo de cloud computing para brindar IaaS.

5. Definir imágenes pre-configuradas de los sistemas operativos que se utilizarán

6. Aplicar el Modelo de Aceptación de Tecnología (TAM), en el prototipo de computación en la nube. (Openinfra, Red Hat, Inc. and others, 2021)

2. CAPÍTULO SEGUNDO: MARCO TEÓRICO

2.1 Cloud Computing:

Computación en la nube / Cloud Computing: Es un término que se utiliza para representar la entrega de servicios de IT, recursos computacionales, aplicaciones y datos, en donde la localización y la forma de implementación del proveedor es transparente para los usuarios. La NIST define al Cloud Computing como "Un modelo que permite el acceso ubicuo, cómodo y bajo demanda a un conjunto compartido de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios), que pueden ser rápidamente aprovisionados y liberados con un mínimo esfuerzo de gestión o interacción del proveedor de servicios." (Mell & Grance, 2011)

2.2 Modelos de entrega de servicios

La nube ofrece servicios que pueden ser clasificados en tres modelos: SaaS (Software as a Service), PaaS (Platform as a Service) y IaaS (Software as a Service).

2.2.1. SaaS

Está enfocado en proporcionar a los usuarios acceso a una o a varias aplicaciones ejecutándose en el proveedor. Los servicios que estos brindan pueden ser accedidos por medio de un navegador web o una interfaz específica desarrollada por el proveedor. El usuario está limitado a utilizar solamente la aplicación, es decir, no tiene el control sobre el ambiente en donde está implementada. (Mell & Grance, 2011)

2.2.2. PaaS

Está dirigido para desarrolladores. El usuario tiene la capacidad de implementar sus propias aplicaciones utilizando herramientas, lenguajes de programación y librerías, soportadas por el proveedor, sin embargo, no tiene control sobre la infraestructura que ejecuta la aplicación. (Mell & Grance, 2011)

2.2.3. IaaS

El usuario tiene acceso directo a los recursos de hardware virtualizados. Puede aprovisionar recursos de procesamiento, red, servidores, sistemas operativos, almacenamiento y cualquier otro recurso que sea fundamental para implementar

cualquier software, pero no tiene el control sobre la implementación de la infraestructura. (Mell & Grance, 2011)

2.3. Modelos de implementación

2.3.1. Público

El servicio es accesible desde Internet y es de uso general. Este puede ser gratis o pagado bajo demanda. (Kumar, Pramod, & Acken, 2020)

2.3.2. Privado

El servicio es exclusivo para una sola organización. Puede ser operada y gestionada por la misma o delegada a terceros. La mayoría de las veces está implementada en sus propias instalaciones. (Kumar, Pramod, & Acken, 2020)

2.3.3. Comunitario

El servicio está enfocado a una comunidad en específico que tiene metas en común. Puede estar manejada por la misma comunidad, terceros o la combinación de ambos. (Mell & Grance, 2011)

2.3.4. Híbrido

Está compuesta por dos o más formas de implementación (pública, privada o comunitaria) en donde trabajan de forma colaborativa, pero existe una división clara entre las dos. (Kumar, Pramod, & Acken, 2020)

2.4. Plataformas open-source de cloud computing

El cloud computing nace de la necesidad de brindar servicios a través del internet, uno de los modelos que usa el cloud computing es el IaaS, que permite a los administradores del sistema dividir los recursos de almacenamiento, computación y red que el usuario necesita para ejecutar sistemas operativos y aplicaciones, el hecho de hacer cloud computing basado en open source permite una colaboración a la comunidad, las plataformas open source más populares a nivel mundial son: (Jigsaw, 2021)

- OpenStack: el proyecto de cloud computing basado en el software libre más usado a nivel mudial, financiado por grandes corporaciones, con años de trayectoria y cómodo para el usuario. (Jigsaw, 2021)
- OpenNebula: es una plataforma de cloud computing que se origina en la Universidad Complutense de Madrid, enfocado en la computación distribuida, virtualización e IaaS. (Jigsaw, 2021)
- Eucalyptus: plataforma open source IaaS, usada para crear nubes híbridas o privadas con acceso a infraestructura, soporta servicios de Amazon Web Services. (Jigsaw, 2021)
- CloudStack: plataforma IaaS de citrix, nace de la Apache Software Foundation, soporta servicios web de amazon. (Jigsaw, 2021)

2.5. Virtualización

El término de virtualización se acuñó en 1960 para referirse a las máquinas virtuales, la creación y mantenimiento de dichas máquinas virtuales se ha denominado virtualización de plataformas. La virtualización de plataformas se realiza en hardware con un programa que actúa como host, y un entorno de computación simulado para la máquina virtual huésped. Las máquinas virtuales ejecutan su propio sistema operativo, y se ejecuta como si estuviera directamente en el hardware, esto supone una abstracción de los recursos de una computadora a un plano totalmente virtual, sin que signifique sacrificar algunas de sus funciones. (Sunilkumar Manvi, 2021)

2.6. Tipos de Hipervisor

Las máquinas virtuales necesitan obtener recursos computacionales mediante un hipervisor, este comparte sus recursos de memoria y procesamiento; los hipervisores pueden clasificarse en 2 tipos:

- Tipo 1: llamado hipervisor nativo o unhosted, este hipervisor de ejecuta directamente en el hardware para lograr controlarlo y a su vez controla el sistema operativo huésped, acortando el overhead que genera que una máquina virtual deba transitar el sistema operativo para llegar al hardware, desde este hipervisor se controlan las máquinas virtuales. (Devassy, Mukhedkar, & Vettathu, 2016)

- Tipo 2: denominado como hosted, es el hipervisor que se ejecuta dentro del sistema operativo, por lo que es dicho sistema operativo del hospedador quien se encarga de dividir los recursos. (Devassy, Mukhedkar, & Vettathu, 2016)

2.7. Características de Virtualización en Cloud Computing

Para iniciar el Cloud Computing siempre debe haber una base de virtualización, ya que es precisamente una máquina virtual con su capacidad de almacenamiento y procesamiento la que brinda el beneficio de las nubes como tal, separando estas herramientas de los recursos físicos del Hardware, las características de la virtualización que permiten la implementación del Cloud Computing son las siguientes (Sunilkumar Manvi, 2021):

2.7.1. Particionamiento

Es una división de los recursos, así es como varias aplicaciones y sistemas operativos pueden funcionar en el mismo hardware, funciona de forma similar en las máquinas virtuales, permitiendo la optimización de las herramientas de la máquina. (Sunilkumar Manvi, 2021)

2.7.2. Aislamiento

El hecho de que una máquina virtual esté aislada implica que no tiene contacto con otras máquinas virtuales ni con el sistema físico que la hospeda, por lo que, si existe algún inconveniente con una de las máquinas, las otras no se ven afectadas. (Sunilkumar Manvi, 2021)

2.7.3. Encapsulación

La máquina virtual puede mostrarse y guardarse como un archivo único en el software hospedador, esto ayuda a su fácil identificación, y protege a la máquina virtual de la interferencia de otras aplicaciones del hospedador. (Sunilkumar Manvi, 2021)

2.7.4. Consolidación

La virtualización permite que coexistan varios sistemas operativos en el mismo servidor, sin necesidad de adquirir más hardware, incluso varias versiones del mismo sistema operativo pueden coexistir. (Sunilkumar Manvi, 2021)

2.7.5. Flexibilidad de implementación

En las máquinas virtuales, los desarrolladores de aplicaciones tienen la opción de probarlas en distintos sistemas operativos y adaptarlas a cada uno y personalizarlas acorde a los mismos, además, el aislamiento y las particiones les permiten optimizar los recursos de las máquinas virtuales.

2.7.6. Migración y clonación

Las máquinas virtuales pueden moverse, lo que permite disminuir la carga de trabajo, esta capacidad de migración permite la recuperación de la máquina virtual en caso de una falla del hardware, además la facilidad de clonación permite su uso remoto, independiente del hardware hospedador. (Sunilkumar Manvi, 2021)

2.7.7. Estabilidad y seguridad

Ya que cada máquina virtual está aislada de otra, los trabajos de cada una no interfieren con las demás, y ya que cada una puede tener su propio sistema operativo con seguridad propia, aumenta la estabilidad y la seguridad de este método. (Sunilkumar Manvi, 2021)

2.8. Ventajas y Desventajas de la Virtualización

La virtualización ofrece múltiples ventajas que continuamente impulsan al usuario a buscarla, empezando con el beneficio económico, ya que sin necesidad de adquirir hardware permite manejar gran cantidad de máquinas y sistemas operativos a un menor precio; otra ventaja es la capacidad de predicción de costos, ya que los proveedores de servicios de virtualización ofrecen numerosas opciones con distintos costos, lo que permite predecir los costos y mantener los presupuestos en caso de empresas y usuarios que requieren mayor cantidad de servicios. Ya que solo se requiere de una unidad de hardware y software hospedador, se puede actualizar frecuentemente, manteniendo un buen funcionamiento de las máquinas virtuales, y al ser cada máquina independiente, su tiempo de actividad es ronda el 99.99%, mejorando la percepción del usuario. (Sunilkumar Manvi, 2021)

Otra ventaja de la virtualización es que permite un despliegue rápido de los recursos del sistema, ya que no necesita establecer redes locales ni depende de máquinas físicas, agilizando los procesos; y finalmente, genera un gran ahorro de energía sea para

usuarios particulares o compañías, ya que al necesitar una fuente de poder únicamente para una máquina física que a su vez mantiene a numerosas máquinas virtuales, representa una ganancia neta. (Sunilkumar Manvi, 2021)

2.9. OpenStack

OpenStack es un servicio de cloud computing a base de opensource, ofrece una plataforma multiusuario de servicios compartidos, esta fue creada por la NASA y Rackspace, en la actualidad, existe una serie de compañías que aportan al desarrollo y mantenimiento de este proyecto; (Solberg & Silverman, 2017) es por defecto el software de cloud computing open source para nubes privadas más implementado en el mundo.

OpenStack es manejado por la fundación OpenInfra desde el año 2012, (Openinfra, About the OpenINfra Foundation, 2021) el liderazgo y la toma de decisiones es regido por un comité compuesto por trece personas, de los cuales nueve son elegidos generalmente por los otros miembros. (Solberg & Silverman, 2017)

OpenInfra es una fundación que trabaja a nivel mundial, su nombre deriva de Open Infrastructure, y como tal, esta fundación ayuda a múltiples comunidades de Software libre a construir herramientas, basados en el precepto de la colaboración, crean y prueban componentes de software, y posteriormente los comparten con la misma comunidad, y cada miembro de dicha comunidad es bienvenido a presentar sus ideas, sugerencias y contribuciones que puedan ayudar a mejorar el software. (Openinfra, Open Infrastructure, 2021)

Las decisiones estratégicas y financieras son manejadas por la junta directiva, el desarrollo de OpenStack está enfocado en un proceso de desarrollo y diseño abierto, cuyos ciclos de desarrollo son aproximadamente de 6 meses con metas establecidas; en cada ciclo se reúnen requerimientos del foro de la comunidad, los cuales son enviados por los usuarios, luego los developers se reúnen en el Project Teams Gathering (PTG), y se define el desarrollo y la colaboración entre equipos para lograr un desarrollo satisfactorio, en el que habrá participado activamente la comunidad open source, permitiendo un ciclo de mejora continua. (Openinfra, Open Infrastructure, 2021)

OpenStack es un sistema modularizado que está compuesto por varios proyectos, cada proyecto representa un módulo y está desarrollado por un equipo respectivo; es importante destacar que los proyectos están desarrollados principalmente en Python y

son implementados en la mayoría de los casos, en sistemas operativos GNU/Linux, Keystone es el componente que ofrece el servicio de identidad a la implementación de OpenStack. (Openstack, OpenStack Roadmap, 2021)

2.10. Openstack Xena

OpenStack Xena es la vigésima cuarta versión estable lanzada en octubre 6 del 2021. Tiene un soporte estimado hasta abril 6 del 2023. Precede a la versión en desarrollo Yoga. Las novedades que trae es el soporte de nuevas características de hardware, mejora en la integración de componentes OpenStack y la reducción de deuda técnica en general del proyecto. (Openstack, OpenStack's 24th Release, Xena, Wields Powerful Hardware Support, 2022)

2.11. DevStack

Se trata de una serie de comandos que permiten establecer rápidamente un entorno completo de OpenStack, basado en las últimas actualizaciones de la rama Master, se usa de forma interactiva como un entorno de desarrollo y como la base de numerosos proyectos de pruebas de funcionamiento de OpenStack. (Openstack, DevStack, 2021)

2.12. Componentes de Openstack

2.12.1. OpenstackClient

Es una línea de comandos que puede interactuar de manera estructurada y en conjunto con los componentes de: computación, identidad, imagen, almacenamiento de objetos y API de almacenamiento por bloques. (Openstack, Openstack client, 2021)

Su diseño contiene las siguientes características:

- Tener la capacidad de extender o reemplazar librerías API de OpenStack escritas en Python. (Openstack, Openstack client, 2021)
- Producir un output consistente con formatos estándares para su fácil procedimiento de procesamiento. (Openstack, Openstack client, 2021)
- Contener una Shell embebida para poder ejecutar múltiples comandos con una sola autenticación. (Openstack, Openstack client, 2021)

2.12.2. Keystone

El servicio de identidad es un punto único integrado que se encarga de gestionar la autenticación, autorización y el catálogo de servicios; el catálogo de servicios es una colección de los servicios disponibles en la implementación de OpenStack utilizado por los usuarios y otros servicios. Por razones de seguridad, cada servicio registrado en el catálogo de servicios puede tener uno o más tipos de endpoints: administración, pública o interna. Además, OpenStack, soporta múltiples regiones para escalabilidad, siendo "RegionOne" la región predeterminada. (Openstack, Identity service overview, 2021)

En conjunto, las regiones, los servicios y los endpoints creados en el servicio de identidad engloba el catálogo de servicios. (Openstack, Identity service overview, 2021)

Keystone está formado por los siguientes 3 componentes:

- Servidor: Un servidor centralizado que proporciona servicios de autenticación y autorización mediante una API RESTful. (Openstack, Identity service overview, 2021)
- Controladores: Permiten integrar otros servicios de backend al servidor centralizado. Los servicios de backend son utilizados para acceder a la información de identidad externa a OpenStack como bases de datos SQL o servidores LDAP. (Openstack, Identity service overview, 2021)
- Módulos: Los módulos interceptan las peticiones de servicio, extraen las credenciales del usuario y las envían al servidor centralizado para su respectiva autorización. Se ejecutan dentro del espacio de un componente de OpenStack que está utilizando el servicio de identidad, Keystone tiene sus propios conceptos que definen como se relaciona con OpenStack como un todo. (Openstack, Identity service overview, 2021)

El servicio de Identidad tiene abstracciones que permiten implementar la autorización y gestión de acceso sobre los recursos computacionales. (Martinelli, Nash, & Topol, 2016) Estos se dividen en:

- Proyecto: Es una abstracción utilizada por otros servicios de OpenStack para agrupar y aislar recursos. El propósito fundamental de Keystone es ser el registro de proyectos y tener la capacidad de permitir o denegar el acceso a los recursos. (Martinelli, Nash, & Topol, 2016)

- Dominio: Es una agrupación de usuarios y proyectos que sirve como división lógica y usualmente representa una organización. Es posible dividir los recursos de la nube en silos para uso específico de cada organización. Así, se evita las colisiones de proyectos entre una o más organizaciones. (Martinelli, Nash, & Topol, 2016)
- Usuarios y grupos de usuario (Actores): Son las entidades que reciben el acceso a los recursos que son aislados por Dominios y Proyectos. Los grupos son colecciones de Usuarios. (Martinelli, Nash, & Topol, 2016)
- Funciones: Son utilizados para impartir un sentido de Autorización. Un actor puede tener uno o más roles. (Martinelli, Nash, & Topol, 2016)
- Target: Utilizado para referirse a un Dominio o Proyecto indistintamente, ya que, en ambiente es posible asignar un rol. (Martinelli, Nash, & Topol, 2016)
- Asignación: Es una combinación ternaria entre un actor, un target y un rol. Las asignaciones de roles pueden ser: otorgadas y revocadas y también puede ser heredadas entre grupos y usuarios y / o dominios y proyectos. (Martinelli, Nash, & Topol, 2016)

Keystone soporta principalmente 2 proveedores de Tokens, utilizados en la autenticación de usuarios y procesos. (Martinelli, Nash, & Topol, 2016)

- UUID: Una cadena de 32 caracteres generados aleatoriamente. Son emitidos y validados por el servicio de Identidad. Son convenientes para utilizarlos en URL's y deben estar almacenados en un backend persistente como una base de datos. (Martinelli, Nash, & Topol, 2016)

  La desventaja al utilizar Tokens UUID radica en la necesidad de validarlos cada vez que se realiza una petición al servidor, Keystone se convierte en un cuello de botella y en consecuencia el servicio de OpenStack en general se torna lento. (Martinelli, Nash, & Topol, 2016)
- Tokens Fernet: Contienen una cantidad limitada de información, como identidad del usuario, proyecto, información de expiración del token y otros datos utilizados para auditoría. Los Tokens Fernet son firmados utilizando una llave simétrica, con el fin de evitar la manipulación. La desventaja al utilizar Tokens Fernet, es la distribución de las llaves simétricas a través de varias regiones de OpenStack. (Martinelli, Nash, & Topol, 2016)

Identidad: El servicio de Identidad provee los grupos y usuarios (actores), los cuales pueden proceder de varias localizaciones como SQL, LDAP y proveedores de identidad federados. (Martinelli, Nash, & Topol, 2016)

- SQL: Keystone permite almacenar actores en SQL, las bases de datos soportadas son MySQL, PostgreSQL y DB2. La desventaja es que Keystone actúa como el proveedor de Identidad.
- LDAP: Permite leer y escribir actores en el Protocolo ligero de acceso a directorios (LDAP). En el caso de utilizar LDAP con la opción de solo lectura, la cantidad de privilegios que necesita Keystone para funcionar es mínima.
- Múltiples backends: A partir de la versión Juno, Keystone soporta múltiples backends de identidad. Cada dominio solo puede tener un proveedor de identidad (backend). La mayoría de las veces, el dominio predeterminado utiliza SQL, ya que puede ser utilizado para almacenar cuentas de servicios. La ventaja de utilizar un backend por dominio es la capacidad de integrar múltiples backends en la implementación. (Martinelli, Nash, & Topol, 2016)
- Proveedores de Identidad: Keystone es capaz de utilizar autenticación federada por medio de los módulos Apache. Los usuarios son tratados como efímeros y no son almacenados en Keystone, además, tendrán sus atributos mapeados en un rol de base basada en grupos. Un proveedor de identidad es tratado como una fuente de identidad que puede estar conformado por varios backends (SQL, MongoDB) o inicios de sesión sociales (Google, Facebook, Twitter). (Martinelli, Nash, & Topol, 2016)

2.12.3. Swift

Swift está encargado del almacenamiento basado en objetos, es utilizado para almacenar datos de forma redundante y escalable mediante clústeres. Es un sistema de almacenamiento de largo plazo para una gran cantidad de datos estáticos que pueden ser recuperados y actualizados, no existe un punto de control central, por lo que proporciona una mayor escalabilidad, redundancia y permanencia, Swift se encarga de replicar y mantener la integridad de los datos en todo el clúster. (Openstack, Object Storage service overview, 2020)

Los clústeres de almacenamiento pueden escalar de forma horizontal añadiendo nuevos nodos. En el caso de que un nodo falle, OpenStack replica el contenido a partir de otros nodos activos. También es posible utilizar dispositivos de almacenamiento de bajo costo. Los datos son accesibles mediante una API compatible con S3, que puede

integrarse en otras aplicaciones para realizar copias de seguridad, archivar y retener datos. (Openstack, Object Storage API overview, 2020)

2.12.4. Glance

Glance provee el servicio de imagen para OpenStack, en donde los usuarios pueden subir y encontrar imágenes y definiciones de metadatos. Las imágenes pueden ser almacenadas en varios lugares como File systems o sistemas de almacenamiento de objetos como OpenStack Swift. Por otro lado, las definiciones de metadatos, brinda la capacidad de determinar programáticamente, los diferentes valores de llave y valores válidos que pueden ser aplicados a los recursos de OpenStack. (Pepple, Deploying OpenStack, 2015) Está diseñado con arquitectura de cliente-servidor que proporciona una API al usuario y está conformado por los siguientes componentes:

- Glance-api: Una aplicación WSGI que acepta llamadas API para el descubrimiento de imágenes, recuperación y almacenamiento. (Pepple, Deploying OpenStack, 2015)
- Glance-registry: Un servicio privado interno que almacena, procesa y recupera metadatos de las imágenes, como tamaño y tipo. (Pepple, Deploying OpenStack, 2015)
- Base de datos: Almacena los metadatos de cada imagen.
- Repositorio de almacenamiento para los archivos de imagen: Repositorio utilizado para almacenar imágenes como Filesystems o storage de objetos. (Pepple, Deploying OpenStack, 2015)
- Servicio de definición de Metadatos: Una API usada para definir metadatos de forma personalizada. Los metadatos pueden ser utilizados para diferentes recursos como imágenes, artefactos, volúmenes, flavors, entre otros. Los metadatos pueden ser utilizados para diferentes recursos como imágenes, artefactos, volúmenes, entre otros. (Pepple, Deploying OpenStack, 2015)

2.12.5. Placement

El servicio de *API placement* fue introducido en la versión Newton en el repositorio de Nova y fue extraído a un repositorio independiente en la versión Stein. Es un API Rest y modelo de datos utilizado para realizar un tracking de inventarios y usos de los proveedores de recursos; un proveedor de recursos puede ser un nodo de computación, un pool de almacenamiento compartido, o un pool reservado de IPs; por ejemplo, una instancia creada en un nodo de computación puede consumir recursos como CPU, RAM

del nodo de computación, disco de un pool externo de almacenamiento compartido y direcciones IP de un pool externo. (Openstack, Openstack Placement, 2021)

Estos recursos son consumidos y monitoreados como clases. Las clases de recursos (DISK_GB, MEMORY_GB y VCPU) proveen la capacidad de definir clases de recursos personalizados dependiendo de la necesidad. Cada proveedor de recursos puede tener un conjunto de características que escriban aspectos cualitativos, como el espacio de disco disponible de Disco de estado sólido (SSD). (Openstack, Openstack Placement, 2021)

2.12.6. Nova

Nova, es el componente encargado de proveedores servidores virtuales, contenedores de aplicación o un sistema físico a larga escala. Similar en funcionalidad y alcance al servicio EC2 de Amazon, permite crear, gestionar y destruir servidores virtuales mediante una API programable, además, brinda el control completo de los recursos computacionales. (Openstack, OpenStack Compute (nova), 2020)

Existen 3 herramientas para poder interactuar con Nova:

- Horizon: La web UI oficial del proyecto OpenStack
- Cliente OpenStack: El CLI oficial del proyecto.
- Nova Client: Utilizado para configuraciones muy avanzadas o comandos administrativo que OpenStack Client no soporta.

Nova es típicamente implementado en conjunto con otros componentes de OpenStack como parte de una infraestructura más grande, Nova depende de la correcta implementación de Keystone, Glance, Placement y Neutron, en la tabla 1 se describen los hipervisores, además, la ilustración a continuación muestra la interacción de los hipervisores en este proyecto. (Pepple, Deploying OpenStack, 2015)

Tabla 1 Tipos de Hipervisores

Hipervisor	**Descripción**
KVM	Kernel-based Virtual Machine: Soporta formatos de disco virtual QEMU, utiliza QEMU para ejecutar la máquina virtual. Los formatos soportados incluyen

	imágenes raw, qcow2 y formatos WMware.
LXC	Linux Containers (mediante libvirt), utilizados para ejecutar máquinas virtuales basadas en Linux.
QEMU	Quick EMUlator, utilizado generalmente para propósitos de desarrollo
VMware vSphere 5.1.0 en adelante	Ejecuta imágenes VMware de Linux y Windows mediante una conexión al servidor vCenter.
Xen (utilizando libvirt)	El hipervisor del proyecto Xen, utilizando libvirt como interfaz de administración en nova-compute para ejecutar máquinas virtuales de Linux, Windows, FreeBSD y NetBSD.
XenServer	XenServer, Xen Cloud Platform (XCP) y otras variantes Xen basadas en XAPI que ejecutan máquinas virtuales deLinux o Windows. Se debe instalar el servicio de nova-compute en una máquina virtual paravirtualizada
Hyper-V	Virtualización de servidores con Microsoft Hyper-V, utilizado para ejecutar máquinas virtuales de Windows, Linux y FreeBSD. Ejecuta nova-compute de forma nativa en la plataforma de virtualización de Windows.

Virtuoxxo 7.0.0 en adelante	Contenedores de sistema operativo y máquinas virtuales KVM soportadas mediante libvirt. Los formatos de imagen soportados incluyen ploop y qcow2.
Power VM	Servidor de virtualización con workloads de IBM Power VM para AIX, IBM i y Linux en la plataforma de PowerSystems.

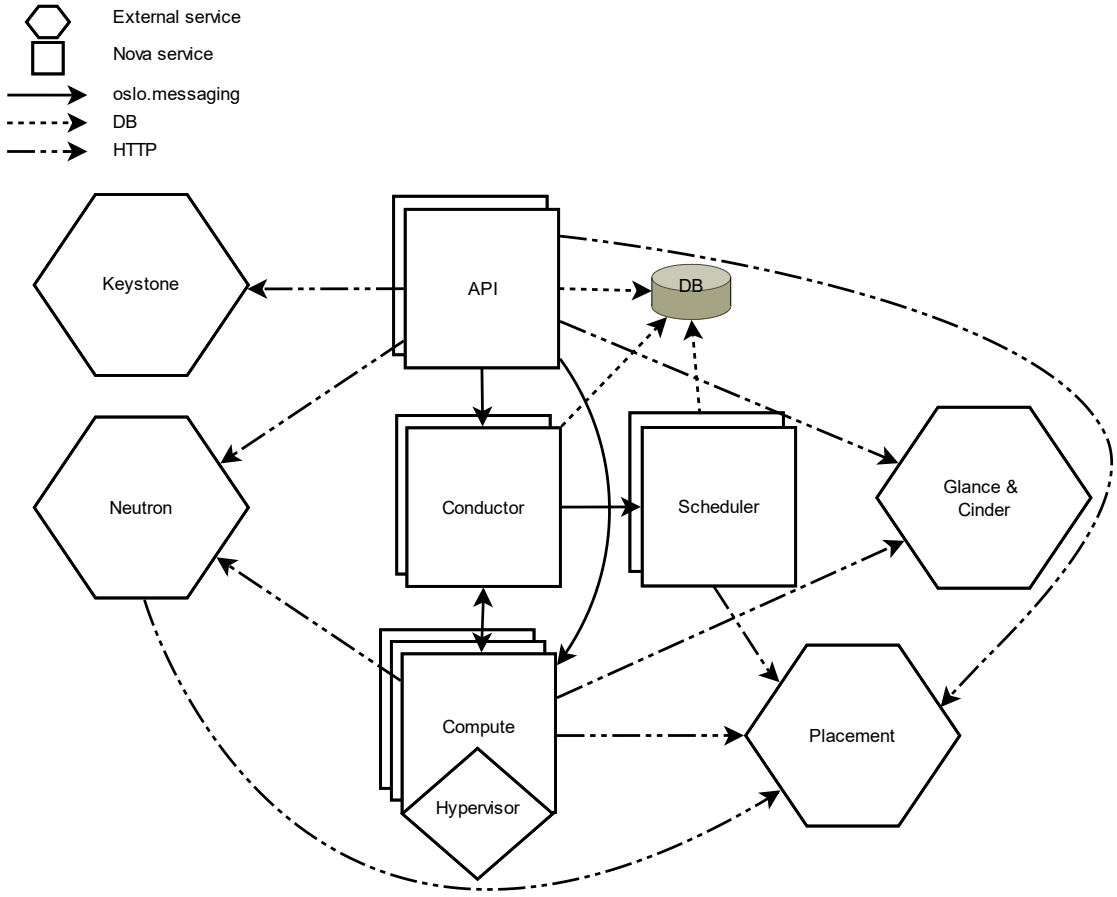


Ilustración 1 Diagrama de arquitectura de Nova

2.12.7. Neutron

Neutrón provee "Conectividad de red como servicio (NaaS)". El servicio de redes de OpenStack proporciona una API que permite construir topologías de red complejas, configurar y definir conectividad de la red y establecer políticas avanzadas de red y direccionamiento en la nube. Se encarga de la creación y gestión de una infraestructura

de red virtual, que incluye redes, switches, subnets y routers para dispositivos manejados por el componente Nova; también es posible utilizar servicios avanzados como firewalls o redes privadas virtuales (VPN). (Openstack, Neutron, 2021)

Neutron está integrado con los siguientes componentes:

- El servicio de identidad de OpenStack (Keystone): Utilizado para autenticación y autorización de peticiones al API. (Openstack, ML2 plug-in Neutron, 2021)
- El servicio de computación (Nova): Utilizado para conectar cada NIC virtual de la máquina virtual en una red concreta. (Openstack, ML2 plug-in Neutron, 2021)
- Panel de OpenStack (Horizon): Utilizado por administradores y usuarios de cada proyecto para crear y manejar servicios de red mediante una web UI. (Openstack, ML2 plug-in Neutron, 2021)

2.12.8. Cinder

Cinder es un servicio de almacenamiento en bloque de OpenStack, virtualiza el manejo de almacenamiento en bloques y provee a los usuarios una API para lograr solicitar y usar recursos sin necesidad de saber cómo se divide el almacenamiento de un dispositivo, esto puede lograrse mediante la implementación de referencia de LVM o controladores de complemento. (Openstack, Cinder block storage, 2020)

Cinder añade almacenamiento permanentemente a una máquina virtual, el hecho de que sea almacenamiento en bloques crea una infraestructura para manejar volúmenes, el servicio también permite la gestión de volúmenes instantáneos y tipos de volúmenes. (Openstack, Cinder block storage service overview, 2021)

El servicio de almacenamiento en bloque consta de componentes como:

- Cinder API
- Cinder volumen
- Cinder-scheduler Daemon
- Cinder backup Daemon
- Cola de mensajes (Openstack, Cinder block storage service overview, 2021)

2.12.9. Horizon

Horizon; es un proyecto basado en Django con la finalidad de proporcionar un marco extensible para crear nuevos Dashboards a partir de componentes reusables. Los

administradores y usuarios pueden acceder y gestionar otros servicios como Nova, Cinder, Neutron, Swift, etc. (Openstack, Horizon: the Openstack Dashboard project, 2021)

Horizon tiene tres tableros principales: usuario, sistema y ajustes, estos tres, en conjunto abordan las aplicaciones principales de OpenStack. Un cliente puede utilizar el Dashboard para aprovisionar los recursos de cómputo, red y memoria, impuestos por los administradores para implementar una infraestructura virtual.

Generalmente Horizon se sincroniza con Django, y soporta versiones de Python Django a partir de la consolidación de funciones de OpenStack. (Openstack, Supported Software, 2021)

2.13. Software adicional

2.13.1. UnRaid

Es un sistema operativo que permite optimizar los recursos del hardware, siendo capaz de servir como hospedador para máquinas virtuales, se instala mediante una memoria flash y consigo todos los datos necesarios para la configuración, tiene una interfaz intuitiva que facilita su uso, y permite al usuario personalizarlo de acuerdo a sus necesidades, siendo sus utilidades divididas en tres pilares: software definido NAS, servidor de aplicaciones y virtualización localizada. (Unraid, 2021)

Las funcionalidades de UnRaid son separadas en 3 partes: NAS, servidor de aplicaciones y virtualización.

UnRaid es un sistema operativo diseñado para proveer un entorno de virtualización, actúa como un servidor de aplicación y host de máquinas virtuales; UnRaid se instala y arranca de un dispositivo flash. Utilizando una versión reciente del Kernel de Linux con drivers de hardware actualizados, UnRaid es capaz de operar como cualquier sistema de 64 bits (x86_64) con el mínimo consumo de memoria del sistema. Toda la configuración relacionada al sistema operativo es almacenada en el dispositivo flash y cargado al mismo tiempo que el sistema operativo, la administración del sistema UnRaid puede ser realizado por medio de una interfaz web o línea de comandos; incluye versiones modernas de libvirt, VFIO, VirtIO, and VirtFS y soporta Open Virtual Machine Firmware (OVMF) lo cual permite el soporte de UEFI para las máquinas virtuales (SecureBoot y GPU pass through). (Unraid, 2021)

2.13.2. MariaDB

Es una de las bases de datos relacionales open source más populares, siendo parte de la mayoría de los servicios de nube y es la base de datos aplicada por defecto en la mayoría de las distribuciones Linux; los pilares de esta base de datos son la estabilidad, el buen desempeño y el compromiso de mantenerse open-source. (MariaDB Foundation, 2021)

2.13.3. RabbitMQ

Es el Broker de mensajes de código abierto más popular a nivel mundial, permitiendo la comunicación entre aplicaciones, sistemas y servicios, permite adaptar su configuración según las necesidades y se adapta a varios sistemas operativos y entornos de nubes, haciéndolo versátil y cómodo para el usuario. (RabbitMQ, 2021)

2.13.4. Memcached

Memcached es un sistema open-source de almacenamiento en memoria de objetos en caché, de alto rendimiento utilizado para mejorar la velocidad de las aplicaciones web, alivianando la carga de trabajo de la base de datos. Su diseño simple permite una implementación rápida y su API está disponible para los lenguajes más populares. (Memached, 2021)

2.13.5. Apache2

Apache es el servidor web más usado en todos los sistemas Linux, los servidores web son usados para mostrar páginas web solicitadas por el usuario mediante buscadores como Chromium o Firefox. Los usuarios ingresan una URL para ingresar a un sitio web, a su vez introducen un FQDN, indicando el recurso que necesitan de esa página. (Ubuntu, 2021)

2.13.6. WSGI

Es la interfaz que permite la comunicación entre el servidor y las aplicaciones, y cómo las aplicaciones pueden encadenarse en secuencia para cumplir una solicitud del usuario. (WSI, 2021) Las aplicaciones de WSGI se pueden apilar, aquellas que queden en la mitad deben implementar ambas partes de la interfaz de WSGI, tanto de la aplicación como del servidor, para que la aplicación que en la pila esté sobre esta actúe como servidor y la que está debajo actúe como aplicación. El servidor de WSGI recibe una petición del usuario, la cual pasa a la aplicación y cuya respuesta pasa de vuelta al usuario. (Intro, 2021)

2.13.7. LVM

El gestor de volúmenes lógicos es una herramienta para el manejo de volúmenes lógicos, incluye la asignación de discos, creación de bandas, duplicación y restablecimiento de volúmenes lógicos. Los volúmenes físicos se pueden colocar en dispositivos de bloque, que pueden abarcar dos o más discos. (Red Hat, 2021)

2.13.8. VAULT

Vault es un sistema de manejo de encriptados basado en la identidad, se basa en los secretos, un secreto es cualquier cosa a la que el usuario quiera tener un acceso controlado, para lo cual Vault provee servicios de encriptación dados por ciertos métodos de autenticación y autorización, al usar los API, HTTP, UI o CLI el acceso a los secretos puede ser controlado, restringido y auditable. Ya que los sistemas modernos requieren acceso a una gran cantidad de información, mucha de la cual puede considerarse secreta, es indispensable saber quién puede acceder a esa información, permitido por pocas plataformas, siendo Vault una herramienta que incrementa la seguridad del usuario. (Vault, 2020)

2.14. Personal Kanban

Personal Kanban: es un mecanismo simple y elegante que brinda una representación visual del trabajo. Permite gestionar el tiempo personal y al mismo tiempo compartir el progreso, objetivos y tareas pendientes. Esto conlleva una mejor planificación que requiere bajo mantenimiento y obtiene grandes resultados. Expone qué está sucediendo en el presente y los estados de las tareas (por realizar, completadas y atrasadas). Es flexible al momento de realizar cambios y útil para buscar previsibilidad en el trabajo mediante visualización y refinamiento. Todo esto es posible gracias al seguimiento y cumplimiento de 2 reglas: visualizar el trabajo y limitar el trabajo en (WIP) por sus siglas en inglés. (Benson, 2011)

2.15. Technology Acceptance Model (TAM)

El modelo de aceptación de tecnología (TAM), fue introducido en 1986 y continúa siendo el modelo teórico más aplicado para describir la aceptación de un de un sistema de

información. TAM puede ser aplicado en diversas situaciones definidas por (Lee, Kozar, & Larsen, 2003):

- Ambientales: tiempo y cultura
- Factores de control: género, tipo de organización y tamaño de esta.
- Sujetos: Estudiantes de pregrado, grado y trabajadores en genera.

Acorde con TAM, la actitud potencial en general de un individuo respecto al uso del sistema esta hipotetizado a ser el mayor determinante, aún si el usuario usa o no el sistema. La actitud respecto al uso (ATT) está dado por dos creencias principales: utilidad percibida (PU) y percepción de facilidad de uso (EOU). La percepción de facilidad de uso tiene un efecto causal en la utilidad percibida. El diseño del sistema afecta de manera indirecta a las 2 creencias. (Davis, 1985)

La percepción de facilidad de uso está definida como "el grado en el cual un individuo cree que utilizar un sistema en particular lo liberará de esfuerzo físico y mental". Además, tiene un efecto directo en la utilidad percibida ya que un sistema que es fácil de usar resultará en un mejor rendimiento de trabajo para el usuario. Si el usuario se vuelve más productivo por medio de la facilidad de uso, entonces se convierte más productivo en general, por lo que las características del sistema afectarán de forma indirecta a la utilidad de este, mediante la influencia de la facilidad de uso, en la siguiente ilustración 2 se muestra la aplicación del modelo TAM. (Davis, 1985)

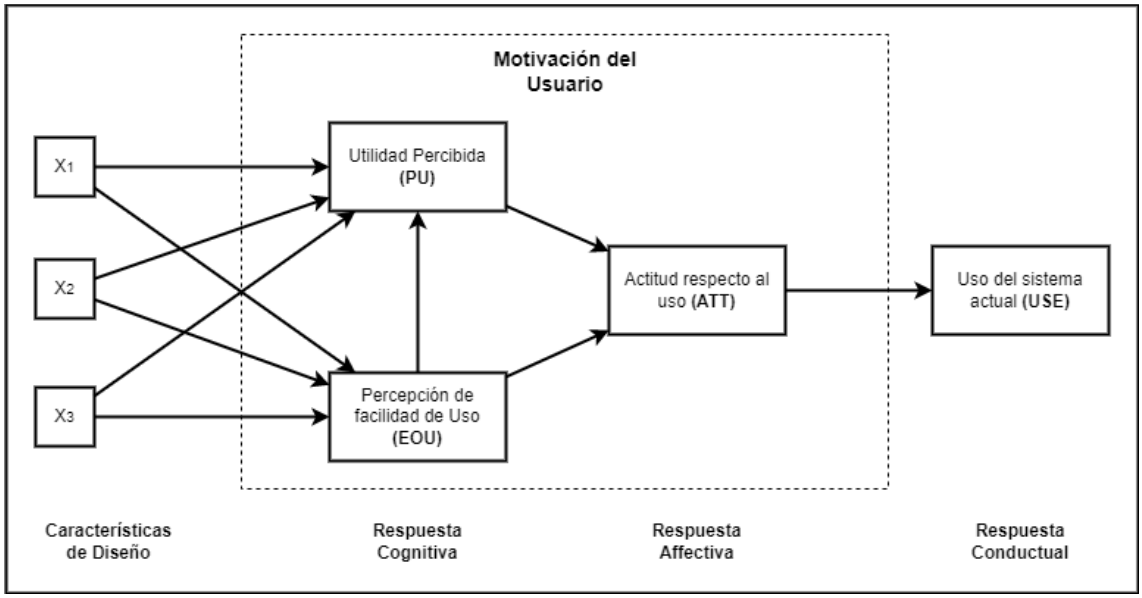


Ilustración 2 modelo de aceptación de Tecnología (TAM)

3. CAPÍTULO TERCERO: DESARROLLO DEL PROTOTIPO

El prototipo de Cloud Computing es una implementación de nube privada OpenStack diseñada para brindar IaaS, esta se compone de 5 nodos (máquinas virtuales) alojados en el servidor del laboratorio Smartlab de la Facultad de Sistemas, los cuales cumplen un rol en específico en el prototipo, el cual reside detrás de un proxy reverso que se encarga de redirigir las peticiones de los usuarios al proyecto. Gracias a la disponibilidad de una IP pública, es accesible vía Internet por medio de la URL www.smartlab-epn.com/horizon.

La implementación de OpenStack está conformado principalmente por 8 componentes, en donde es posible gestionar recursos de red, almacenamiento y procesamiento, para el correcto funcionamiento de los componentes se utiliza software adicional como: MariaDB, RabbitMQ, Memcached, LVM 2, entre otros; la instalación, configuración y uso se describirá más adelante.

3.1. Hardware

OpenStack está diseñado para brindar Infraestructura como Servicio a una o varias organizaciones, por lo que los requisitos de recursos computacionales y red están lejos de ser cumplidos en el desarrollo del prototipo actual, el hardware utilizado para la implementación del prototipo son un router y el servidor que serán descritos a continuación.

3.1.1. Router

El router brinda acceso a Internet por medio de la red de la Universidad a toda la infraestructura del laboratorio y expone los servicios del prototipo por medio de port-forwarding, la configuración y especificaciones generales se enlistan en las tablas 2, 3 y 4 a continuación:

Tabla 2 Configuración general del Router

Router	
Tipo de Conexión WAN	IP Estática
IP WAN	172.31.97.40/24
WAN Default Gateway	172.31.97.1
DNS	8.8.8.8, 8.8.4.4
IP LAN	192.168.0.1

Tabla 3 Especificaciones generales del Router

Router	
Nombre	ipTIME
Modelo	A3004NS-M
CPU	MediaTek MT7621 Dual Core
CPU Clock	880 MHz
FLASH	16 MB
Versión del Firmware	10.08.4
NAT	NAT Hardware
WAN	1 puerto (gigabit)
LAN	4 puertos (gigabit)

Tabla 4 Servicios de Port-Forwarding

Servicios Port-forwarding			
Nombre	LAN IP	Puerto Externo	Puerto Interno
openstack-dashboard	192.168.0.120	80	80

SSL	192.168.0.120	443	443
openstack-vnc	192.168.0.120	64002	6080
ssh master	192.168.0.120	64001	22

3.1.2. Servidor

El servidor cuenta con los recursos necesarios para brindar almacenamiento, virtualizar redes y ejecutar las máquinas virtuales (nodos de OpenStack y proxy reverso) necesarias para ejecutar el prototipo OpenStack. El servidor ejecuta el Hypervisor tipo 1 UnRaid, el cual utiliza Quick EMUlator (QEMU) junto con Kernel virtual-machine (KVM), la información general del sistema de Unraid se muestra en la ilustración a continuación en la ilustración 3.

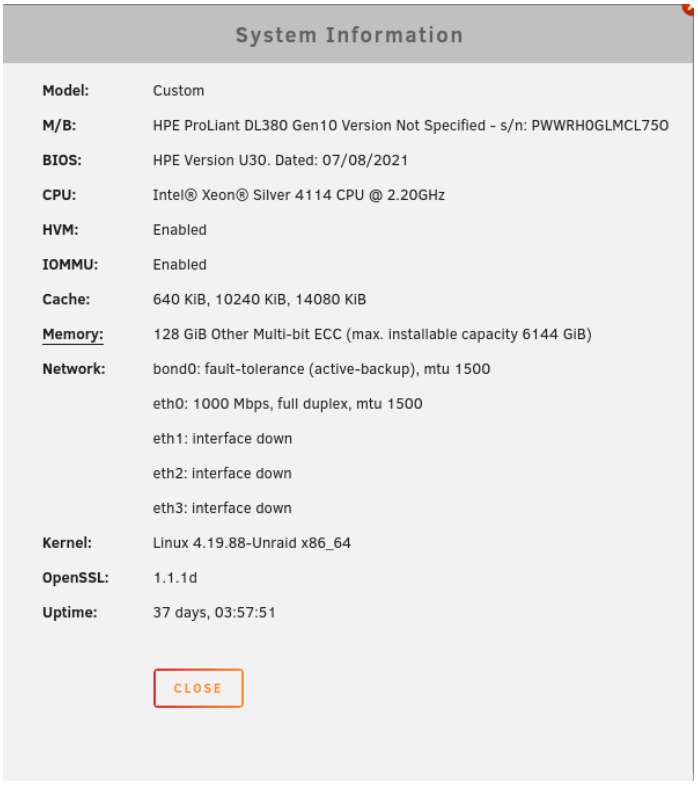


Ilustración 3 Información general del Sistema obtenido del Dashboard de Unraid

3.1.2.1. Módulo kvm para nested virtualization

El nodo de cómputo de OpenStack se encargará de ejecutar las máquinas virtuales de los proyectos, por esa razón, es necesario habilitar la virtualización anidada (nested) y así, permitir virtualizar dentro de los nodos de OpenStack virtualizados, la virtualización se muestra a continuación en la ilustración 4.

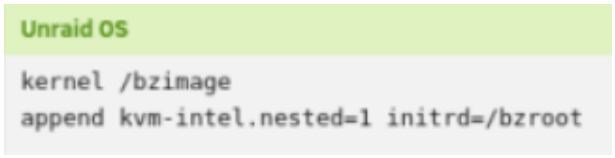


Ilustración 4 Argumento para habilitar nested virtualization al momento de arrancar el Kernel

3.1.2.2. Memoria

Cuenta con la cantidad de memoria RAM de 128 GiB de 4 generación (DDR4) de los cuales 125.6 GiB de memoria física son utilizables con corrección de errores de código múltiple (EEC), la especificación de la cantidad y uso de memoria se muestran a continuación en la ilustración 5.

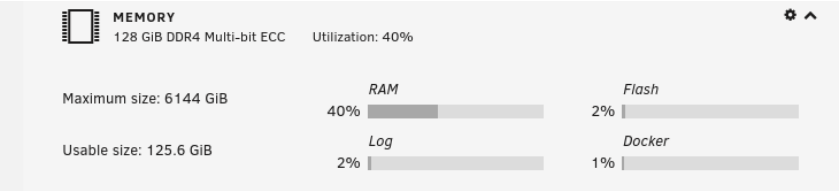


Ilustración 5 Especificación de la cantidad y uso de memoria

3.1.2.3. Red

El servidor cuenta con 4 puertos Gigabit ethernet, solamente 1 puerto está en uso y está conectado físicamente al router por medio de conexión ethernet, la especificación de la interfaz de red habilitada se muestra en la ilustración 6.

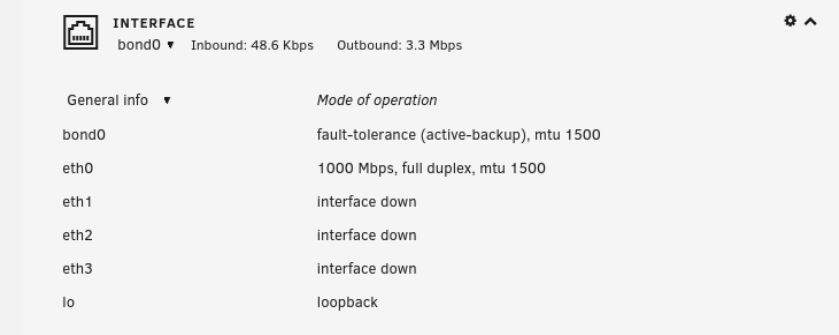


Ilustración 6 Especificación de la interfaz de red habilitada

3.1.2.4. Almacenamiento

UnRaid permite el uso de discos de almacenamiento de diferentes tamaños para crear un arreglo de discos heterogéneos con un disco dedicado a la paridad para la recuperación de información en el caso de que uno de los discos falle, en la siguiente ilustración se enlistan las 7 unidades de almacenamiento en la ilustración 7.

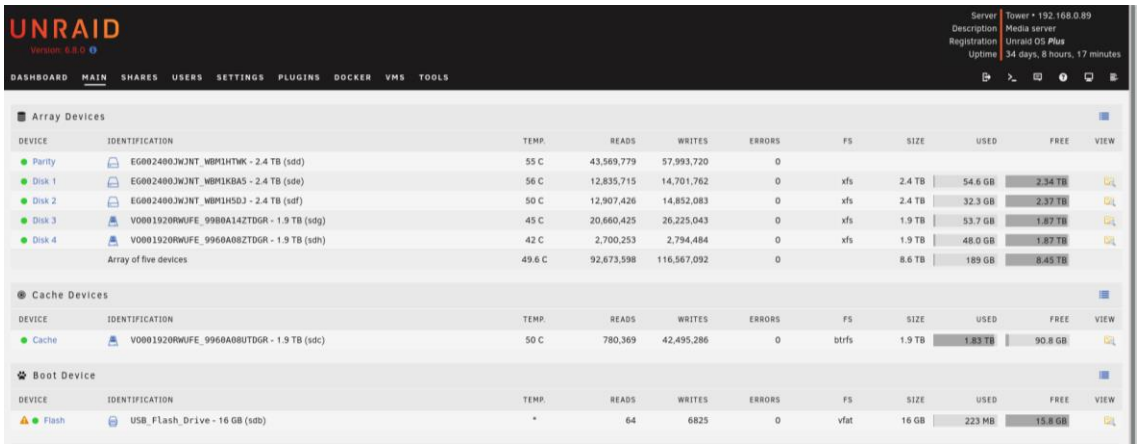


Ilustración 7 Página principal del Dashboard de UnRaid en donde se enlistan las 7 unidades de almacenamiento

El servidor cuenta con 7 unidades de almacenamiento, estos se dividen en 3 discos duros de 2.4 TB, 3 discos de estado sólido de 1.9 TB y 1 memoria flash de 16 GB; De las 7 unidades de almacenamiento solo 2 discos duros y 2 discos de estado sólido son utilizados para almacenamiento. Los demás desempeñan otras funcionalidades necesarias para la ejecución optima del arreglo de discos; el disco duro restante se encarga de realizar paridad, mientras que el disco de estado sólido restante sirve de cache, por último, la memoria flash el hyperivsor UNRAID.

El disco de paridad evita la pérdida de datos cuando 1 disco del arreglo se daña, y el disco de cache aumenta el rendimiento del sistema en general cuando se realizan operaciones de escritura, el almacenamiento utilizable del sistema es de 8.6 TB.

3.1.2.5. Procesamiento

El servidor cuenta con un procesador Intel Xeon de 10 núcleos de la línea Silver con una frecuencia de 2.20 GHz y hyperthreading. El Hyperthreading permite al hypervisor duplicar el número de cores, por ende existirán 20 cores disponibles para virtualizar, a continuación se especifican los 10 cores físicos con Hyperthreading en la ilustración 8.

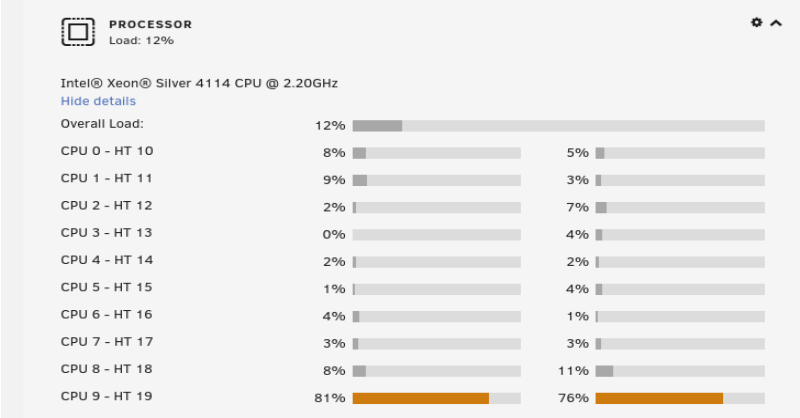


Ilustración 8 Especificación de uso de los 10 cores físicos con Hyperthreading

3.2. Red

El router y el servidor son los únicos componentes que forman parte de la red LAN para el proyecto. La interfaz de red física del servidor tiene la dirección IP 192.168.0.89. Todos los nodos requieren acceso a Internet para cumplir con las actividades de administración de sistemas como instalación de paquetes, actualizaciones de seguridad, DNS y NTP. El prototipo utiliza la red LAN y una red virtual que serán descritas a continuación en la ilustración 9.

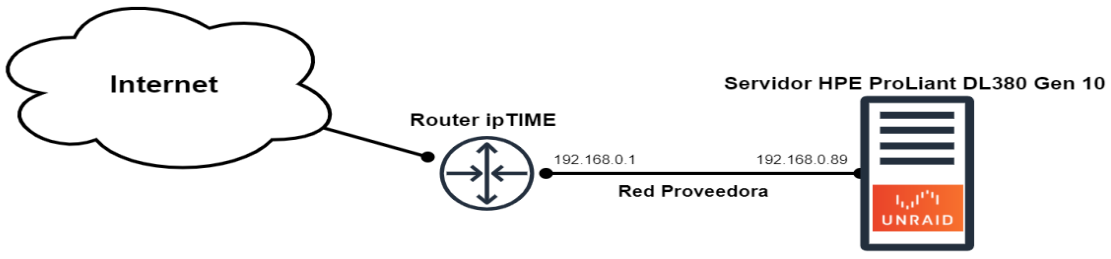


Ilustración 9 Red y componentes físicos del proyecto

3.2.1. Red Proveedora

Esta red es gestionada por el router y engloba a todos los dispositivos físicos del laboratorio Smartlab. Es una LAN física privada clase C con notación CDIR 192.168.0.0/24 y tiene acceso a internet. OpenStack utiliza el rango desde 192.168.0.216 hasta 192.168.0.254 como pool de IP flotantes que están disponibles para exponer a la red LAN a las máquinas virtuales que residen dentro de la red virtual de OpenStack.

3.2.2. Red de Administración

UnRaid utiliza la red NAT predeterminada de libvirt como una red interna virtual privada de clase C 192.168.122.0/24. Esta red permite la comunicación interna entre los nodos de OpenStack y el proxy reverso y está dedicada exclusivamente para el proyecto. También tiene acceso a internet por medio de un puente virtual conectado al puente principal de la interfaz de red física, a continuación, se describe la topología física y virtual de la red proveedora y administración en la ilustración 10.

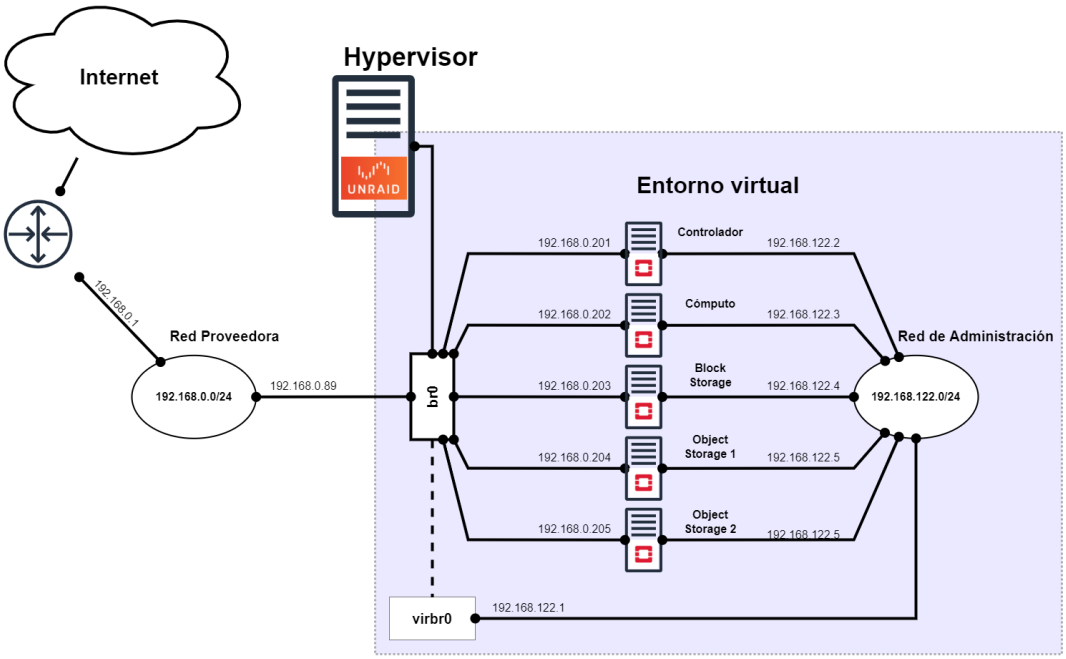


Ilustración 10 Topología física y virtual de la red Proveedora y Administración.

Como se puede observar en la figura anterior, la interfaz física de red del servidor representa 5 IPs virtuales adicionales correspondientes a los nodos del prototipo.

3.3. Sistema Operativo

OpenStack puede ser implementado en cualquier sistema operativo GNU/Linux que cumpla con las dependencias de software de cada componente y tenga instalado Python 3; los SO que están incluidos en las guías de instalación oficiales de cada componente son:

- Ubuntu
- CentOS o RHEL 8
- OpenSUSE Leap o SUSE Linux Enterprise
- Debian GNU/Linux

Es importante destacar que la elección del sistema operativo es muy importante, ya que las últimas actualizaciones de seguridad, mantenimiento y actualización de software están directamente relacionadas con el método de instalación.

El sistema operativo seleccionado para la instalación de OpenStack es Ubuntu 20.04 LTS Focal Fossa (server install image) https://releases.ubuntu.com/20.04/. La razón principal de por qué se seleccionó este SO radica en la cantidad de pruebas realizadas en DevStack https://docs.openstack.org/devstack/latest/ y por tanto existen menos errores al momento de implementarlo. Por otro lado, al ser una versión LTS existe un soporte extendido. A continuación, se describirá las razones de por qué se han descartado las demás distribuciones:

CentOS Linux 8: CentOS ha anunciado su EOL (end of life) para diciembre 31 del 2021 https://www.centos.org/centos-linux-eol/, por lo que ya no será una reconstrucción de RHEL.

RHEL 8: La distribución RHEL 8 requiere de un pago por la subscripción de actualizaciones.

OpenSUSE: OpenSUSE no mantiene actualizado el repositorio de la última versión estable de OpenStack (Xena). Además, no cuenta con una versión LTS. https://build.opensuse.org/project/subprojects/Cloud:OpenStack
https://en.opensuse.org/Lifetime

SUSE Linux Enterprise: SUSE requiere de un pago por la subscripción de actualizaciones.

Debian GNU/Linux: No todas las guías de instalación contienen las instrucciones para instalar en Debian GNU/Linux.

3.4. Nodos de OpenStack

El prototipo de OpenStack está formado por 5 nodos. Cada nodo tiene asignado una diferente cantidad recursos dependiendo de las funciones de este, se muestra en la ilustración 11 la reserva y organización de recursos del servidor para los 5 nodos.

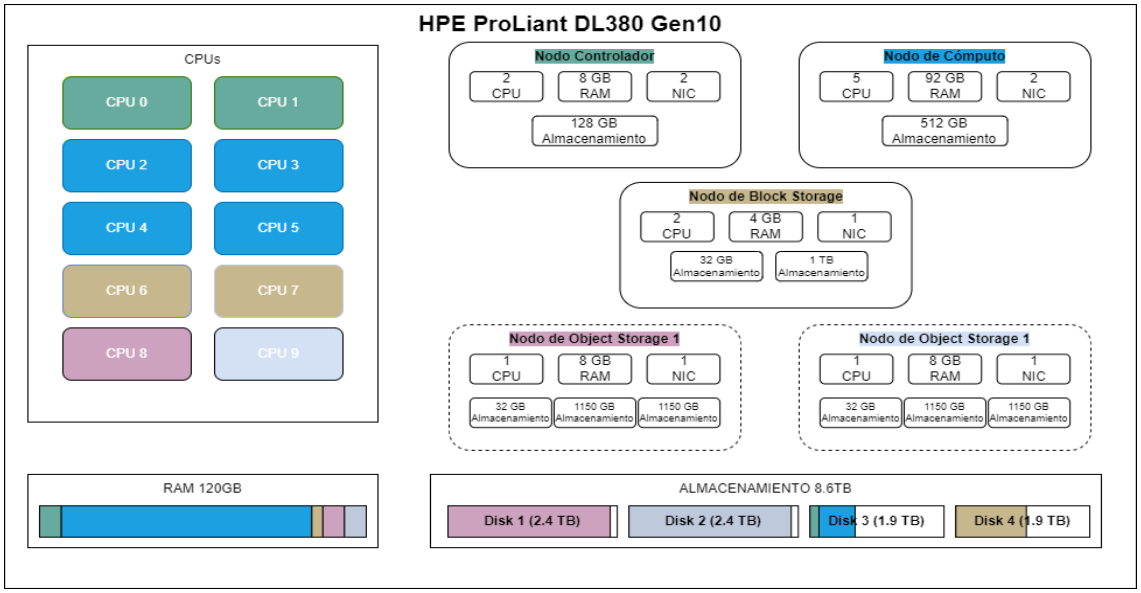


Ilustración 11 Reserva y organización de recursos del servidor para los 5 nodos

3.4.1. Instalación y configuración de los nodos

3.4.1.1. Nodo Controller

Encargado de la gestión de los demás nodos y de ejecutar de forma centralizada la administración de los componentes y el servicio de identidad. Ejecuta todos los servicios de OpenStack de forma total o parcial e incluye servicios de apoyo como una base de datos SQL (MariaDB), cola de mensajes (RabbitMQ) y NTP. Tiene asignado 2 CPU físicos, 8 GB de memoria RAM, 2 interfaces de red (red proveedora y de administración) y 1 disco de almacenamiento de 128 GB. https://docs.openstack.org/install-guide/overview.html

3.4.1.1.1. *Configuración del nodo en UnRaid*

Establecer el nombre del nodo **controller** como: **Ubuntu OpenStack Controller.** Especificar los 4 CPUS lógicos que utilizará el nodo y reservar 8 GB de memoria RAM, la ilustración 12 muestra el establecimiento del nombre del nodo controller.

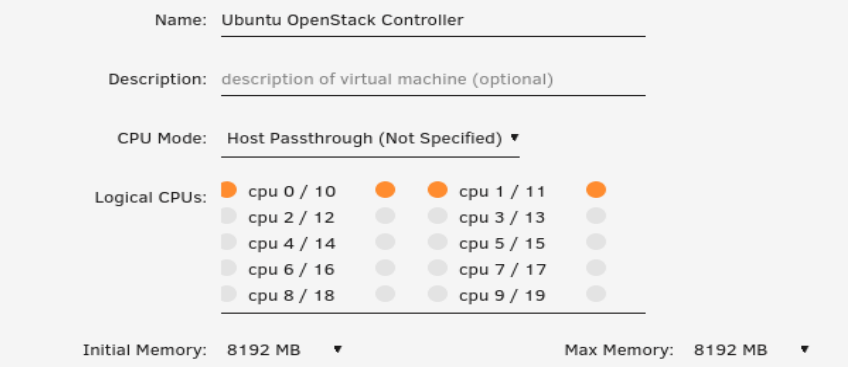


Ilustración 12 Establecimiento del nombre del nodo controller

Se muestra la Creación del disco de almacenamiento tipo raw de 128 GB en el disco físico 3 en la ilustración 13.

Ilustración 13 Creación del disco de almacenamiento tipo raw

La ilustración 14 muestra la asignación al nodo controller una interfaz de red para la red LAN física (red proveedora) y otra interfaz de red para la red virtual interna (red de administración).

Ilustración 14 Asignación al nodo controller una interfaz de red para la red LAN

3.4.1.2. Nodo Compute

Actúa como Hipervisor, donde se ejecutan las máquinas virtuales. Tiene asignado 4 CPU físicos, 92 GB de memoria RAM, 2 interfaces de red (red proveedora y de administración) y 1 disco de almacenamiento de 512 GB.

3.4.1.2.1. *Configuración del nodo en UnRaid*

Establecer el nombre del nodo **compute** como: **Ubuntu OpenStack Compute.** Especificar los 8 CPUS lógicos que utilizará el nodo y reservar 92 GB de memoria RAM, se indica en la ilustración 15 el establecimiento del nombre del nodo compute.

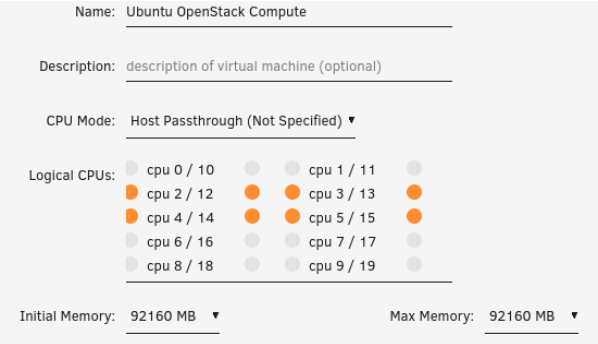


Ilustración 15 Establecimiento del nombre del nodo compute

Se muestra la creación del disco de almacenamiento tipo raw de 512 GB en el disco físico 3 en la ilustración 16.

Ilustración 16 Creación del disco de almacenamiento tipo raw de 512 GB

Se visualiza en la ilustración 17 la asignación al nodo compute una interfaz de red para la red LAN física (red proveedora) y otra interfaz de red para la red virtual interna (red de administración)

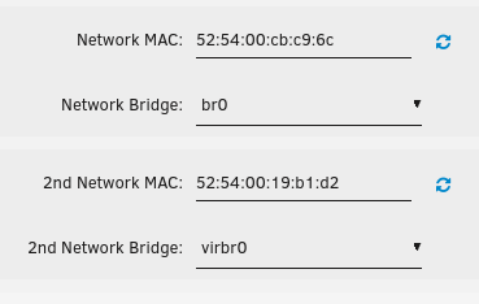


Ilustración 17 Asignación al nodo compute una interfaz de red para la red LAN

3.4.1.3. Nodo Block Storage

Alberga los volúmenes reservados para las instancias virtualizadas en el nodo de cómputo. Tiene asignado 2 CPU físicos, 4 GB de memoria RAM, 1 interfaz de red (red de administración) y 2 discos de almacenamiento, uno de 32 GB y uno de 1000 GB para alojar el sistema operativo y almacenar los discos virtuales respectivamente.

3.4.1.3.1. *Configuración del nodo en UnRaid*

Establecer el nombre del nodo **storage** como: **Ubuntu OpenStack Storage.** Especificar los 4 CPUS lógicos que utilizará el nodo y reservar 4 GB de memoria RAM, en la ilustración 18 se visualiza el establecimiento del nombre del nodo storage.

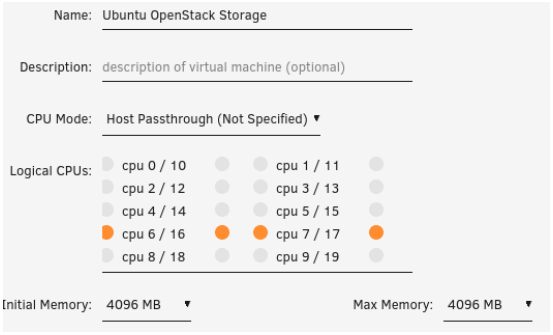


Ilustración 18 Establecimiento del nombre del nodo storage

Crear los discos de almacenamiento tipo raw de 32 GB y 1 TB en el disco físico 4 respectivamente. El primer disco se utilizará para instalar el sistema operativo y el segundo almacenará los volúmenes virtuales creados por el servicio de almacenamiento en bloques (Cinder), la creación de los discos de almacenamiento se muestra en la ilustración 19, creación de los discos de almacenamiento tipo raw.

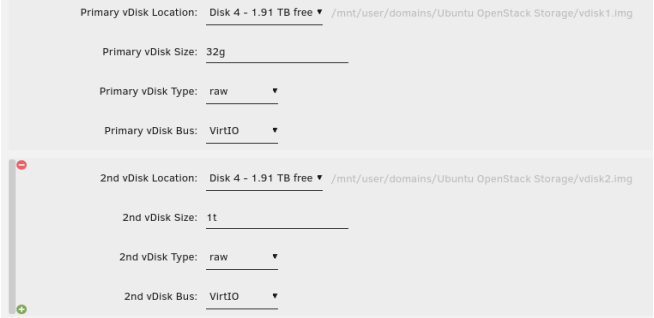


Ilustración 19 Creación de los discos de almacenamiento tipo raw de 32 GB y 1 TB en el disco físico

Asignar al nodo compute una interfaz de red para la red virtual interna (red de administración), que se visualiza en la ilustración 20 de asignación de interfaz al nodo compute.

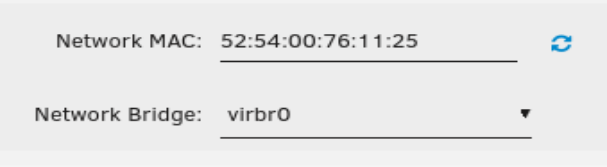


Ilustración 20 Asignación al nodo compute una interfaz de red para la red virtual interna

3.4.1.4. Nodos Object Storage

Encargados de almacenar archivos de forma redundante y replicada como: imágenes de sistema, snapshots, backups de discos y archivos pequeños en general que los usuarios necesiten almacenar para uso personal. Cada nodo tiene asignado 1 CPU físico, 8 GB de memoria RAM, 1 interfaz de (red de administración) y 3 discos de almacenamiento, uno de 32 GB y dos de 1150 GB para alojar el sistema operativo y almacenar archivos (objetos) con Swift respectivamente.

3.4.1.4.1. *Nodo Object Storage 1*

Miembro del clúster de object storage. Aquí se guardarán los backups, imágenes y archivos que los usuarios necesitan para operar en la nube.

3.4.1.4.1.1. Configuración del nodo en UnRaid

Establecer el nombre del nodo **object-1** como: **Ubuntu OpenStack Object 1.** Especificar los 2 CPUS lógicos que utilizará el nodo y reservar 8 GB de memoria RAM, se visualiza en la ilustración 21, establecimiento del nombre del nodo object.

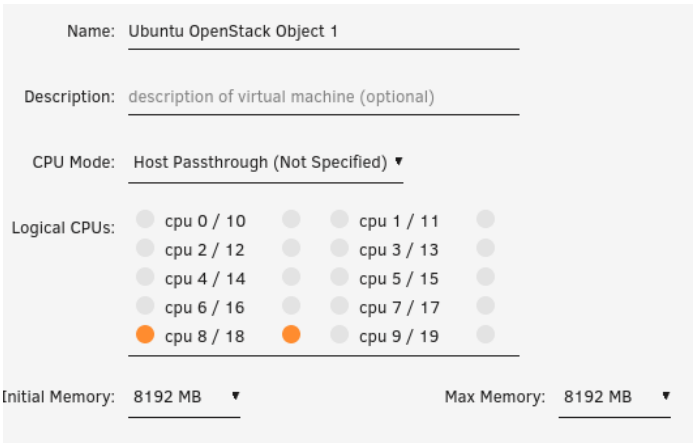


Ilustración 21 Establecimiento del nombre del nodo object-1

Crear los discos de almacenamiento tipo raw de 32 GB, 1150 GB y 1150 GB. El primer disco se utilizará para instalar el sistema operativo, el segundo y el tercero pertenecerán al clúster de servicio de almacenamiento por objectos (Swift), la creación de los discos de almacenamiento se evidencia en la ilustración 22.

Ilustración 22 Creación de los discos de almacenamiento tipo raw de 32 GB, 1150 GB y 1150 GB

La asignación al nodo compute una interfaz de red para la red virtual interna (red de administración) se muestra en la ilustración 23.

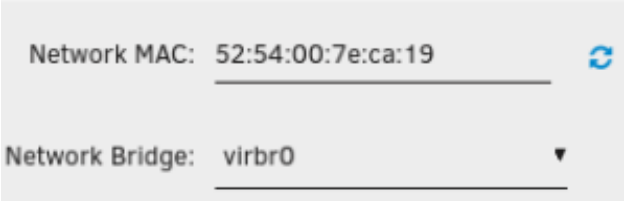


Ilustración 23 Asignación al nodo compute una interfaz de red para la red virtual interna

3.4.1.4.2. *Nodo Object Storage 2*

Miembro del clúster de object storage. Aquí se guardarán los backups, imágenes y archivos que los usuarios necesitan para operar en la nube.

3.4.1.4.2.1. Configuración del nodo en UnRaid

Establecer el nombre del nodo **object-2** como: **Ubuntu OpenStack Object 2.** Especificar los 2 CPUS lógicos que utilizará el nodo y reservar 8 GB de memoria RAM, el establecimiento del nombre del nodo object 2 se visualiza en la ilustración 24.

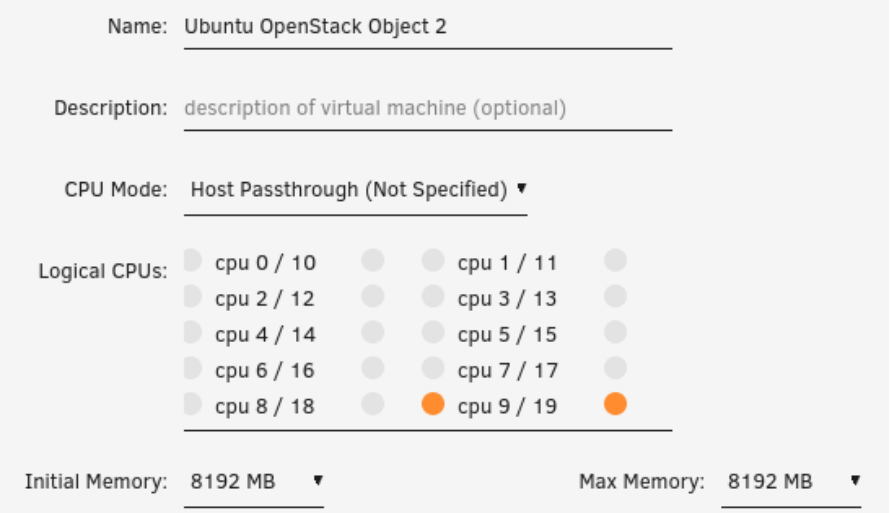


Ilustración 24 Establecimiento del nombre del nodo object-2

Crear los discos de almacenamiento tipo raw de 32 GB, 1150 GB y 1150 GB en el disco físico 1 respectivamente. El primer disco se utilizará para instalar el sistema operativo, el segundo y el tercero pertenecerán al clúster de servicio de almacenamiento por objectos (Swift), la creación de los discos de almacenamiento tipo Raw se muestra en la ilustración 25.

Ilustración 25 Creación de los discos de almacenamiento tipo raw de 32 GB, 1150 GB y 1150 GB en el disco físico 1 respectivamente

La asignación al nodo compute una interfaz de red para la red virtual interna (red de administración) se indica en la ilustración 26.

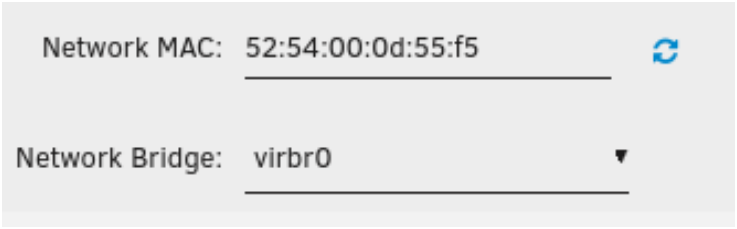


Ilustración 26 Asignarción al nodo compute una interfaz de red para la red virtual interna

3.5. Componentes OpenStack

El proyecto está conformado por 8 componentes. La implementación de OpenStack más simple funcional está conformada por 5 componentes: Nova, Neutron, Keystone, Glance y Placement. Se incluyeron adicionalmente más componentes para brindar servicio de almacenamiento y cliente web. Estos componentes son: Cinder, Swift y Horizon, que se muestran en la ilustración 27.

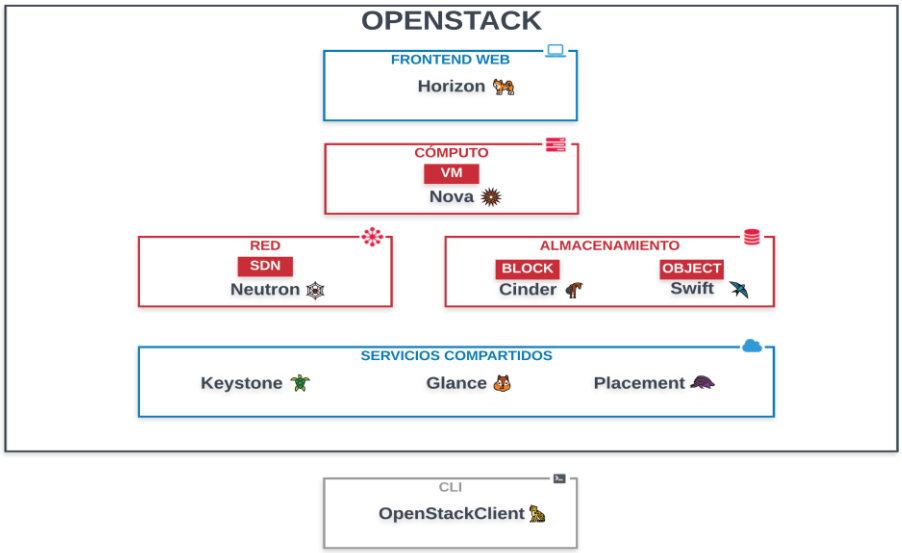


Ilustración 27 Componentes OpenStack que conforman el prototipo de Cloud computing

Cada nodo ejecuta uno o más componentes de forma total y parcial. Existen componentes que están divididos en servicios según la funcionalidad, por ejemplo: el servicio proxy de Swift, encargado de redirigir las peticiones a los nodos Object Storage, se ejecuta en el Nodo Controlador, pero el servicio de almacenamiento de objetos en sí se ejecuta en los nodos de Object Storage; la instalación de los componentes de cada nodo se muestra en la ilustración 28.

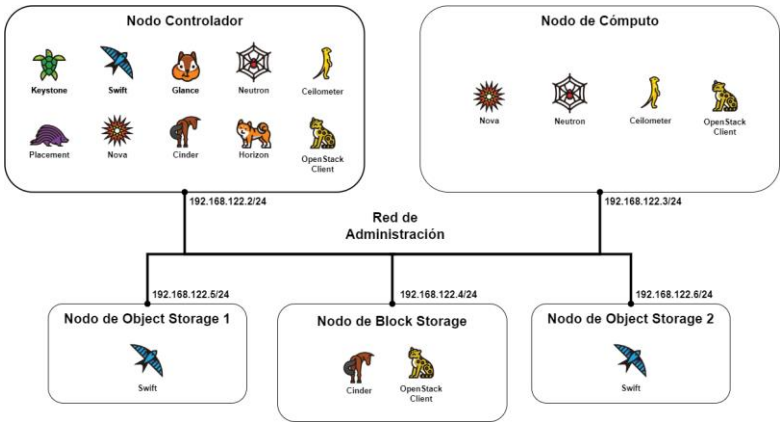


Ilustración 28 Instalación de componentes en cada Nodo

3.5.1. Keystone

El servicio de Identidad Keystone almacena toda la información la base de datos **keystone** creada en MariaDB. El servicio es accesible dentro de la red de administración con la URL http://controller:5000/v3 mediante el servicio HTTP apache2. Todos los proyectos pertenecen al dominio **default** y a la región **RegionOne**. Los Endpoints **admin**, **internal** y **public** de cada servicio residen en la misma red con la misma URL http://controller.

Cada componente está representado por su respectivo usuario y contraseña con rol **admin** que pertenece al proyecto de servicios llamado **service**. Keystone utiliza el repositorio de tokens fernet y los provee a los componentes que se han autenticado. A su vez, los componentes utilizan el token para validar que tienen los permisos respectivos para realizar operaciones sobre otros componentes. Por otro lado, el usuario puede hacer uso del cliente web Horizon o el CLI de OpenStack para realizar operaciones en Keystone, la interacción entre componentes se evidencia en la ilustración 29.

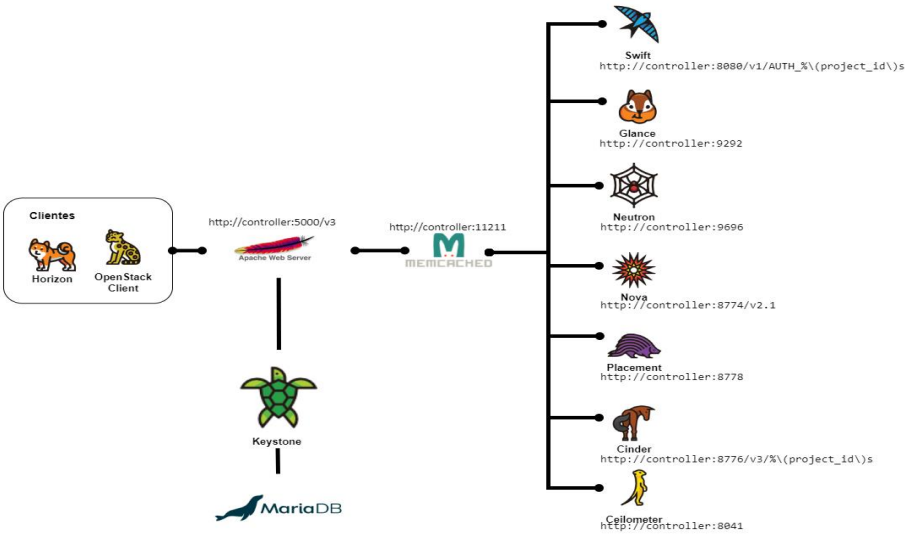


Ilustración 29 Interacción entre Keystone y los demás componentes.

3.5.2. Swift

El nodo controlador se encarga de servir como proxy del clúster de almacenamiento de objetos. El servicio está expuesto con la URL http://controller:8080/v1/AUTH_%\(project_id\)s y registrado en el catálogo como servicio object-store.

3.5.2.1. Anillos del clúster

Los 3 anillos tienen un número máximo de 1024 particiones, con 3 réplicas de cada objeto y 12 horas como tiempo mínimo para mover una partición. El clúster de almacenamiento de objetos utiliza los rings para cuenta, contenedor y objecto.

3.5.2.2. Número de particiones máximas

El clúster está formado por 4 discos (2 por nodo). Cada disco puede tener hasta 256 particiones, lo que equivale a un total de 1024 particiones.

3.5.2.3. Número de réplicas

Cada objeto tiene 3 copias almacenadas y distribuidas en lo mayor posible en los discos del clúster por región. Esto garantiza una mayor probabilidad en la disponibilidad de los datos en caso de que uno o más discos del clúster fallen.

3.5.2.4. Zonas

El clúster tiene 2 zonas y cada zona tiene 2 discos. En ambientes de producción la división por zonas es geográfica, en este caso es lógica ya que el clúster reside dentro del mismo servidor, la ilustración 30 muestra el diagrama de implementación de Swift.

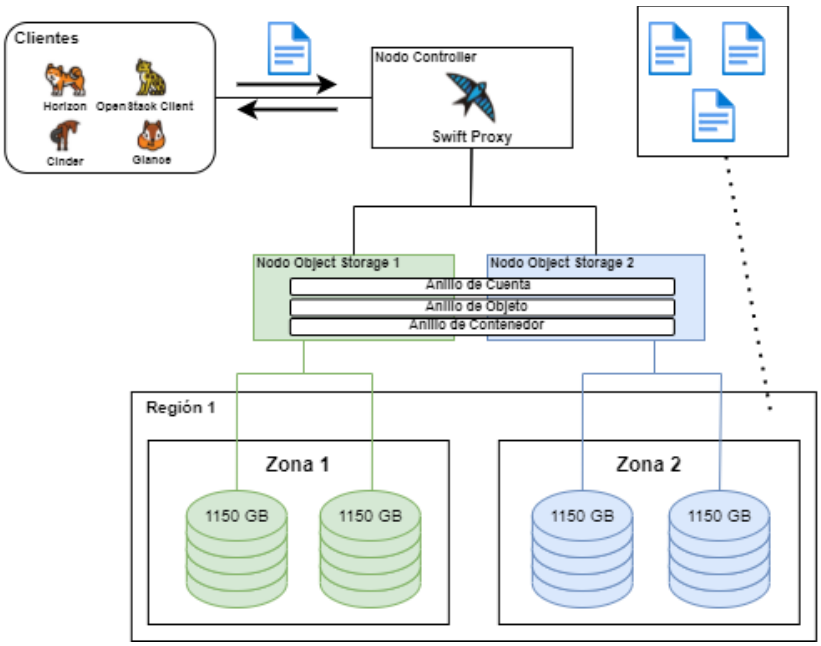


Ilustración 30 Diagrama de implementación de Swift

3.5.3. Glance

El servicio de imagen utiliza el servicio de almacenamiento de objetos (Swift) como backend para guardar las imágenes de sistema, está expuesto con la URL http://controller:9292 y registrado en el catálogo como servicio **image**. Las imágenes son transmitidas por medio de HTTP.

Los usuarios pueden cargar y descargar imágenes que el servicio de cómputo Nova virtualiza. El servicio Swift se encarga de identificar el tamaño en megabytes de la imagen que va a ser almacenada. Si el tamaño es mayor a 100 megabytes, la imagen es dividida en segmentos de 75 megabytes, en donde cada segmento es almacenado como parte de un objeto (la imagen) en el backend de almacenamiento. Swift se encarga de reconstruir la imagen con los segmentos divididos almacenados cuando el servicio de imagen lo requiera, esto evita que el backend de Almacenamiento responda con un Request Timeout, la ilustración 31 muestra el diagrama de implementación de glance.

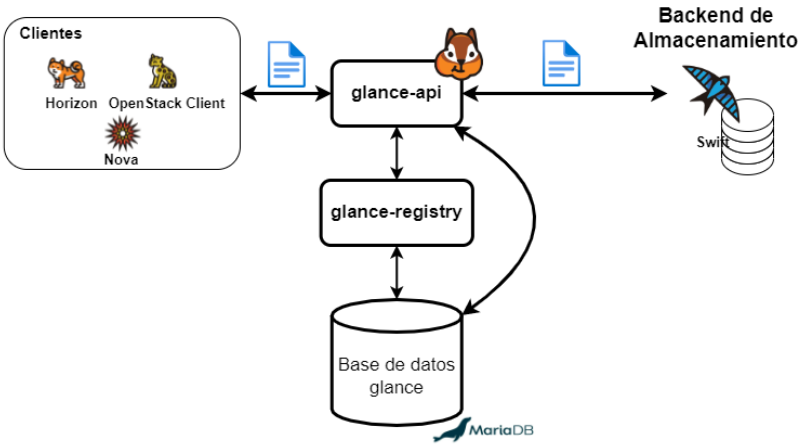


Ilustración 31 Diagrama de implementación de Glance

3.5.4. Neutron

El servicio de red está expuesto con la URL http://controller:9696 y registrado en el catálogo como servicio **network**. Está dividido por agentes que brindan servicios de DHCP, routing, bridging, metadatos, entre otros.

Neutron tiene la opción de que los usuarios trabajen sobre la red proveedora o permitir que puedan crear sus propias redes subredes y routers a nivel de proyecto. Estas pueden estar conectadas a la red proveedora para obtener acceso a internet. El

prototipo utiliza la segunda opción llamada redes Self-Service, la cual utiliza túneles VXLAN para poder virtualizar las redes a través de la red de administración.

El nodo Controlador ejecuta el Agente L3, Agente Linux Bridge, Agente de DHCP, y Agente de metadatos, mientras que el nodo de Cómputo se ejecuta el agente de Linux Bridge; la ilustración 32 muestra el nodo controlador y de cómputo con la opción de redes self-service.

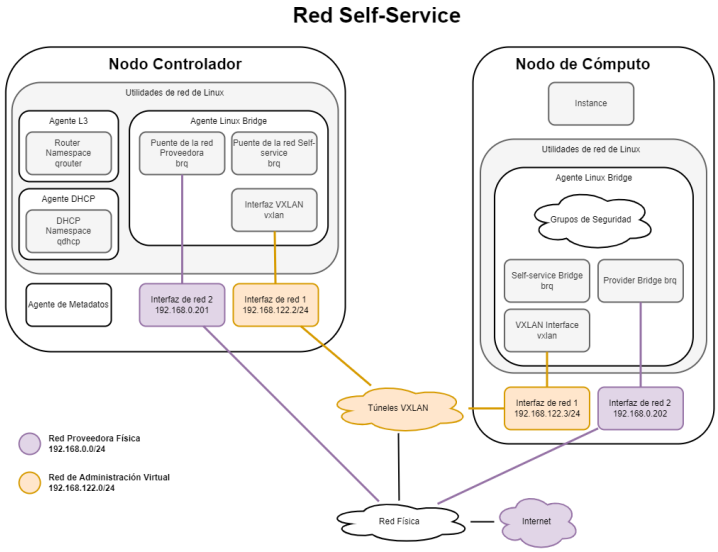


Ilustración 32 Nodo controlador y de cómputo con la opción de Redes Self-Service

Las instancias pueden ser ejecutadas en la red proveedora o en la red Self-Service. Cada red tiene su propio puente conectado a cada una de las interfaces red del nodo controlador y de cómputo, (red proveedora y red de administración) respectivamente. Cada red tiene servicio de DHCP por separado en su propio namespace, mientras que los routers y el servidor de metadatos residen dentro del mismo namespace compartido por las dos redes.

Las interfaces de red de una instancia están conectadas al puente de red por medio de una interfaz TAP. El puente de red tiene una interfaz VXLAN para virtualizar las redes entre el nodo de Cómputo y Controlador; la ilustración 33 evidencia la conectividad entre redes con opciones de redes self-service.

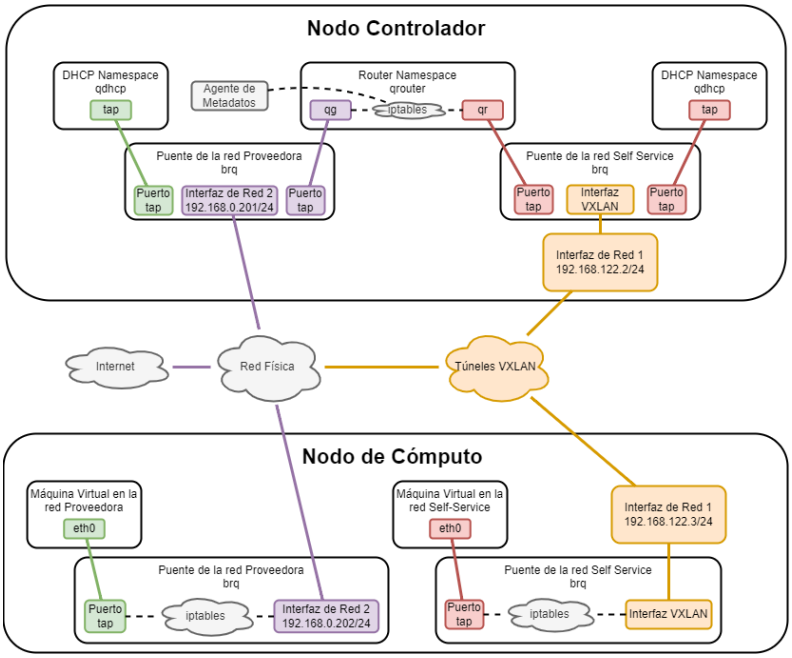


Ilustración 33 Conectividad entre redes con la opción de Redes Self-Service

3.5.5. Nova

El componente Nova interactúa directamente con todos los componentes a excepción del servicio de almacenamiento de objetos Swift. Nova utiliza la separación lógica de celdas V2 para agrupar los nodos de cómputo, el prototipo tiene las celdas cell0 y cell1 como bases de datos adicionalmente a la base de datos principal de nova api_db. La primera celda es considerada la celda falsa y ningún nodo de cómputo pertenece a esa celda, la razón de la existencia esta es brindar retrocompatibilidad con implementaciones anteriores de Nova, por otro lado, el nodo de cómputo pertenece a la celda cell1.

Todos los servicios de Nova se ejecutan en el Nodo controlador a excepción del servicio Compute. Sus funcionalidades son descritas a continuación:

- API (nova-api): Es un demonio que sirve a las APIs de metadatos y compute en greenthreads separadas.
- Conductor (nova-conductor): Servicio que gestiona las peticiones de los nodos de cómputo que van dirigidas a la base de datos.
- NoVNC Proxy (nova-novncproxy): Proxy de websockets que es compatible con las consolas noVNC de Nova para las instancias que se ejecutan en el nodo de cómputo.

- Scheduler (nova-scheduler): Encargado de escoger el nodo de cómputo para ejecutar las instancias.
- Compute (nova-compute): Encargado de recibir y procesar las peticiones de los servicios API y Conductor e interactuar con el Hipervisor, para realizar la virtualización en sí, la ilustración 34 muestra el diagrama de nova en los nodos controlador y cómputo.

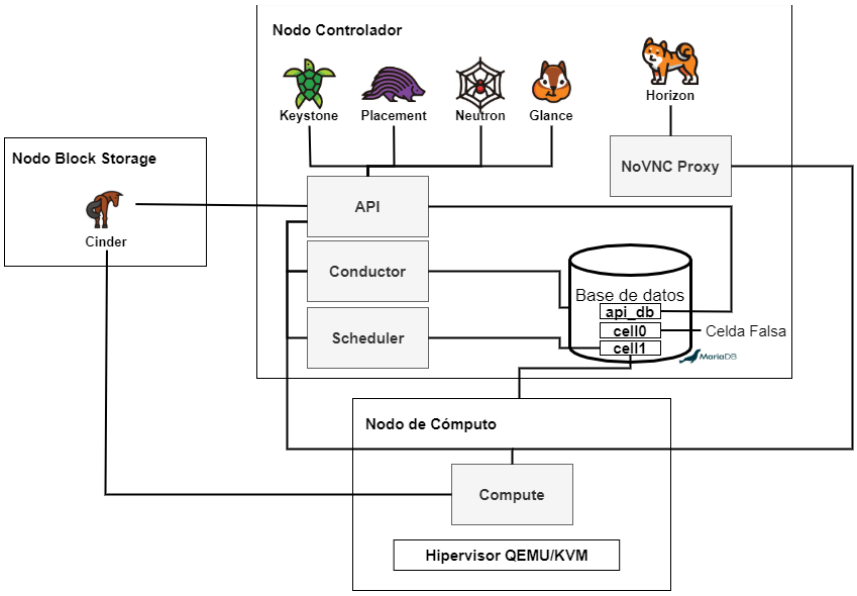


Ilustración 34 Diagrama de Nova en los nodos Controlador y Cómputo.

3.5.6. Cinder

El componente Cinder está dividido en 4 servicios: API, Scheduler, Volume, Backup. Los 2 primeros se ejecutan en el nodo Controlador y los otros dos en el nodo Block Storage. Además, el nodo Block Storage actúa como Target Server y permite a las máquinas virtuales ejecutadas en el nodo de Cómputo acceder a discos virtuales por medio del protocolo ISCSI. El servicio volume de Cinder utiliza como backend LVM v2. Existe un solo LVM Volume Group llamado cinder-volumes que utiliza el disco de almacenamiento de 1000 GB adicional del nodo Block Storage.

El servicio de API Cinder procesa las peticiones realizadas por el servicio Compute de Nova para reservar los discos virtuales. Por otro lado, el servicio de Backup se encarga de guardar una copia de seguridad de los discos y almacenarlos en el cluster de Swift,

la ilustración 35 muestra el diagrama de Cinder en los nodos controlador, cómputo y Block storage.

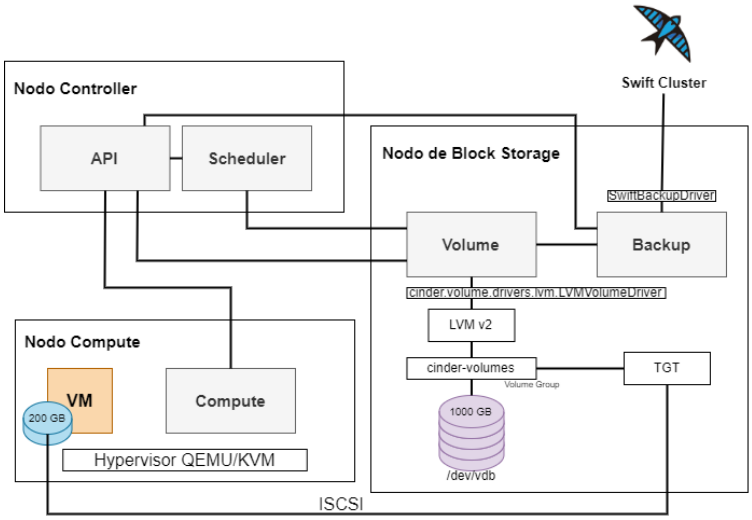


Ilustración 35 Diagrama de Cinder en los nodos Controlador, Cómputo y Block Storage

3.5.7. Horizon

El laboratorio Smartlab cuenta con una IP pública 190.96.111.123 y los siguientes puertos: 80, 443 y 64001 – 64010. El récord A del dominio smartlab-epn.com está vinculado a la IP mencionada anteriormente. Un cliente que desee acceder al servicio de nube de OpenStack desde Internet, puede hacerlo mediante la URL smartlab-epn.com/horizon. Además, se cuenta con un certificado SSL emitido por la Autoridad Certificadora Let's encrypt.

Las peticiones HTTPS del cliente son procesadas y enviadas por el router hacia el Proxy reverso implementado en NGINX. El proxy reverso se encarga de redirigir el tráfico HTTPS hacia el servidor Apache que se ejecuta en el Nodo Controlador y este hacia el componente Horizon. El tráfico para noVNC trabaja con el procedimiento, pero se origina con la petición del usuario hacia el puerto 64002, la petición es procesada por el Port Forwarding del router y transformado al 6080, la ilustración 36 muestra el flujo de tráfrico entre los servicios de Openstack y el cliente desde internet.

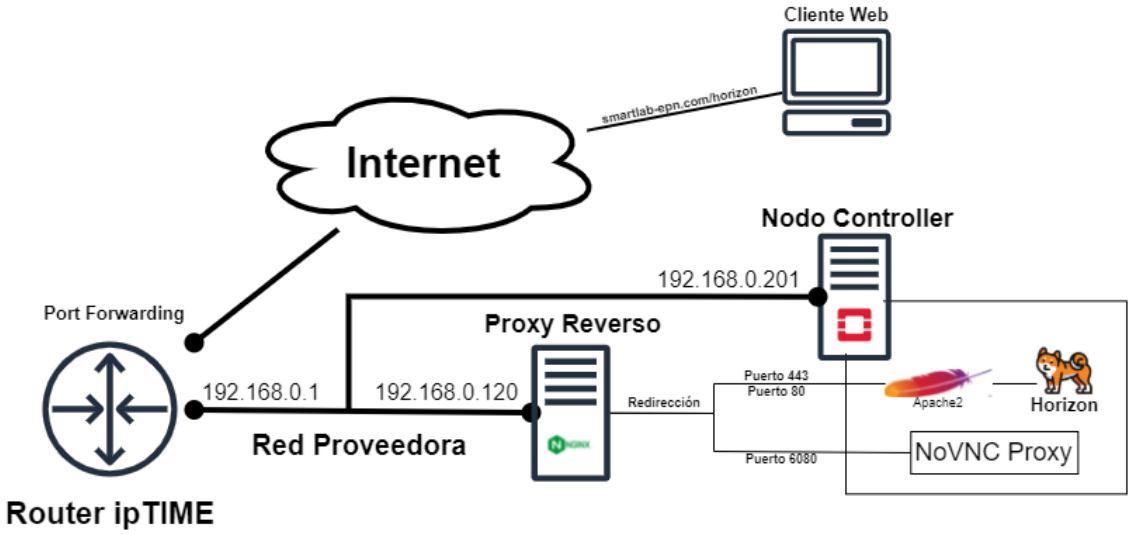


Ilustración 36 Flujo del tráfico entre los servicios de OpenStack y el cliente desde Internet.

3.6. Configuración general de todos los nodos

Instalar chrony para utilizar el protocolo de tiempo de red y mantenerlos sincronizados.

```
# apt install chrony
```

Comando 1 Instalación de Chrony

Agregar el repositorio de OpenStack versión Xena.

```
# add-apt-repository cloud-archive:xena
```

Comando 2 Registro del repositorio de OpenStack Xena

Registrar el nombre de host de cada nodo con su respectiva IP (red administración).

```
# sed -i '/127.0.0.1 localhost/a 192.168.122.6 object-2' /etc/hosts
# sed -i '/127.0.0.1 localhost/a 192.168.122.5 object-1' /etc/hosts
# sed -i '/127.0.0.1 localhost/a 192.168.122.4 storage-1' /etc/hosts
# sed -i '/127.0.0.1 localhost/a 192.168.122.3 compute' /etc/hosts
# sed -i '/127.0.0.1 localhost/a 192.168.122.2 controller' /etc/hosts
```

Comando 3 Registro de nombres e IP de cada nodo en el archivo /etc/hosts

3.7. Software adicional de la instalación

La ilustración 37 muestra la configuración general de todos los nodos en este proyecto.

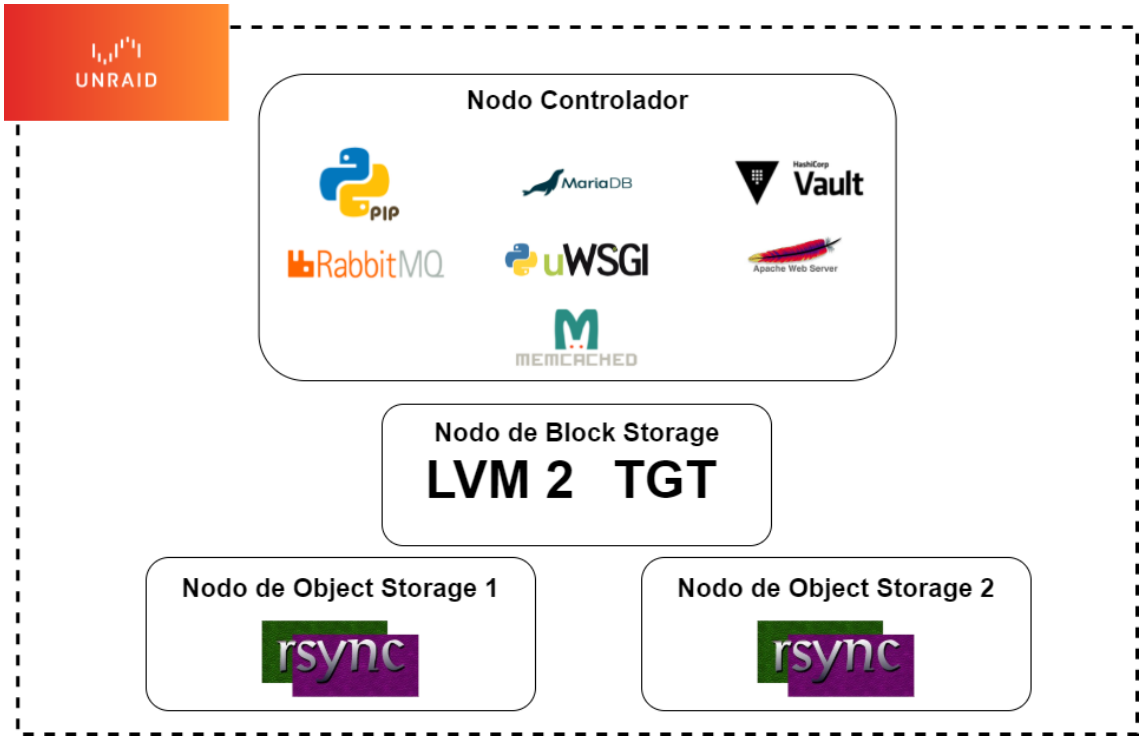


Ilustración 37 Configuración general de todos los nodos

3.7.1. MariaDB

OpenStack utiliza una base de datos tipo SQL para almacenar la información generada por los componentes. Actualmente los motores soportados son MySQL, MariaDB, PostgreSQL.

Instalar la base de datos relacional MariaDB y el cliente de MySQL (compatible con MariaDB) de Python 3 mediante el gestor de paquetes de Ubuntu.

```
# apt install mariadb-server python3-pymysql
```

Comando 4 Instalación de MariaDB y cliente MySQL

Crear el archivo de configuración **99-openstack.cnf.** En el archivo se especificará en **bind-address** para habilitar el acceso remoto a los demás nodos. Establecer la opción de para almacenar con el conjunto de caracteres UTF-8.

```
# nano /etc/mysql/mariadb.conf.d/99-openstack.cnf
```

Comando 5 Configuración de acceso para MariaDB

Reiniciar el servicio de la base de datos para cargarlos nuevos cambios establecidos en el archivo anterior.

```
# service mysql restart
```

Comando 6 Reinicio del servicio de MySQL

Ejecutar el script de **mysql_secure_installation** para mejorar la seguridad de la instalación de MariaDB. Las acciones realizadas son:

- Establecer una contraseña para el usuario root
- Remover el usuario anónimo
- Remover la base de datos de pruebas.

```
# mysql_secure_installation
```

Comando 7 Configuración inicial de MariaDB

3.7.2. RabbitMQ

OpenStack utiliza un encolador de mensajes para coordinar las operaciones y el status de información entre los servicios. El servicio será instalado en el nodo controller. OpenStack actualmente soporta RabbitMQ, Qpid y ZeroMQ, sin embargo, algunos componentes trabajan con un encolador de mensajes en específico.

Instalar el encolador de mensajes RabbitMQ mediante el gestor de paquetes de Ubuntu.

Agregar el usuario **openstack**

```
# rabbitmqctl add_user openstack r4BB1T-90d3
```

Comando 8 Creación del usuario openstack en RabbitMQ

Establecer los permisos de lectura y escritura para el usuario creado anteriormente.

```
# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

Comando 9 Establecimiento de permisos del usuario openstack

3.7.3. Memcached

El mecanismo de autenticación del servicio de Identidad utilizado por los servicios utiliza Memcached para almacenar los tokens en cache.

Instalar el servicio de cache memcached y la interfaz de Python 3 mediante el gestor de paquetes de Ubuntu.

```
# apt install memcached python3-memcache
```

Comando 10 Instalación de Memcached

Editar el archivo de configuración **memcached.conf.** En el archivo se especifica la dirección IP de la red de gestión utilizada para aceptar peticiones de los demás nodos

```
# nano /etc/memcached.conf
```

Comando 11 Edición del archivo de configuración de Memcached

Reiniciar el servicio de cache para cargar los nuevos cambios establecidos en el archivo anterior.

```
# service memcached restart
```

Comando 12 Reinicio del servicio Memcached

3.7.4. Vault

Todos los secretos (usuarios, contraseñas, llaves privadas) están almacenados en un baúl sellado por 5 llaves maestras. Solo se puede acceder al baúl de forma local es decir desde el nodo controller.

Crear el archivo de configuración vault.hcl. En el archivo se especifica el nombre de la base de datos y sus respectivas credenciales. La ruta del certificado SSL local con su llave privada, el certificado de la autoridad certificadora y la dirección IP local por donde procesará las peticiones.

```
# nano /etc/vault/vault.hcl
```

Comando 13 Creación del archivo de configuración de Vault

```
ui = true

storage "mysql" {
  username = "vault"
  password = "vault-sudk2nio"
  database = "vault"
}

listener "tcp" {
  address       = "0.0.0.0:8200"
  tls_cert_file = "/opt/vault/tls/cert/CA/localhost/localhost.crt"
  tls_key_file  = "/opt/vault/tls/cert/CA/localhost/localhost.decrypted.key"
  tls_client_ca_file = "/opt/vault/tls/cert/CA/CA.pem"
}
```

Configuración 1 Archivo de configuración de Vault

Las credenciales que se hayan utilizado en el prototipo de Cloud Computing están almacenadas en un baúl de secretos (vault). El baúl puede ser abierto solamente con la llave maestra, que está dividida en 5 llaves compartidas, que se puede reconstruir con mínimo 3 llaves de las 5 existentes. Este baúl de secretos utilizará la base de datos MariaDB como backend de almacenamiento y residirá dentro de la red de administración, por lo que no será accesible desde Internet. El tráfico será encriptado con un certificado SSL autogenerado y firmado por una autoridad de certificación propia (CA).

3.7.4.1. Generación de SSL

Generar una llave privada encriptada con Triple DES para el certificado raíz.

```
# openssl genrsa -out CA.key -des3 2048
```

Comando 14 Creación de llave para el certificado raíz.

Generar un certificado raíz utilizando la llave generada anteriormente. Tendrá una validez de aproximadamente 10 años.

```
# openssl req -x509 -sha256 -new -nodes -days 3650 -key CA.key -out CA.pem
```

Comando 15 Creación del certificado raíz

Establecer la información que será escrita en el certificado SSL firmado. El certificado funcionará para localhost y la dirección IP de loopback 127.0.0.1.

```
# nano localhost.ext
```

Comando 16 Registro de información del certificado SSL

```
authorityKeyIdentifier = keyid,issuer
basicConstraints = CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = localhost
IP.1 = 127.0.0.1
```

Configuración 2 Información del certificado SSL

Generar una llave privada encriptada con Triple DES para la solicitud de firma de certificado (CSR).

```
# openssl genrsa -out localhost.key -des3 2048
```

Comando 17 Creación de una llave privada

Generar la solicitud de firma de certificado.

```
# openssl req -new -key localhost.key -out localhost.csr
```

Comando 18 Creación de solicitud de firma de certificado

Utilizar la solicitud de firma de certificado (CSR) para solicitar a la autoridad certificadora (CA) que firme el certificado con la información establecida anteriormente.

```
# openssl x509 -req -in localhost.csr \
-CA CA.pem -CAkey CA.key -CAcreateserial -days 3650 \
-sha256 -extfile localhost.ext -out localhost.crt
```

Comando 19 Firma de certificado

Desencriptar la llave privada del certificado.

```
# openssl rsa -in localhost.key -out localhost.decrypted.key
```

Comando 20 Desencriptación de la llave privada

3.7.4.2. Configuración del backend de almacenamiento

Ingresar a MariaDB, crear la base de datos **vault** y establecer los permisos respectivos al usuario **vault**.

```
# mysql
MariaDB [(none)]> CREATE DATABASE vault;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON vault.* TO 'vault'@'localhost' \
  IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON vault.* TO 'vault'@'%' \
  IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> EXIT;
```

Comando 21 Creación de la base de datos vault

3.7.4.3. Instalación de Vault

Descargar la llave GPG del repositorio apt de Hashicorp.

```
# curl -fsSL https://apt.releases.hashicorp.com/gpg | apt-key add -
```

Comando 22 Descarga de la llave del repositorio de HashiCorp

Agregar el repositorio apt a la lista de repositorios.

```
# apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com
$(lsb_release -cs) main"
```

Comando 23 Registro del repositorio de HashiCorp

Actualizar la lista de repositorios e instalar vault por medio de mediante el gestor de paquetes de Ubuntu.

```
# apt-get update && apt-get install vault
```

Comando 24 Instalación de vault

Crear la unidad de archivo de configuración de servicio de systemd de Vault

```
# nano /lib/systemd/system/vault-server.service
```

Comando 25 Creación del servicio de Vault

Editar el archivo de configuración vault.hcl

```
# nano /etc/vault.d/vault.hcl
```

Comando 26 Edición del archivo de configuración de vault.hcl

Habilitar la interfaz web.

```
ui = true
```

Configuración 3 Configuración de interfaz web de Vault

Indicar las credenciales, la base de datos y el archivo de la autoridad certificadora para utilizar TLS en el backend de almacenamiento

```
storage "mysql" {
  username = "vault"
  password = "SECRET-PASSWORD"
  database = "vault"
  tls_ca_file = "/opt/vault/tls/cert/CA/CA.pem"
}
```

Configuración 4 Configuración del backed de almacenamiento de Valut

Especificar el puerto y la interfaz perteneciente a la red de administración para la conexión al baúl junto con el certificado SSL, su llave privada y el archivo de la autoridad certificadora respectivamente.

```
listener "tcp" {
  address       = "192.168.122.2:8200"
  tls_cert_file = "/opt/vault/tls/cert/CA/localhost/localhost.crt"
  tls_key_file  = "/opt/vault/tls/cert/CA/localhost/localhost.decrypted.key"
  tls_client_ca_file = "/opt/vault/tls/cert/CA/CA.pem"
}
```

Configuración 5 Configuración SSL de Vault

Reiniciar el servicio de vault para cargar la nueva configuración.

```
# server vault restart
```

Comando 27 Reinicio del servicio de Vault

Inicializar el baúl con 5 llaves compartida y un token root.

```
# vault operator init
```

Comando 28 Inicialización de Vault

Almacenar las llaves en un lugar secreto y seguro.

```
Unseal Key 1: SECRET-SHARED-KEY-1
Unseal Key 2: SECRET-SHARED-KEY-2
Unseal Key 3: SECRET-SHARED-KEY-3
Unseal Key 4: SECRET-SHARED-KEY-4
Unseal Key 5: SECRET-SHARED-KEY-5

Initial Root Token: SECRET-TOKEN
```

Configuración 6 Llaves y token del baúl de Vault

3.7.4.4. Abertura del baúl

Utilizando un navegador dirigirse a la dirección del servidor de llaves mediante una petición HTTPS. **https://192.168.122.2:8200.** Ingresar una de las llaves compartidas (SECRET-SHARED-KEY-1), la ilustración 38 evidencia el ingreso de una de las llaves compartidas.

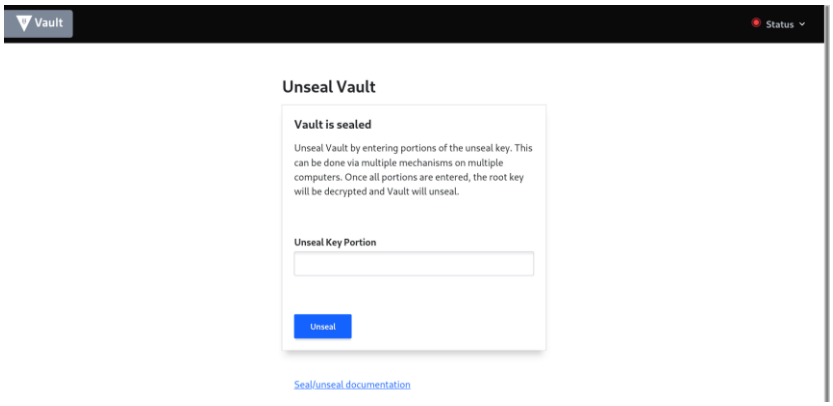


Ilustración 38 Ingreso de una de las llaves compartidas

Ingresar una de las llaves compartidas. La ilustración 39 igualmente muestra el ingreso de una de las llaves compartidas (SECRET-SHARED-KEY-2)

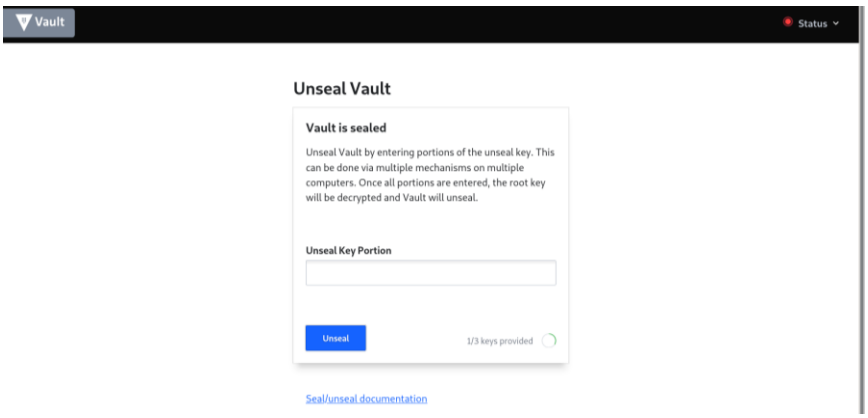


Ilustración 39 Ingreso de una de las llaves compartidas. (SECRET-SHARED-KEY-2)

Ingresar una de las llaves compartidas. La ilustración 40 muestra el ingreso de la llave compartida (SECRET-SHARED-KEY-3).

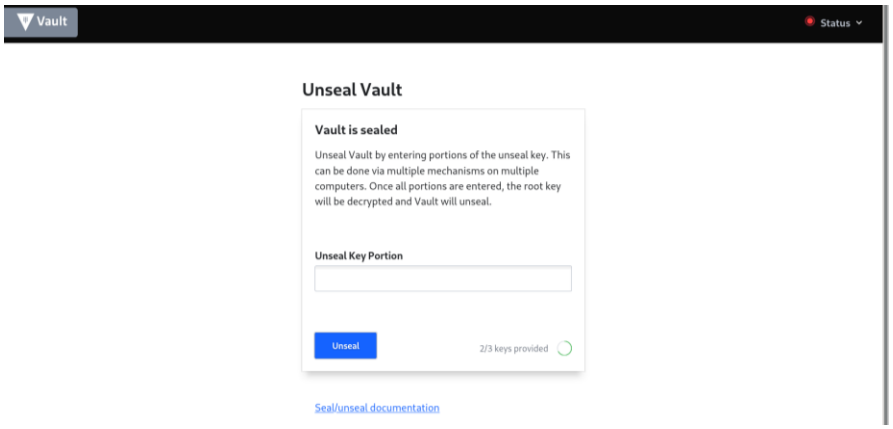


Ilustración 40 Ingreso una de las llaves compartidas. (SECRET-SHARED-KEY-3)

La ilustración 41 muestra el ingreso del root token para autenticarse en vault.

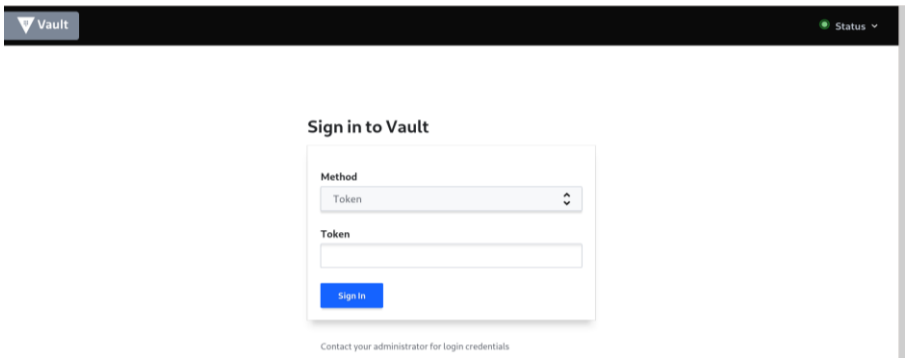


Ilustración 41 Ingreso del root token para autenticarse en vault.

La ilustración 42 muestra la visualización del status del baúl.

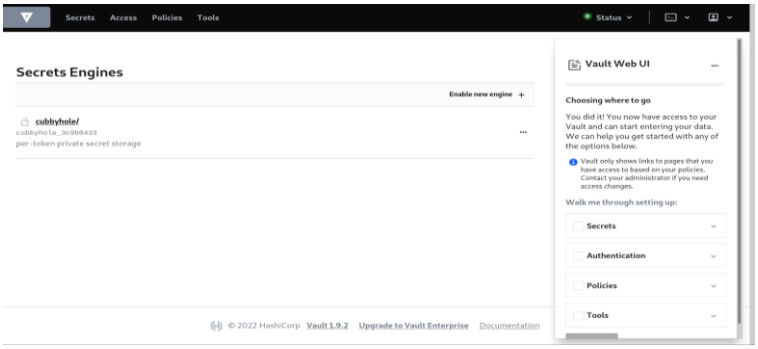


Ilustración 42 Visualización del status del baul.

La ilustración 43 indica la habilitación del backend genérico KV v2.

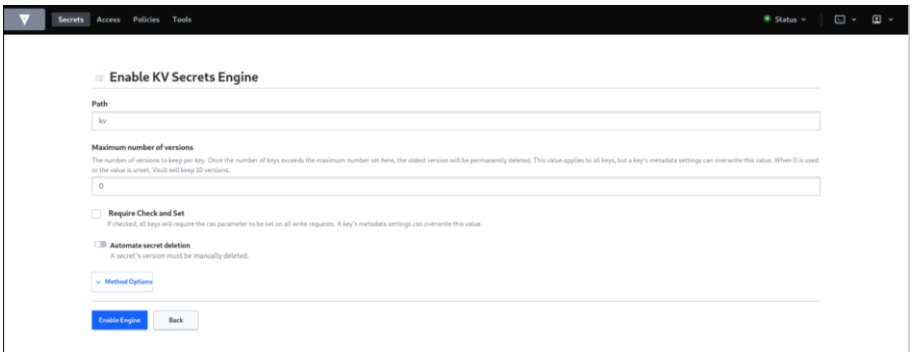


Ilustración 43 Habilitación del backend genérico KV v2

Credenciales de todo el proyecto almacenado como secretos, la ilustración 44 muestra el almacenamiento de credenciales del proyecto como secretos.

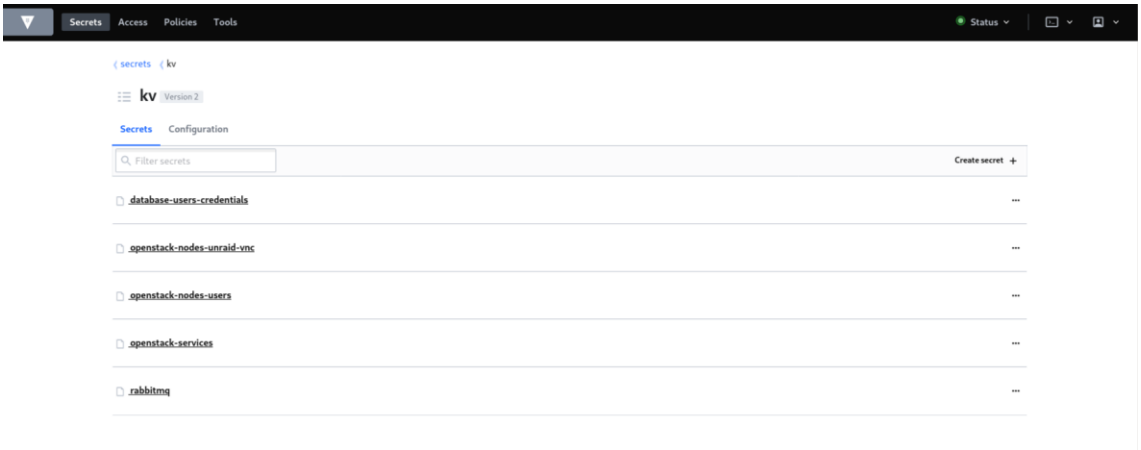


Ilustración 44 Almacenamiento de credenciales del proyecto como secretos.

3.8. Instalación de OpenStack

3.8.1. Nodo Controller

3.8.1.1. OpenStack Client

Interactúa directamente con el componente Keystone para ejecutar comandos en todos los componentes. El método de autenticación utilizado es usuario y contraseña. El cliente utiliza los parámetros como variables de Shell Bash y son almacenados en un archivo de texto no encriptado como script con permisos de solo lectura del usuario **root.**

```
# export OS_USERNAME=admin
# export OS_PASSWORD=SECRET-PASSWORD
# export OS_PROJECT_NAME=admin
# export OS_USER_DOMAIN_NAME=admin
# export OS_PROJECT_DOMAIN_NAME=admin
# export OS_OS_AUTH_URL=http://controller:5000/v3
# export OS_IDENTITY_API_VERSION=3
```

Comando 29 Comandos de Establecimiento de parámetros de autenticación.

Instalar la versión Python 3 del cliente de OpenStack mediante el gestor de paquetes de Ubuntu.

```
# apt install python3-openstackclient
```

Comando 30 Instalación de OpenStack Client

3.8.1.2. Keystone

A continuación, se enlistan los pasos para realizar la instalación y configuración del componente:

Instalar el servicio de Identidad Keystone, el servidor HTTP apache2 con el módulo wsgi mediante el gestor de paquetes de Ubuntu.

```
# apt install keystone apache2 libapache2-mod-wsgi
```

Comando 31 Instalación de Keystone, apache y el módulo wsgi.

Ingresar a MariaDB, crear la base de datos **keystone** y establecer los permisos respectivos al nuevo usuario creado **keystone**.

```
# mysql
MariaDB [(none)]> CREATE DATABASE keystone;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> EXIT;
```

Comando 32 Creación de la base de datos de Keystone

```
# nano /etc/keystone/keystone.conf
```

Comando 33 Edición del archivo de configuración de Keystone

Establecer los parámetros de conexión a la base de datos

```
[database]
connection = mysql+pymysql://keystone:SECRET-PASSWORD@controller/keystone
```

Configuración 7 Parámetro de conexión a la base de datos

Establecer el uso de tokens tipo fernet.

```
[token]
provider = fernet
```

Configuración 8 Configuración de uso de Tokens fernet

Inicializar la base de datos **keystone**.

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

Comando 34 Inicialización de la base de datos Keystone

Inicializar el repositorio de tokens fernet.

```
# keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
# keystone-manage credential_setup --keystone-user keystone --keystone-group keystone
```

Comando 35 Inicalización del repositorio de Tokens fernet

Inicializar el servicio de Identidad, creando la región **RegionOne,** y los endpoints de administración, interno y público respectivamente.

```
# keystone-manage bootstrap --bootstrap-password SECRET-PASSWORD \
  --bootstrap-admin-url http://controller:5000/v3/ \
  --bootstrap-internal-url http://controller:5000/v3/ \
  --bootstrap-public-url http://controller:5000/v3/ \
  --bootstrap-region-id RegionOne
```

Comando 36 Inicialización de Keystone

Editar el archivo de configuración de Apache

```
# nano /etc/apache2/apache2.conf
```

Comando 37 Edición del archivo de configuración de Apache

Establecer el nombre de host controller

```
ServerName controller
```

Comando 38 Definición del nombre del host controller

Reiniciar el servidor HTTP para cargar los nuevos cambios establecidos en el archivo anterior.

```
# service apache2 restart
```

Comando 39 Reinicio del servidor Apache

Crear el proyecto service. En el proyecto service se almacena los usuarios con rol de administrador de cada componente.

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

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Service Project                  |
| domain_id   | default                          |
| enabled     | True                             |
| id          | 9a48ae1f45af47f9b79dc4a322512716 |
| is_domain   | False                            |
| name        | service                          |
| options     | {}                               |
| parent_id   | default                          |
| tags        | []                               |
+-------------+----------------------------------+
```

Comando 40 Creación del proyecto service en OpenStack

3.8.1.3. Swift

Instalar las utilidades de Swift, el proxy por donde se interactuará con el clúster y sus dependencias.

```
# apt-get install swift swift-proxy python-swiftclient \
  python-keystoneclient python-keystonemiddleware
```

Comando 41 Instalación de Swift

```
# nano /etc/swift/swift.conf
```

Comando 42 Edición del archivo de configuración de Swift

Establecer el policy predeterminado con índice 0. Este policy permite retrocompatibilidad con contenedores legacy.

```
[storage-policy:0]
name = Policy-0
default = yes
```

Configuración 9 Configuración del Policy predeterminado de Swift

Establecer el prefijo y sufijo del hash de la ruta. Estos valores son secretos y determinan el lugar en donde se almacenarán los datos en los nodos del clúster de almacenamiento de objetos. Además, son fijos, por lo que una vez inicializado el servicio no pueden ser cambiados.

```
[swift-hash]
swift_hash_path_suffix = SECRET_STRING
swift_hash_path_prefix = SECRET_STRING
```

Configuración 10 Configuración del prefijo y sufijo del Hash de ruta de Swift

```
# nano /etc/swift/proxy-server.conf
```

Comando 43 Edición del archivo de configuración del Proxy de Swift

Establecer el puerto habilitado para interactuar con el clúster, el usuario de servicio y el directorio donde se encuentra de Swift.

```
[DEFAULT]
bind_port = 8080
user = swift
swift_dir = /etc/swift
```

Configuración 11 Configuración del puerto, usuario y directorio de Swift

Lista predeterminada de módulos necesarios para correcta operación del clúster, cabe destacar que el orden de los módulos importa. Los módulos adicionales agregados son:

- Módulo authtoken: Encargado de validar el token de autenticación y recuperar la información del usuario correspondiente.
- Módulo keystoneauth: Middleware encargado de comunicarse con Keystone.

```
[pipeline:main]
pipeline = catch_errors gatekeeper healthcheck proxy-logging cache container_sync
bulk ratelimit authtoken keystoneauth container-quotas account-quotas slo dlo
versioned_writes proxy-logging proxy-server
```

Configuración 12 Configuración de los módulos de Swift

Permitir que las cuentas que todavía no existen en el clúster sean creadas automáticamente.

```
[app:proxy-server]
account_autocreate = True
```

Configuración 13 Configuración de autocreación de cuentas en Swift

Establecer los roles reconocidos por el servicio de almacenamiento de objetos.

- Rol admin: Tiene los permisos de administrador y todos los permisos.
- Rol user: Rol asginado a los usuarios comunes, pueden poder crear sus propios contenedores y objectos y también eliminarlos.

```
[filter:keystoneauth]
operator_roles = admin, user
```

Configuración 14 Configuración de los roles en Swift

Establecer los datos del token y parámetros para comunicarse con el servicio de Identidad.

```
[filter:authtoken]
paste.filter_factory = keystonemiddleware.auth_token:filter_factory
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_id = default
user_domain_id = default
project_name = service
username = swift
password = SECRET-PASSWORD
delay_auth_decision = True
service_token_roles_required = True
service_token_roles: service
```

Configuración 15 Configuración de credenciales de Swift

Establecer la dirección y Puerto del servidor de cache.

```
[filter:cache]
memcache_servers = controller:11211
```

Configuración 16 Configuración de cache de Swift

Crear el usuario de servicio **swift**, asignar el rol de administrador e incluírlo en el proyecto **service**.

```
# openstack user create --domain default --password-prompt swift
# openstack role add --project service --user swift admin
```

Field	Value
domain_id	default
enabled	True
id	374b8db58a9446c1ae064d59948a4626
name	swift
options	{}
password_expires_at	None

Comando 44 Creación del usuario de servicio Swift en OpenStack

Crear el servicio object-store con el nombre de swift

```
# openstack service create --name swift \
  --description "OpenStack Object Storage" object-store
```

Field	Value
description	OpenStack Object Storage
enabled	True
id	893ae1b0cbee44c18162a36eae219243
name	swift
type	object-store

Comando 45 Creación del servicio object-store en OpenStack

Crear los endpoints de administración, interno y público respectivamente del componente en la región **RegionOne.**

```
# openstack endpoint create --region RegionOne \
  object-store public http://controller:8080/v1/AUTH_%\(project_id\)s
```

Field	Value
enabled	True
id	6bd5f9f2369f406db4d5745919711da9
interface	public
region	RegionOne
region_id	RegionOne
service_id	893ae1b0cbee44c18162a36eae219243
service_name	swift
service_type	object-store
url	http://controller:8080/v1/AUTH_%(project_id)s

Comando 46 Creación del Endpoint del servicio object-store público

```
# openstack endpoint create --region RegionOne \
  object-store internal http://controller:8080/v1/AUTH_%\(project_id\)s
```

Field	Value
enabled	True
id	61635640fc084686868ebf8ef9d71b15
interface	internal
region	RegionOne
region_id	RegionOne
service_id	893ae1b0cbee44c18162a36eae219243
service_name	swift
service_type	object-store
url	http://controller:8080/v1/AUTH_%(project_id)s

Comando 47 Creación del Endpoint del servicio object-store interno

```
# openstack endpoint create --region RegionOne \
  object-store admin http://controller:8080/v1
```

Field	Value
enabled	True
id	d45ba5ea7b0744318cab44fc4713ee57
interface	admin
region	RegionOne
region_id	RegionOne
service_id	81499488267d424a8e185f79fcf1cee1
service_name	swift
service_type	object-store
url	http://controller:8080/v1

Comando 48 Creación del Endpoint del servicio object-store admin

3.8.1.3.1. *Creación de los rings iniciales*

Se debe determinar el número de particiones que residirán en el anillo (ring). Mientras menos sea el número de particiones, menos trabajo deberá realizar el clúster de almacenamiento de objetos. Luego se agrega cada nodo de almacenamiento al anillo indicando, la zona, la dirección IP del nodo del clúster, el puerto habilitado para cuenta y los discos designados. Por último, se rebalancea cada anillo.

Los anillos se crean con el comando swift-ring-builder. Se debe especificar el número de particiones máximas, número de réplicas de cada objeto y el tiempo mínimo para mover una partición. En el almacenamiento de objetos, una partición indica un directorio.

Cambiarse al directorio de configuración de Swift

```
# cd /etc/swift
```

Comando 49 Traslado al directorio de configuración de Swift

3.8.1.3.1.1. Anillo de cuenta

El anillo de cuenta mantiene la lista de contenedores.

```
# swift-ring-builder account.builder create 10 3 12
# swift-ring-builder account.builder add \
  --region 1 --zone 1 --ip 192.168.122.5 --port 6202 --device vdb --weight 115
# swift-ring-builder account.builder add \
  --region 1 --zone 1 --ip 192.168.122.5 --port 6202 --device vdc --weight 115
# swift-ring-builder account.builder add \
  --region 1 --zone 2 --ip 192.168.122.6 --port 6202 --device vdb --weight 115
# swift-ring-builder account.builder add \
  --region 1 --zone 2 --ip 192.168.122.6 --port 6202 --device vdc --weight 115

# swift-ring-builder account.builder
# swift-ring-builder account.builder rebalance
```

Comando 50 Creación del anillo de cuenta de Swift

3.8.1.3.1.2. Anillo de contenedor

El anillo de contenedor mantiene la lista de objetos. Sin embargo, no rastrea la localización del objecto.

```
# swift-ring-builder container.builder create 10 3 12
# swift-ring-builder container.builder add \
  --region 1 --zone 1 --ip 192.168.122.5 --port 6201 --device vdb --weight 115
# swift-ring-builder container.builder add \
  --region 1 --zone 1 --ip 192.168.122.5 --port 6201 --device vdc --weight 115
# swift-ring-builder container.builder add \
  --region 1 --zone 2 --ip 192.168.122.6 --port 6201 --device vdb --weight 115
# swift-ring-builder container.builder add \
  --region 1 --zone 2 --ip 192.168.122.6 --port 6201 --device vdc --weight 115

# swift-ring-builder container.builder
# swift-ring-builder container.builder rebalance
```

Comando 51 Creación del anillo de contenedor de Swift

3.8.1.3.1.3. Anillo de objeto

El anillo de objecto mantiene la lista de la localización de objetos.

```
# swift-ring-builder object.builder create 10 3 12
# swift-ring-builder object.builder add \
  --region 1 --zone 1 --ip 192.168.122.5 --port 6200 --device vdb --weight 115
# swift-ring-builder object.builder add \
  --region 1 --zone 1 --ip 192.168.122.5 --port 6200 --device vdc --weight 115
# swift-ring-builder object.builder add \
  --region 1 --zone 2 --ip 192.168.122.6 --port 6200 --device vdb --weight 115
# swift-ring-builder object.builder add \
  --region 1 --zone 2 --ip 192.168.122.6 --port 6200 --device vdc --weight 115

# swift-ring-builder object.builder
# swift-ring-builder object.builder rebalance
```

Comando 52 Creación del anillo de objeto de Swift

Distribuir los archivos de configuración de anillo al nodo del clúster de almacenamiento de objectos **object1**.

```
# scp account.ring.gz object1@192.168.122.5:/home/object1/
# scp container.ring.gz object1@192.168.122.5:/home/object1/
# scp object.ring.gz object1@192.168.122.5:/home/object1/
# scp swift.conf object1@192.168.122.5:/home/object1/
```

Comando 53 Envío de archivos de configuración al nodo Object1

Distribuir los archivos de configuración de anillo al nodo del clúster de almacenamiento de objectos **object2**.

```
# scp account.ring.gz object2@192.168.122.6:/home/object2
# scp container.ring.gz object2@192.168.122.6:/home/object2
# scp object.ring.gz object2@192.168.122.6:/home/object2
# scp swift.conf object2@192.168.122.6:/home/object2
```

Comando 54 Envío de archivos de configuración al nodo Object2

Establecer la propiedad de los archivos del directorio de configuración a root y el grupo de swift, con el fin de proteger los datos de configuración confidenciales.

```
# chown -R root:swift /etc/swift
```

Comando 55 Cambio de propiedad a root y grupo swift de todos los archivos del directorio de configuración Swift

Reiniciar el servicio de cache memcached y el servicio de proxy del clúster de almacenamiento de objetos.

```
# service memcached restart
# service swift-proxy restart
```

Comando 56 Reinicio del servicio swift-proxy

3.8.1.4. Glance

Ingresar a MariaDB, crear la base de datos **glance** y establecer los permisos respectivos al nuevo usuario creado **glance**.

```
# mysql
MariaDB [(none)]> CREATE DATABASE glance;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
  IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
  IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> EXIT;
```

Comando 57 Creación de la base de datos de Glance

Crear el usuario de servicio **glance**, asignar el rol de administrador e incluirlo en el proyecto **service**.

```
# openstack user create --domain default --password-prompt glance
# openstack role add --project service --user glance admin
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | c011c0be70dc48d59be9b502b9bb91f8 |
| name                | glance                           |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Comando 58 Creación del usuario de servicio de Swift en OpenStack

Crear el servicio de imagen con el nombre de glance

```
# openstack service create --name glance \
  --description "OpenStack Image" image
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Image                  |
| enabled     | True                             |
| id          | 8191ee4c076642dca92510f13b3570a2 |
| name        | glance                           |
| type        | image                            |
+-------------+----------------------------------+
```

Comando 59 Creación del servicio image en OpenStack

Crear los endpoints de administración, interno y público respectivamente del componente en la región RegionOne.

```
# openstack endpoint create --region RegionOne \
  image public http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | e77100b012e240618a82bd2d7d7fa89b |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8191ee4c076642dca92510f13b3570a2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```

Comando 60 Creación del Endpoint del servicio image público

```
# openstack endpoint create --region RegionOne \
  image internal http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 631649a35ee44fd8906793af45e322c3 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8191ee4c076642dca92510f13b3570a2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```

Comando 61 Creación del Endpoint del servicio image interno

```
# openstack endpoint create --region RegionOne \
  image admin http://controller:9292
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 82dbe4c8eb9d48da8b493730bd513ab1 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 8191ee4c076642dca92510f13b3570a2 |
| service_name | glance                           |
| service_type | image                            |
| url          | http://controller:9292           |
+--------------+----------------------------------+
```

Comando 62 Creación del Endpoint del servicio image interno

Instalar el servicio de imagen glance y su dependencia para operar con el servicio de AWS S3 mediante el gestor de paquetes de Ubuntu.

```
# apt install glance python3-boto3
```

Comando 63 Instalación de dependencia de software de Glance

Establecer el clúster de almacenamiento de objetos como backend de almacenamiento, la configuración de la conexión de la base de datos. L

```
# nano /etc/glance/glance-api.conf
```

Comando 64 Edición del archivo de configuración del api de Glance.

Establecer la conexión a RabbitMQ y especificar el backend de almacenamiento de objetos.

```
[default]

enabled_backends= cheap:swift
transport_url=rabbit://openstack:SECRET-PASSWORD@controller
```

Configuración 17 Configuración de RabbitMQ en Glance

Establecer la conexión a la base de datos glance con las credenciales del usuario glance.

```
[database]
connection = mysql+pymysql://glance:SECRET-PASSWORD@controller/glance
```

Configuración 18 Configuración de la base de datos en Glance

Establecer los datos del token y parámetros para comunicarse con el servicio de Identidad.

```
[keystone_authtoken]
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = SECRET-PASSWORD
service_token_roles_required = True
```

Configuración 19 Configuración de credenciales de Glance

Especificar el backend predeterminado. Para este caso, el único backend habilitado es el de almacenamiento de objetos.

```
[glance_store]
default_backend=cheap
```

Configuración 20 Configuración del Backend de Glance

Establecer la configuración de almacenamiento de objetos. Se describe los parámetros a continuación respectivamente:

- La comunicación será a través del Endpoint público
- El tipo de almacenamiento es almacenamiento de objetos
- No verifica el certificado SSL del servidor. La comunicación entre el servicio de Identidad y el clúster de almacenamiento de objetos no está encriptada con el fin de mejorar rendimiento.

Número de veces que intentará descargar una imagen del clúster

```
[glance.store.swift.store]
swift_store_endpoint_type = publicURL
swift_store_service_type = object-store
swift_store_auth_insecure = True
swift_store_retry_get_count = 2
```

Configuración 21 Configuración del almacenamiento de imagen

Establecer los parámetros del backend de almacenamiento de objetos. Se describe los parámetros a continuación respectivamente:

- Descripción del backend de almacenamiento.
- Ruta al archivo de datos del token para comunicarse con el servicio de Identidad.
- Referencia a la sección del archivo anterior. Glance puede trabajar con más de un clúster de almacenamiento, en consecuencia, se necesita especificar que credenciales utilizar.
- Crea un contenedor si todavía no existe, cuando se carga una imagen al clúster.
- Define que a un archivo como grande cuando sobrepasa los 100 megabytes. Cuando un archivo es considerado como grande necesita ser segmentado.
- Divide a un objeto grande en porciones de 75 megabytes. El valor establecido es bajo porque garantiza que la transmisión no termine dando un resultado de conexion timeout.
- Define el nombre del contenedor

```
[cheap]
store_description = "Unraid Swift Storage"
swift_store_config_file = /etc/glance/glance-swift.conf
default_swift_reference = store1
swift_store_create_container_on_put = True
swift_store_large_object_size = 100
swift_store_large_object_chunk_size = 75
swift_store_container = glance
```

Configuración 22 Configuración del backend de almacenamiento de imagen

```
# nano /etc/swfit/glance-swift.conf
```

Comando 65 Edición del archivo de backend de almacenamiento de imagen

Se establecen de datos del token para comunicarse con el servicio de Identidad.

```
[store1]
user = service:swift
key = SECRET-PASSWORD
user_domain_id = default
project_domain_id = default
auth_version = 3
auth_address = http://controller:5000/v3
```

Configuración 23 Configuración de credenciales del backend de almacenamiento de imagen.

A continuación, se enlistan los pasos para realizar la instalación y configuración del componente:

Ingresar a MariaDB, crear la base de datos **neutron** y establecer los permisos respectivos al nuevo usuario creado **neutron**.

```
# mysql
MariaDB [(none)]> CREATE DATABASE neutron;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
  IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
  IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> EXIT;
```

Comando 66 Creación de la base de datos de Neutron

Crear el usuario de servicio **neutron**, asignar el rol de administrador e incluirlo en el proyecto **service**.

```
# openstack user create --domain default --password-prompt neutron
# openstack role add --project service --user neutron admin
```

```
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | e524eecf906041a4a6e45e2f804227fa |
| name                | neutron                          |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Comando 67 Creación del usuario de servicio de Neutron en OpenStack

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

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Networking             |
| enabled     | True                             |
| id          | 07a336cd8736443bbdce0824ed27f644 |
| name        | neutron                          |
| type        | network                          |
+-------------+----------------------------------+
```

Comando 68 Creación del servicio neutron en OpenStack

Crear los endpoints de administración, interno y público respectivamente del componente en la región **RegionOne.**

```
# openstack endpoint create --region RegionOne \
  network public http://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 27d35df24a964514a1f1dcd94261f305 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 07a336cd8736443bbdce0824ed27f644 |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

Comando 69 Creación del Endpoint del servicio network público

```
# openstack endpoint create --region RegionOne \
  network internal http://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 92e0eaf18e0f4725b014a8a32552dfbc |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 07a336cd8736443bbdce0824ed27f644 |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

Comando 70 Creación del Endpoint del servicio network interno

```
# openstack endpoint create --region RegionOne \
  network admin http://controller:9696
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 4e8da997cb414ebdbb8a9ddd6a9b7256 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 07a336cd8736443bbdce0824ed27f644 |
| service_name | neutron                          |
| service_type | network                          |
| url          | http://controller:9696           |
+--------------+----------------------------------+
```

Comando 71 Creación del Endpoint del servicio network admin

Instalar el servicio de red, sus respectivos agentes y la dependencia de Firewall como Servicio mediante el gestor de paquetes de Ubuntu. La dependencia de Firewall no se utilizará en el proyecto, pero desde la versión Xena es necesario incluirlo en la implementación para su correcta ejecución.

```
# apt install neutron-server neutron-plugin-ml2 neutron-linuxbridge-agent \
  neutron-l3-agent neutron-dhcp-agent neutron-metadata-agent \
  python3-neutron-fwaas
```

Comando 72 Instalación de dependencias de Software de Neutron

El prototipo solamente utilizará IPv4 por lo que se debe deshabilitar el servicio de anuncio de routers para IPv6.

```
# systemctl disable radvd
```

Comando 73 Desactivación del servicio radvd

Editar el archivo de configuración del agente de metadatos

```
# nano /etc/neutron/metadata_agent.ini
```

Comando 74 Edición del archivo de configuración del Plugin de metadatos de Neutron

Establecer el nombre del host que ejecutará el servicio de metadatos para las instancias y el string secreto para el proxy de metadatos.

```
[DEFAULT]
nova_metadata_host = controller
metadata_proxy_shared_secret = SECRET_STRING
```

Configuración 24 Configuración de host y secreto compartido del Plugin de metadatos de Neutron

```
# nano /etc/neutron/neutron.conf
```

Comando 75 Edición del archivo de configuración de Neutron

Se describe los parámetros de la sección DEFAULT a continuación respectivamente:

- Conexión a RabbitMQ.
- Habilita el plugin Modular Layer 2 que permite utilizar de manera simultánea una variedad de tecnologías de capa 2 como openvswitch, linuxbridge y agentes L2 de Hyper-V.
- Habilita el servicio de routing.
- Permite tener IPs repetidas (superposicionadas) en 2 redes iguales.
- Establece la autenticación mediante el servicio de Identidad Keystone.
- Envía notificaciones al servicio de cómputo (nova) el cambio de estatus de un puerto.

- Notifica al servicio de cómputo (nova) el cambio de datos de un puerto así es posible actualizar su propio cache.

```
[DEFAULT]
transport_url = rabbit://openstack:SECRET-PASSWORD@controller
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = true
auth_strategy = keystone
notify_nova_on_port_status_changes = true
notify_nova_on_port_data_changes = true
```

Configuración 25 Configuración general de Neutron

Establecer los datos del token y parámetros para comunicarse con el servicio de Identidad.

```
[keystone_authtoken]
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = SECRET-PASSWORD
```

Configuración 26 Configuración de credenciales de Neutron

Establecer la conexión a la base de datos neutron con las credenciales del usuario neutron.

```
[database]
connection = mysql+pymysql://neutron:SECRET-PASSWORD@controller/neutron
```

Configuración 27 Configuración de la base de datos de Neutron

Establecer los parámetros de autenticación del servicio de cómputo (nova) a Keystone

```
[nova]
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = SECRET-PASSWORD
```

Configuración 28 Configuración de credenciales de Nova en Neutron

Indicar el directorio para la ejecución de la librería de Oslo_concurrency.

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

Configuración 29 Configuración de Oslo en Neutron

```
# nano /etc/neutron/plugins/ml2/ml2_conf.ini
```

Comando 76 Edición del archivo de configuración del Plugin ML2 de Neutron

Se describe los parámetros del Plugin Modular Layer 2 a continuación respectivamente:

- Habilitar los drivers para redes FLAT, VLAN y VXLAN
- Establecer las redes tipo VXLAN para los proyectos donde se ejecutarán las instancias.
- Especificar los mecanismos de driver Linux Bridge y L2 population. El segundo es utilizado para escalar en grandes redes sobrepuestas (overlay).
- Habilitar el driver de extensión Port Security, para permitir que una instancia pueda enrutar tráfico en el caso de que sea necesario.

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

Configuración 30 Configuración general de ML2 en Neutron

Establecer la red provider como tipo flat

```
[ml2_type_flat]
flat_networks = provider
```

Configuración 31 Configuración de red de ML2 en Neutron

Especificar un máximo de 1000 redes en todo el prototipo de cloud computing.

```
[ml2_type_vxlan]
vni_ranges = 1:1000
```

Configuración 32 Configuración de rangos de VXLAN de ML2 en Neutron

Habilitar la opción de IPset para mejorar el rendimiento de los grupos de seguridad de red.

```
[securitygroup]
enable_ipset = true
```

Configuración 33 Configuración de grupos de seguridad en Neutron

```
# nano /etc/neutron/plugins/ml2/linuxbridge_agent.ini
```

Comando 77 Edición del archivo de configuración del agente Linuxbridge de Neutron

Especificar la interfaz que pertenece a la red proveedora física. La red proveedora virtual será mapeada en la red proveedora física.

```
[linux_bridge]
physical_interface_mappings = provider:enp1s0
```

Configuración 34 Configuración de la interfaz de la red Proveedora en Neutron

En la sección de VXLAN se especifica los siguientes parámetros respectivamente:

- Habilitar las redes sobrepuestas VXLAN (overlay)
- Especificar la dirección IP de la interfaz perteneciente a la red proveedora física.
- Habilitar el driver L2 Population.

```
[vxlan]
enable_vxlan = true
local_ip = 192.168.0.201
l2_population = true
```

Configuración 35 Configuración VXLAN en Neutron

Habilitar los grupos de seguridad y especificar el uso driver de firewall de Iptables

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

Configuración 36 Configuración el uso de IpTables en Neutron

Cargar el módulo de Kernel **br_netfilter** para habilitar el soporte de red bridging y facilitar la comunicación entre VxLAN.

```
# modprobe br_netfilter
# echo "net.bridge.bridge-nf-call-iptables = 1" >> /etc/sysctl.conf
# echo "net.bridge.bridge-nf-call-ip6tables = 1" >> /etc/sysctl.conf
# sysctl -p /etc/sysctl.conf
```

Comando 78 Activación del módulo br_netfilter

```
# nano /etc/neutron/l3_agent.ini
```

Comando 79 Edición del archivo de configuración del agente L3 de Neutron

Especificar el driver Linuxbridge para permitir al agente de capa 3 (L3) a proveer enrutamiento y servicio NAT a las redes virtuales self-service de cada proyecto.

```
[DEFAULT]
interface_driver = linuxbridge
```

Configuración 37 Registro del driver linuxbridge en el Agente L3 de Neutron

```
# nano /etc/neutron/dhcp_agent.ini
```

Comando 80 Edición del archivo de configuración del agente DCHP de Neutron

Establecer los siguientes parámetros respectivamente:

- Habilitar Linuxbridge como driver de interfaz
- Habilitar el driver DHCP de Dnsmasq
- Habilitar la opción de metadatos aislados para permitir a las instancias que no residen en las redes self-service (red proveedora) el acceso a los metadatos

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

Comando 81 Configuración del agente DHCP de Neutron

3.8.1.5. Nova Parte 1

Ingresar a MariaDB, crear las bases de datos **nova_api, nova y nova_cell0** y establecer los permisos respectivos al nuevo usuario creado **nova**.

```
# mysql
MariaDB [(none)]> CREATE DATABASE nova_api;
MariaDB [(none)]> CREATE DATABASE nova;
MariaDB [(none)]> CREATE DATABASE nova_cell0;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
  IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
  IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%' \
  IDENTIFIED BY 'SECRET-PASSWORD';
EXIT;
```

Comando 82 Creación de las bases de datos de Nova

Crear el usuario de servicio **nova**, asignar el rol de administrador e incluirlo en el proyecto **service**.

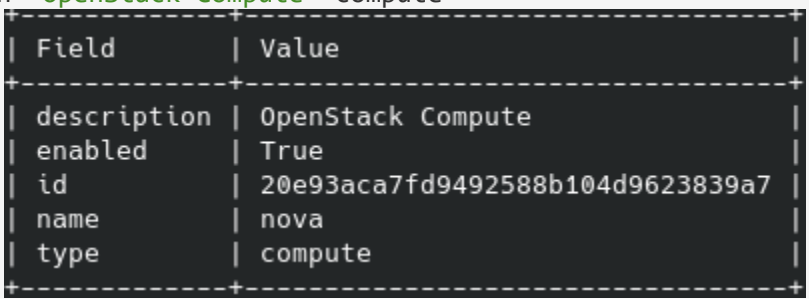
```
# openstack user create --domain default --password-prompt nova
# openstack role add --project service --user nova admin
```

Field	Value
domain_id	default
enabled	True
id	35593a65ac0d4527845ada5740bc1b5f
name	nova
options	{}
password_expires_at	None

Comando 83 Creación del usuario de servicio Nova en OpenStack

Crear el servicio de cómputo con el nombre de nova

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

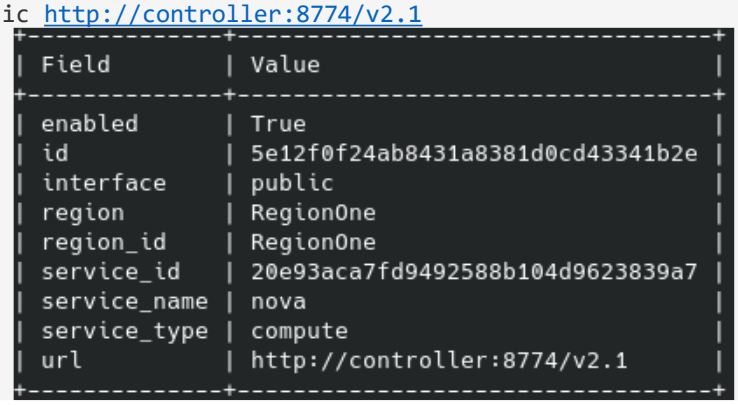


```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Compute                |
| enabled     | True                             |
| id          | 20e93aca7fd9492588b104d9623839a7 |
| name        | nova                             |
| type        | compute                          |
+-------------+----------------------------------+
```

Comando 84 Creación del servicio compute en OpenStack

Crear los endpoints de administración, interno y público respectivamente del componente en la región **RegionOne.**

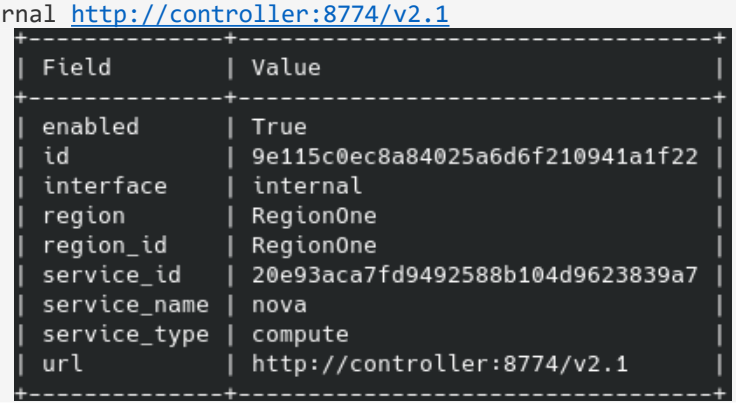
```
# openstack endpoint create --region RegionOne \
  compute public http://controller:8774/v2.1
```



```
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 5e12f0f24ab8431a8381d0cd43341b2e |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 20e93aca7fd9492588b104d9623839a7 |
| service_name | nova                             |
| service_type | compute                          |
| url          | http://controller:8774/v2.1      |
+--------------+----------------------------------+
```

Comando 85 Creación del Endpoint del servicio compute público

```
# openstack endpoint create --region RegionOne \
  compute internal http://controller:8774/v2.1
```



```
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 9e115c0ec8a84025a6d6f210941a1f22 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 20e93aca7fd9492588b104d9623839a7 |
| service_name | nova                             |
| service_type | compute                          |
| url          | http://controller:8774/v2.1      |
+--------------+----------------------------------+
```

Comando 86 Creación del Endpoint del servicio compute interno

```
# openstack endpoint create --region RegionOne \
  compute admin http://controller:8774/v2.1
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 4e3a01f113be4879a0c7305f86c6dee1 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 20e93aca7fd9492588b104d9623839a7 |
| service_name | nova                             |
| service_type | compute                          |
| url          | http://controller:8774/v2.1      |
+--------------+----------------------------------+
```

Comando 87 Creación del Endpoint del servicio compute admin

3.8.1.6. Placement

Ingresar a MariaDB, crear la base de datos **placement** y establecer los permisos respectivos al nuevo usuario creado **placement**.

```
# mysql
MariaDB [(none)]> CREATE DATABASE placement;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'localhost' \
  IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'%' \
  IDENTIFIED BY 'SECRET-PASSWORD';
EXIT;
```

Comando 88 Creación de la base de datos de Placement

Crear el usuario de servicio **placement**, asignar el rol de administrador e incluirlo en el proyecto **service**.

```
# openstack user create --domain default --password-prompt placement
# openstack role add --project service --user placement admin
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | be9312a6343749e1ab9174a184de525e |
| name                | placement                        |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Comando 89 Creación del usuario de servicio de Placement en OpenStack

Crear el servicio de tracking de recursos con el nombre de placement

```
# openstack service create --name placement \
  --description "Placement API" placement
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Placement API                    |
| enabled     | True                             |
| id          | 4ffcbc7dc76545b387070d3ba85359a6 |
| name        | placement                        |
| type        | placement                        |
+-------------+----------------------------------+
```

Comando 90 Creación del servicio image en OpenStack

Crear los endpoints de administración, interno y público respectivamente del componente en la región **RegionOne.**

```
# openstack endpoint create --region RegionOne \
  placement public http://controller:8778
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | dc9e6631682746be8e446a6a79b6291c |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 4ffcbc7dc76545b387070d3ba85359a6 |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://controller:8778           |
+--------------+----------------------------------+
```

Comando 91 Creación del Endpoint del servicio placement público

```
# openstack endpoint create --region RegionOne \
  placement internal http://controller:8778
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | d78c9591ccd0499c94c5f5c6d5d34d56 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 4ffcbc7dc76545b387070d3ba85359a6 |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://controller:8778           |
+--------------+----------------------------------+
```

Comando 92 Creación del Endpoint del servicio placement interno

```
# openstack endpoint create --region RegionOne \
  placement admin http://controller:8778
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 42b9e32200e446298d57e907f31d8d41 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 4ffcbc7dc76545b387070d3ba85359a6 |
| service_name | placement                        |
| service_type | placement                        |
| url          | http://controller:8778           |
+--------------+----------------------------------+
```

Comando 93 Creación del Endpoint del servicio placement admin

Instalar el servicio de tracking de mediante el gestor de paquetes de Ubuntu.

```
# apt install placement-api
```

Comando 94 Instalación de Placement

Editar el archivo de configuración de Placement

```
# nano /etc/placement/placement.conf
```

Comando 95 Edición del archivo de configuración de Placement

Establecer la conexión a la base de datos placement con las credenciales del usuario placement.

```
[placement_database]
connection = mysql+pymysql://placement:SECRET-PASSWORD@controller/placement
```

Configuración 38 Configuración de la base de datos de Placement

Establecer la autenticación mediante el servicio de Identidad Keystone.

```
[api]
auth_strategy = keystone
```

Configuración 39 Configuración del modo de autenticación en Placement

Establecer los datos del token y parámetros para comunicarse con el servicio de Identidad.

```
[keystone_authtoken]
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = placement
password = SECRET-PASSWORD
```

Configuración 40 Configuración de credenciales de Placement

Inicializar la base de datos placement creada anteriormente.

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

Comando 96 Inicialización de la base de datos Placement

Reiniciar el servidor HTTP para cargar los nuevos cambios establecidos en el archivo anterior.

```
# service apache2 restart
```

Comando 97 Reinicio del servicio Apache

Instalar una Cli para el servicio de traking de recursos (placement) mediante el gestor de paquetes de Ubuntu.

```
# apt install python3-osc-placement
```

Comando 98 Instalación del cliente de Placement

3.8.1.7. Nova Parte 2

Instalar los paquetes de nova mediante el gestor de paquetes de Ubuntu:

```
# apt install nova-api nova-conductor nova-novncproxy nova-scheduler
```

Comando 99 Instalación del software de Nova

```
# nano /etc/nova/nova.conf
```

Comando 100 Edición del archivo de configuración de Nova

Se describe los parámetros de la sección DEFAULT a continuación respectivamente:

- Habilitar la API de cómputo y de metadatos.

- Establecer la conexión a RabbitMQ
- Establecer la variable my_ip con el valor de la dirección IP de la interfaz de red conectada a la red física proveedor.

```
[DEFAULT]
enabled_apis = osapi_compute, metadata
transport_url = rabbit://openstack:SECRET-PASSWORD@controller:5672/
my_ip = 192.168.0.201
```

Configuración 41 Configuración general y de RabbitMQ en Nova

Establecer la conexión a la base de datos nova_api con las credenciales del usuario nova.

```
[api_database]
connection = mysql+pymysql://nova:SECRET-PASSWORD@controller/nova_api
```

Configuración 42 Configuración de la base de datos nova_api en Nova

Establecer la conexión a la base de datos nova con las credenciales del usuario nova.

```
[database]
connection = mysql+pymysql://nova:SECRET-PASSWORD@controller/nova
```

Configuración 43 Configuración de la base de datos nova en Nova

Establecer la autenticación mediante el servicio de Identidad Keystone.

```
[api]
auth_strategy = keystone
```

Configuración 44 Configuración del modo de autenticación en Nova

Establecer los datos del token y parámetros para comunicarse con el servicio de Identidad.

```
[keystone_authtoken]
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = SECRET-PASSWORD
```

Configuración 45 Configuración de credenciales de Nova

Habilitar la consola VNC especificando la dirección IP definida anteriormente.

```
[vnc]
enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = $my_ip
novncproxy_base_url = http://smartlab-epn.com:64002/vnc_auto.html
```

Configuración 46 Configuración de VNC en Nova

Establecer la dirección y puerto del servidor de imagen.

```
[glance]
api_servers = http://controller:9292
```

Configuración 47 Configuración de Glance en Nova

Inidicar el directorio para la ejecución de la libería de Oslo_concurrency.

```
[oslo_concurrency]
lock_path = /var/run/nova
```

Configuración 48 Configuración de Oslo en Nova

Establecer los parámetros de autenticación del servicio de red (neutron) a Keystone. Se debe especificar el string secreto para el proxy de metadatos definido en la configuración del servidor de metadatos.

```
[neutron]
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = SECRET-PASSWORD
service_metadata_proxy = true
metadata_proxy_shared_secret = SECRET-STRING
```

Configuración 49 Configuración de Neutron en Nova

Establecer los parámetros de autenticación del servicio de tracking de recursos (placement) a Keystone.

```
[placement]
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = SECRET-PASSWORD
```

Configuración 50 Configuración de Placement en Nova

Especificar la región en donde el servicio de almacenamiento en bloques (cinder) se ejecuta

```
[cinder]
os_region_name = RegionOne
```

Configuración 51 Configuración de Cinder en Nova

Se inicializará las bases de datos del servicio de cómputo descritos a continuación respectivamente:

- Inicializar la base de datos nova-api
- Registrar la base de datos cell0
- Crear la celda cell1
- Inicializar la base de datos nova

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

Comando 101 Inicialización de Nova

Reiniciar los servicios de cómputo.

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

Comando 102 Reinicio de los servicios de Nova

```
# su -s /bin/sh -c “nova-manage cell_v2 list_cells” nova
+-------+--------------------------------------+------------------------------------------+-------------------------------------------+----------+
|  Name |                 UUID                 |              Transport URL               |           Database Connection             | Disabled |
+-------+--------------------------------------+------------------------------------------+-------------------------------------------+----------+
| cell0 | 00000000-0000-0000-0000-000000000000 |                  none:/                  | sqlite://var/lib/nova/nova.sqlite_cell0 |  False   |
| cell1 | e9d564f5-04f5-4a38-a025-08c24b7547b6 | rabbit://openstack:****@controller:5672/ |     sqlite://var/lib/nova/nova.sqlite     |  False   |
+-------+--------------------------------------+------------------------------------------+-------------------------------------------+----------+
```

Comando 103 Inicialización de celdas de Nova

```
# su -s /bin/sh -c “nova-manage cell_v2 discover_hosts --verbose” nova
Found 2 cell mappings.
Skipping cell0 since it does not contain hosts.
Getting computes from cell 'cell1': e9d564f5-04f5-4a38-a025-08c24b7547b6
Checking host mapping for compute host 'compute': 9d29fed3-5d99-4ff2-8fcb-ff98cba36330
Creating host mapping for compute host 'compute': 9d29fed3-5d99-4ff2-8fcb-ff98cba36330
Found 1 unmapped computes in cell: e9d564f5-04f5-4a38-a025-08c24b7547b6
```

Comando 104 Búsqueda de hosts de cómputo

3.8.1.8. Cinder

Ingresar a MariaDB, crear las bases de datos **cinder** y establecer los permisos respectivos al nuevo usuario creado **cinder**.

```
# mysql
MariaDB [(none)]> CREATE DATABASE cinder;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' \
  IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' \
  IDENTIFIED BY 'SECRET-PASSWORD';
MariaDB [(none)]> EXIT;
```

Comando 105 Creación de la base de datos Cinder

Crear el usuario de servicio **cinder**, asignar el rol de administrador e incluirlo en el proyecto **service**.

```
# openstack user create --domain default --password-prompt cinder
# openstack role add --project service --user cinder admin
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 7fa2a8089aa7410bb415712fd248be19 |
| name                | cinder                           |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Comando 106 Creación del usuario de servicio Cinder en OpenStack

Crear el servicio de volumen con el nombre de cinderv3

```
# openstack service create --name cinderv3 \
  --description "OpenStack Block Storage" volumev3
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Block Storage          |
| enabled     | True                             |
| id          | 7c6da928cb3b4b6bb2e59cf8ae696caa |
| name        | cinderv3                         |
| type        | volumev3                         |
+-------------+----------------------------------+
```

Comando 107 Creación del servicio volumev3 en OpenStack

Crear los endpoints de administración, interno y público respectivamente del componente en la región **RegionOne.**

```
# openstack endpoint create --region RegionOne \
  volumev3 public http://controller:8776/v3/%\(project_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | a6ae28e126304ab2aaed822f225f7483        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 7c6da928cb3b4b6bb2e59cf8ae696caa        |
| service_name | cinderv3                                |
| service_type | volumev3                                |
| url          | http://controller:8776/v3/%(project_id)s |
+--------------+-----------------------------------------+
```

Comando 108 Creación del Endpoint del servicio volumev3 público

```
# openstack endpoint create --region RegionOne \
  volumev3 internal http://controller:8776/v3/%\(project_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | f2b67274170c45e6ba71871d9ba91644        |
| interface    | internal                                |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 7c6da928cb3b4b6bb2e59cf8ae696caa        |
| service_name | cinderv3                                |
| service_type | volumev3                                |
| url          | http://controller:8776/v3/%(project_id)s |
+--------------+-----------------------------------------+
```

Comando 109 Creación del Endpoint del servicio volumev3 interno

```
# openstack endpoint create --region RegionOne \
  volumev3 admin http://controller:8776/v3/%\(project_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | e374cc326919484fa18261d3b052a29e        |
| interface    | admin                                   |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 7c6da928cb3b4b6bb2e59cf8ae696caa        |
| service_name | cinderv3                                |
| service_type | volumev3                                |
| url          | http://controller:8776/v3/%(project_id)s |
+--------------+-----------------------------------------+
```

Comando 110 Creación del Endpoint del servicio volumev3 admin

Instalar el API de cinder y el demonio que selecciona el proveedor óptimo para almacenamiento.

```
# apt install cinder-api cinder-scheduler
```

Comando 111 Instalación de Cinder

```
# nano /etc/cinder/cinder.conf
```

Comando 112 Edición del archivo de configuración de Cinder

Establecer la conexión a la base de datos cinder con las credenciales del usuario cinder.

```
[database]
connection = mysql+pymysql://cinder:SECRET-PASSWORD@controller/cinder
```

Configuración 52 Configuración de la base de datos de Cinder

Se describe los parámetros de la sección DEFAULT a continuación respectivamente:

- Establecer la variable my_ip con el valor de la dirección IP de la interfaz de red conectada a la red física proveedor.
- Establecer la conexión a RabbitMQ.
- Establecer la autenticación mediante el servicio de Identidad Keystone.

```
[DEFAULT]
my_ip = 192.168.0.201
transport_url = rabbit://openstack:SECRET-PASSWORD@controller
auth_strategy = keystone
```

Configuración 53 Configuración general y de RabbitMQ en Cinder

Establecer los datos del token y parámetros para comunicarse con el servicio de Identidad.

```
[keystone_authtoken]
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = cinder
password = SECRET-PASSWORD
```

Configuración 54 Configuración de credenciales de Cinder

Inidicar el directorio para la ejecución de la libería de Oslo_concurrency.

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

Configuración 55 Configuración de Oslo en Cinder

Inicializar la base de datos cinder creada anteriormente.

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

Comando 113 Inicialización de la base de datos Cinder

Reiniciar los servicios para cargar los nuevos parámetros de configuración establecidos anteriormente.

```
# service cinder-scheduler restart
# service apache2 restart
```

Comando 114 Reinicio del servicio Cinder

3.8.1.9. Horizon

El Dashboard de Openstack permite tener un cliente web amigable con el usuario integrado con los demás componentes.

Instalar el Dashboard de Openstack mediante el gestor de paquetes de Ubuntu.

```
# apt install openstack-dashboard
```

Comando 115 Instalación de Horizon

```
# nano /etc/openstack-dashboard/local_settings.py
```

Comando 116 Edición del archivo de configuración de Horizon

Especificar el nombre del host

```
OPENSTACK_HOST = "controller"
```

Configuración 56 Configuración del host en Horizon

Permitir a todos los Hosts el acceso al dashboard.

```
ALLOWED_HOSTS = ['*']
```

Configuración 57 Configuración de acceso al Dashboard en Horizon

Especificar memcached como el servicio de cache.

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

Configuración 58 Configuración de Memcached en Horizon

Establecer la URL para comunicarse mediante HTTP con el servicio de Identidad (Keystone)

```
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
```

Configuración 59 Configuración de Keystone en Horizon

Especificar las versiones implementadas en el prototipo de cloud computing del servicio de identidad, imagen y volumen.

```
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
    "volume": 3,
}
```

Configuración 60 Configuración de Cinder en Horizon

Asignar el dominio default como dominio predeterminado. Todas las operaciones realizadas serán efectuadas solamente en el dominio Default.

```
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"
```

Configuración 61 Configuración de dominio predeterminado en Horizon

Asignar automáticamente el rol user a los nuevos usuarios creados desde el Dashboard.

```
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
```

Configuración 62 Configuración del usuario predeterminado en Horizon

Especificar el huso horario geográfico.

```
TIME_ZONE = "America/Guayaquil"
```

Configuración 63 Configuración del huso horario en Horizon

```
# nano /etc/apache2/conf-available/openstack-dashboard.conf
```

Comando 117 Edición del archivo de configuración de Horizon en Apache

Asegurarse de que el grupo de aplicación de WSGI es global. Si la directiva no está establecida, el Dashboard no será usuable y tendrá largos periodos de latencia al momento de iniciar sesión.

```
WSGIApplicationGroup %{GLOBAL}
```

Configuración 64 Configuración del grupo de aplicación WSGI en Apache

Actualizar la configuración del servidor HTTP y de los ajustes del Dashboard establecidos anteriormente.

```
# systemctl reload apache2.service
```

Comando 118 Recarga de configuración de Apache

3.8.2. Nodo Compute

3.8.2.1. Neutron

Instalar el agente de Linuxbridge

```
# apt install neutron-linuxbridge-agent
```

Comando 119 Instalación de agente de Linuxbridge

```
# nano /etc/neutron/neutron.conf
```

Comando 120 Edición del archivo de configuración de Neutron

Establecer la conexión de RabbitMQ y establecer la autenticación mediante el servicio de Identidad Keystone.

```
[DEFAULT]
transport_url = rabbit://openstack:SECRET-PASSWORD@controller
auth_strategy = keystone
```

Configuración 65 Configuración general de Neutron

Establecer la conexión a la base de datos nova con las credenciales del usuario nova.

```
[database]
connection = mysql+pymysql://neutron:SECRET-PASSWORD@controller/neutron
```

Configuración 66 Configuración de la base de datos en Neutron

Establecer los datos del token y parámetros para comunicarse con el servicio de Identidad.

```
[keystone_authtoken]
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = SECRET-PASSWORD
```

Configuración 67 Configuración de credenciales de Neutron

Indicar el directorio para la ejecución de la libería de Oslo_concurrency.

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

Configuración 68 Configuración de Oslo en Neutron

```
# nano /etc/neutron/plugins/ml2/linuxbridge_agent.ini
```

Comando 121 Edición del archivo de configuración del plugin del agente Linuxbridge de Neutron

Especificar la interfaz que pertenece a la red proveedora física. La red proveedora virtual será mapeada en la red proveedora física.

```
[linux_bridge]
physical_interface_mappings = provider:enp1s0
```

Configuración 69 Configuración de la red proveedora en el agente de Linuxbridge de Neutron

En la sección de VXLAN se especifica los siguientes parámetros respectivamente:

- Habilitar las redes sobrepuestas VXLAN (overlay)
- Especificar la dirección IP de la interfaz perteneciente a la red proveedora física.
- Habilitar el driver L2 Population.

```
[vxlan]
enable_vxlan = true
local_ip = 192.168.0.202
l2_population = true
```

Configuración 70 Configuración VXLAN en LinuxBridge

Habilitar los grupos de seguridad y especificar el uso driver de firewall de Iptables

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

Configuración 71 Configuración el uso de IpTables en LinuxBridge

Cargar el módulo de Kernel **br_netfilter** para habilitar el soporte de red bridging y facilitar la comunicación entre VxLAN.

```
# modprobe br_netfilter
# echo "net.bridge.bridge-nf-call-iptables = 1" >> /etc/sysctl.conf
# echo "net.bridge.bridge-nf-call-ip6tables = 1" >> /etc/sysctl.conf
# sysctl -p /etc/sysctl.conf
```

Comando 122 Activación del módulo br_netfilter

3.8.2.2. *Nova*

El servicio de cómputo utiliza el hypervisor Quick EMUlator (QEMU) con la extensión kernel-basede VM (KVM), ya que el nodo de cómputo permite la acelaración de hardware para las máquinas virtuales.

Instalar el servicio de cómputo.

```
# apt install nova-compute
```

Comando 123 Instalación de Nova

```
# nano /etc/nova/nova.conf
```

Comando 124 Edición del archivo de configuración de Nova

Se describe los parámetros de la sección DEFAULT a continuación respectivamente:

- Habilitar la API de cómputo y de metadatos.
- Establecer la conexión a RabbitMQ
- Establecer la variable my_ip con el valor de la dirección IP de la interfaz de red conectada a la red física proveedor.
- Habilitar la funcionalidad para auditaría sobre el uso en un rango de tiempo del nodo de cómputo.
- Establecer el periodo de tiempo con la que se ejecuta la tarea de recolección de datos para auditoría

```
[DEFAULT]
enabled_apis = osapi_compute,metadata
compute_driver = libvirt.LibvirtDriver
transport_url = rabbit://openstack:SECRET-PASSWORD@controller
my_ip = 192.168.0.202
instance_usage_audit = True
instance_usage_audit_period = hour
```

Configuración 72 Configuración general de Nova

Establecer la autenticación mediante el servicio de Identidad Keystone.

```
[api]
auth_strategy = keystone
```

Configuración 73 Configuración del método de autenticación de Nova

Establecer los datos del token y parámetros para comunicarse con el servicio de Identidad.

```
[keystone_authtoken]
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = SECRET-PASSWORD
```

Configuración 74 Configuración de credenciales de Nova

La sección VNC se establecen los siguientes parámetros respectivamente:

- Habilitar la consola VNC.
- Permitir al servicio de VNC procesar peticiones de cualquier dirección IP.
- Especificar la dirección IP definida anteriormente.
- Especificar la URL que se retornará al usuario. Esta contiene el nombre de dominio público accesible desde internet smartlab-epn.com y el puerto 64002. El puerto 64002 es redirigido al puerto 6080 por medio de reenvío de puertos (Port forwarding) del router externo físico.

```
[vnc]
enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = $my_ip
novncproxy_base_url = http://smartlab-epn.com:64002/vnc_auto.html
```

Configuración 75 Configuración de VNC en Nova

Establecer la dirección y puerto del servidor de imagen.

```
[glance]
api_servers = http://controller:9292
```

Configuración 76 Configuración de Glance en Nova

Indicar el directorio para la ejecución de la libería de Oslo_concurrency.

```
[oslo_concurrency]
lock_path = /var/run/nova
```

Configuración 77 Configuración de Oslo en Nova

Establecer los parámetros de autenticación del servicio de tracking de recursos (placement) a Keystone.

```
[placement]
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = SECRET-PASSWORD
```

Configuración 78 Configuración de Placement en Nova

Establecer los parámetros de autenticación del servicio de red (neutron) a Keystone.

```
[neutron]
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = SECRET-PASSWORD
```

Configuración 79 Configuración de Neutron en Nova

Especificar la región en donde el servicio de almacenamiento en bloques (cinder) se ejecuta

```
[cinder]
os_region_name = RegionOne
```

Configuración 80 Configuración de Cinder en Nova

Habilitar las notificaciones de las instancias al servicio de cómputo.

```
[notifications]
notify_on_state_change = vm_and_task_state
```

Configuración 81 Configuración de notificaciones en Nova

Reiniciar el servicio de cómputo.

```
# service nova-compute start
```

Comando 125 Reinicio del servicio Nova

3.8.2.3. *Cinder*

Ya que el nodo de almacenamiento utiliza LVM en el disco del sistema operativo, se debe modificar el filtro LVM para incluir solamente el disco del sistema operativo.

```
# nano /etc/lvm/lvm.conf
```

Comando 126 Edición de configuración de LVM

Establecer el filtro para aceptar el disco del sistema operativo y rechazar todos los demás.

```
devices {
  filter = [ "a/vda/", "r/.*/"]
}
```

Configuración 82 Configuración de discos LVM

3.8.3. Nodo Block Storage

3.8.3.1. *Cinder*

Instalar LVM2 y las herramientas para gestionar el sobreuso de espacio de discos.

```
# apt install lvm2 thin-provisioning-tools
```

Comando 127 Instalación de LVM

Crear un volumen físico LVM en el disco /dev/vdb.

```
# pvcreate /dev/vdb
```

Comando 128 Creación de un volumen LVM

Crear un grupo de volumen LVM en el disco /dev/vdb.

```
# vgcreate cinder-volumes /dev/sdb
```

Comando 129 Creación de un grupo de volumen LVM

```
# nano /etc/lvm/lvm.conf
```

Comando 130 Edición del archivo de configuración de LVM

Establecer el filtro para aceptar el disco **/dev/vda** del sistema operativo y el disco **/dev/vdb** que será utilizado por el servicio de almacenamiento por bloques (Cinder)

```
devices {
  filter = [ "a/vda/", "a/vdb/", "r/.*/"]
}
```

Configuración 83 Configuración de discos LVM

Instalar el servicio de almacenamiento por bloques

```
# apt install cinder-volume
```

Comando 131 Instalación de Cinder

```
# nano /etc/cinder/cinder.conf
```

Comando 132 Edición del archivo de configuración de Cinder

Se describe los parámetros de la sección DEFAULT a continuación respectivamente:

- Establecer la conexión a RabbitMQ
- Establecer la autenticación mediante el servicio de Identidad Keystone.
- Habilitar el backend LVM
- Especificar la URL del servicio de imagen.
- Especificar el driver de almacenamiento de objetos (Swift) para realizar copias de seguridad.
- Especificar la URL del servicio de almacenamiento de objetos (Swift) para las copias de seguridad.

```
[DEFAULT]
transport_url = rabbit://openstack:SECRET-PASSWORD@controller
auth_strategy = keystone
enabled_backends = lvm
glance_api_servers = http://controller:9292
backup_driver = cinder.backup.drivers.swift.SwiftBackupDriver
backup_swift_url = http://controller:8080/v1
```

Configuración 84 Configuración general de Cinder

Establecer la conexión a la base de datos cinder con las credenciales del usuario cinder.

```
[database]
connection = mysql+pymysql://cinder:SECRET-PASSWORD@controller/cinder
```

Configuración 85 Configuración de la base de datos en Cinder

Establecer los datos del token y parámetros para comunicarse con el servicio de Identidad.

```
[keystone_authtoken]
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = cinder
password = SECRET-PASSWORD
```

Configuración 86 Configuración de credenciales de Cinder

Establecer los parámetros de LVM descritos a continuación respectivamente:

- Especificar el uso del driver de volúmenes LVM.
- Nombre del grupo de volúmenes que contendrán los volúmenes exportados.
- Especificar iSCSI como el protocolo de red para la conexión de discos.
- Establecer tgtadm como herramienta de administración de servidor iSCSI.

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

Configuración 87 Configuración LVM en Cinder

Indicar el directorio para la ejecución de la libería de Oslo_concurrency.

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

Configuración 88 Configuración de Oslo en Cinder

Instalar las herramientas de administración de servidor iSCSI.

```
# apt install tgt
```

Comando 133 Instalación de TGT

Indicar el directorio donde se crearán los volúmenes LVM del servicio de almacenamiento por bloques (Cinder)

```
# echo 'include /var/lib/cinder/volumes/*' >> /etc/tgt/conf.d/cinder_tgt.conf
```

Comando 134 Creación de archivo de configuración de TGT

Incluir el target anterior a la lista la lista de servidores iSCSI.

```
# echo 'include /etc/tgt/conf.d/cinder_tgt.conf' >> /etc/tgt/targets.conf
```

Comando 135 Creación de Targets TGT

Reiniciar el servicio tgt y de almacenamiento por bloques (Cinder).

```
# service tgt restart
# service cinder-volume restart
```

Comando 136 Reinicio del servicio Cinder

Instalar la herramienta para realizar copias de seguridad.

```
# apt install cinder-backup
```

Comando 137 Instalación de Cinder Backup

3.8.4. Nodo Object1

3.8.4.1. *Swift*

Instalar las herramientas para crear y administrar el sistema de archivos XFS y la herramienta rsync para sincronizar los archivos alrededor del clúster.

```
# sudo apt-get install xfsprogs rsync
```

Comando 138 Instalación de herramientas de sincronización de archivos

Crear el sistema de archivos XFS en los discos vacíos **/dev/vdb** y **/dev/vdc**

```
# mkfs.xfs /dev/vdb
# mkfs.xfs /dev/vdc
```

Comando 139 Creación de XFS en discos vacíos

Crear los directorios para montar los discos respectivamente.

```
# mkdir -p /srv/node/vdb
# mkdir -p /srv/node/vdc
```

Comando 140 Creación de directorios para montar los discos

Editar el archivo fstab en donde se agregarán las opciones de montaje de los discos adicionales.

```
# nano /etc/fstab
```

Comando 141 Edición del archivo fstab

Registrar los discos con su respectivo directorio de monto y configuración

```
UUID=c9063b92-e8c4-4c69-960a-2de1f1cdfa4d /srv/node/vdb xfs noatime 0 2
UUID=3814164f-d7d0-4384-833c-aa89b86b1b42 /srv/node/vdc xfs noatime 0 2
```

Configuración 89 Registro de los discos en el archivo fstab

Montar los discos en los directorios **/srv/node/vdb** y **/srv/node/vdc**

```
# mount /srv/node/vdb
# mount /srv/node/vdc
```

Comando 142 Monto de los discos en sus respectivos directorios

Editar el archivo rsyncd para poder sincronizar los archivos del clúster.

```
# nano /etc/rsyncd.conf
```

Comando 143 Edición del archivo la configuración de Rsync

Definir la siguiente configuración general:

- Establecer al usuario swift como ejecutor del servicio de rsync
- Establecer el grupo swift por razones de permisos
- Indicar el archivo donde se almacenará el log generado por el servicio rsync
- Inidicar dónde se creará el archivo de identificación del servicio
- Indicar la dirección IP por donde recibirá las peticiones

```
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = 192.168.122.5
```

Configuración 90 Configuración general de Rsync

Definir los parámetros de configuración para el anillo de cuenta del servicio de almacenamiento de objetos:

- Establecer el número máximo de 2 conexiones simultáneas para no perjudicar el rendimiento.
- Definir la ruta del directorio donde estarán los discos montados.
- Habilitar los permisos de escritura
- Definir el archivo de bloqueo para trabajar con concurrentemente y evitar corrupción de datos.

```
[account]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/account.lock
```

Configuración 91 Configuración del anillo de cuenta en Rsync

Definir los parámetros de configuración para el anillo de contenedor del servicio de almacenamiento de objetos:

- Establecer el número máximo de 2 conexiones simultáneas para no perjudicar el rendimiento.
- Definir la ruta del directorio donde estarán los discos montados.
- Habilitar los permisos de escritura
- Definir el archivo de bloqueo para trabajar con concurrentemente y evitar corrupción de datos.

```
[container]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/container.lock
```

Configuración 92 Configuración del anillo de contenedor en Rsync

Definir los parámetros de configuración para el anillo de objecto del servicio de almacenamiento de objetos:

- Establecer el número máximo de 2 conexiones simultáneas para no perjudicar el rendimiento.
- Definir la ruta del directorio donde estarán los discos montados.
- Habilitar los permisos de escritura
- Definir el archivo de bloqueo para trabajar con concurrentemente y evitar corrupción de datos.

```
[object]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/object.lock
```

Configuración 93 Configuración del anillo de objecto en Rsync

Editar el archivo de Rsync

```
# nano /etc/default/rsync
```

Comando 144 Edición del archivo de Rsync

Habilitar el servicio de rsync

```
RSYNC_ENABLE=true
```

Configuración 94 Configuración para activar Rsync

Reiniciar el servicio de rsync para cargar los nuevos parámetros de configuración.

```
# service rsync start
```

Comando 145 Reinicio del servicio Rsync

Instalar los 3 componentes (cuenta, contenedor y objeto) del servicio de almacenamiento de objetos.

```
# apt-get install swift swift-account swift-container swift-object
```

Comando 146 Instalación del software de Swift

Editar el archivo de configuración del anillo de cuenta de Swift

```
# nano /etc/swift/account-server.conf
```

Comando 147 Edición del archivo de configuración del anillo de cuenta de Swift

Se describe los parámetros de la sección DEFAULT a continuación respectivamente:

- Especificar la dirección IP de la interfaz de red por donde se expondrá el servicio.
- Especificar el puerto por donde se establecerá la conexión.
- Establecer 2 hilos para la ejecución del servicio.
- Establecer al usuario swift como ejecutor del servicio de cuenta.
- Indicar el directorio donde se encuentran los archivos de configuración.
- Indicar el directorio donde se encuentran los discos que pertenecen al clúster.
- Habilitar la opción para chequear si el disco está montado y así prevenir que se escriba en el disco primario.

```
[DEFAULT]
bind_ip = 192.168.122.5
bind_port = 6202
workers = 2
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

Configuración 95 Configuración general del anillo de cuenta de Swift

Agregar el módulo account-server a la lista.

```
[pipeline:main]
pipeline = healthcheck recon account-server
```

Configuración 96 Configuración de módulos del anillo de cuenta de Swift

Editar el archivo de configuración del anillo de contenedor de Swift

```
# nano /etc/swift/container-server.conf
```

Comando 148 Edición del archivo de configuración del anillo de contenedor de Swift

Se describe los parámetros de la sección DEFAULT a continuación respectivamente:

- Especificar la dirección IP de la interfaz de red por donde se expondrá el servicio.
- Especificar el puerto por donde se establecerá la conexión.
- Establecer 2 hilos para la ejecución del servicio.
- Establecer al usuario swift como ejecutor del servicio de cuenta.
- Indicar el directorio donde se encuentran los archivos de configuración.
- Indicar el directorio donde se encuentran los discos que pertenecen al clúster.
- Habilitar la opción para chequear si el disco está montado y así prevenir que se escriba en el disco primario.

```
[DEFAULT]
bind_ip = 192.168.122.5
bind_port = 6201
workers = 2
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

Configuración 97 Configuración general del anillo de contendor de Swift

Agregar el módulo container-server a la lista.

```
[pipeline:main]
pipeline = healthcheck recon container-server
```

Configuración 98 Configuración de módulos del anillo de contenedor de Swift

Editar el archivo de configuración del anillo de objeto de Swift

```
# nano /etc/swift/object-server.conf
```

Comando 149 Edición del archivo de configuración del anillo de objeto de Swift

Se describe los parámetros de la sección DEFAULT a continuación respectivamente:

- Especificar la dirección IP de la interfaz de red por donde se expondrá el servicio.
- Especificar el puerto por donde se establecerá la conexión.
- Establecer 2 hilos para la ejecución del servicio.
- Establecer al usuario swift como ejecutor del servicio de cuenta.
- Indicar el directorio donde se encuentran los archivos de configuración.
- Indicar el directorio donde se encuentran los discos que pertenecen al clúster.
- Habilitar la opción para chequear si el disco está montado y así prevenir que se escriba en el disco primario.

```
[DEFAULT]
bind_ip = 192.168.122.5
bind_port = 6200
workers = 2
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

Configuración 99 Configuración general del anillo de objeto de Swift

Agregar el módulo object-server a la lista.

```
[pipeline:main]
pipeline = healthcheck recon object-server
```

Configuración 100 Configuración de módulos del anillo de objeto de Swift

Establecer la propiedad del directorio donde se encuentran los discos del clúster al usuario swift y grupo swift de forma recursiva, para que procesos tengan los permisos necesarios para ejecutarse correctamente.

```
# chown -R swift:swift /srv/node
```

Comando 150 Cambio de propiedad del directorio /srv/node y sus respectivos archivos

Crear el directorio para cache, asignar la propiedad a root y el grupo de Swift y establecer permisos de escritura, lectura y ejecución para dichas identidades.

```
# mkdir -p /var/cache/swift
# chown -R root:swift /var/cache/swift
# chmod -R 775 /var/cache/swift
```

Comando 151 Creación y configuración del directorio para cache de Swift

Establecer la propiedad de los archivos del directorio de configuración a root y el grupo de swift, con el fin de proteger los datos de configuración confidenciales.

```
# chown -R root:swift /etc/swift
```

Comando 152 Cambio de propiedad del directorio /etc/swift para el usuario root y grupo swift

Inicializar todos los procesos que forman parte del servicio de object storare (Swift).

```
# swift-init all start
```

Comando 153 Inicialización de Swift

3.8.5. Nodo Object2

3.8.5.1. *Swift*

Instalar las herramientas para crear y administrar el sistema de archivos XFS y la herramienta rsync para sincronizar los archivos alrededor del clúster.

```
# sudo apt-get install xfsprogs rsync
```

Comando 154 Instalación de herramientas de sincronización de archivos

Crear el sistema de archivos XFS en los discos vacíos **/dev/vdb** y **/dev/vdc**

```
# mkfs.xfs /dev/vdb
# mkfs.xfs /dev/vdc
```

Comando 155 Creación de XFS en discos vacíos

Crear los directorios para montar los discos respectivamente.

```
# mkdir -p /srv/node/vdb
# mkdir -p /srv/node/vdc
```

Comando 156 Creación de directorios para montar los discos

Registrar los discos con su respectivo directorio de monto y configuración

```
# nano /etc/fstab
```

Comando 157 Edición del archivo fstab

Registrar los discos con su respectivo directorio de monto y configuración

```
UUID=c9063b92-e8c4-4c69-960a-2de1f1cdfa4d /srv/node/vdb xfs noatime 0 2
UUID=3814164f-d7d0-4384-833c-aa89b86b1b42 /srv/node/vdc xfs noatime 0 2
```

Configuración 101 Registro de los discos en el archivo fstab

Montar los discos en los directorios **/srv/node/vdb** y **/srv/node/vdc**

```
# mount /srv/node/vdb
# mount /srv/node/vdc
```

Comando 158 Monto de los discos en sus respectivos directorios

```
# nano /etc/rsyncd.conf
```

Comando 159 Edición del archivo la configuración de Rsync

Definir la siguiente configuración general:

- Establecer al usuario swift como ejecutor del servicio de rsync
- Establecer el grupo swift por razones de permisos
- Indicar el archivo donde se almacenará el log generado por el servicio rsync
- Inidicar dónde se creará el archivo de identificación del servicio
- Indicar la dirección IP por donde recibirá las peticiones

```
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = 192.168.122.6
```

Configuración 102 Configuración general de Rsync

Definir los parámetros de configuración para el anillo de cuenta del servicio de almacenamiento de objetos:

- Establecer el número máximo de 2 conexiones simultáneas para no perjudicar el rendimiento.
- Definir la ruta del directorio donde estarán los discos montados.
- Habilitar los permisos de escritura
- Definir el archivo de bloqueo para trabajar con concurrentemente y evitar corrupción de datos.

```
[account]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/account.lock
```

Configuración 103 Configuración del anillo de cuenta en Rsync

Definir los parámetros de configuración para el anillo de contenedor del servicio de almacenamiento de objetos:

- Establecer el número máximo de 2 conexiones simultáneas para no perjudicar el rendimiento.
- Definir la ruta del directorio donde estarán los discos montados.
- Habilitar los permisos de escritura
- Definir el archivo de bloqueo para trabajar con concurrentemente y evitar corrupción de datos.

```
[container]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/container.lock
```

Configuración 104 Configuración del anillo de contenedor en Rsync

Definir los parámetros de configuración para el anillo de objecto del servicio de almacenamiento de objetos:

- Establecer el número máximo de 2 conexiones simultáneas para no perjudicar el rendimiento.
- Definir la ruta del directorio donde estarán los discos montados.

- Habilitar los permisos de escritura
- Definir el archivo de bloqueo para trabajar con concurrentemente y evitar corrupción de datos.

```
[object]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/object.lock
```

Configuración 105 Configuración del anillo de objecto en Rsync

Editar el archivo rsyncd para poder sincronizar los archivos del clúster.

```
# nano /etc/default/rsync
```

Comando 160 Edición del archivo de Rsync

Habilitar el servicio de rsync

```
RSYNC_ENABLE=true
```

Configuración 106 Configuración para activar Rsync

Reiniciar el servicio de rsync para cargar los nuevos parámetros de configuración.

```
# service rsync start
```

Comando 161 Reinicio del servicio Rsync

Instalar los 3 componentes (cuenta, contenedor y objeto) del servicio de almacenamiento de objetos.

```
# apt-get install swift swift-account swift-container swift-object
```

Comando 162 Instalación del software de Swift

Editar el archivo de configuración del anillo de cuenta de Swift

```
# nano /etc/swift/account-server.conf
```

Comando 163 Edición del archivo de configuración del anillo de cuenta de Swift

Se describe los parámetros de la sección DEFAULT a continuación respectivamente:

- Especificar la dirección IP de la interfaz de red por donde se expondrá el servicio.
- Especificar el puerto por donde se establecerá la conexión.

- Establecer 2 hilos para la ejecución del servicio.
- Establecer al usuario swift como ejecutor del servicio de cuenta.
- Indicar el directorio donde se encuentran los archivos de configuración.
- Indicar el directorio donde se encuentran los discos que pertenecen al clúster.
- Habilitar la opción para chequear si el disco está montado y así prevenir que se escriba en el disco primario.

```
[DEFAULT]
bind_ip = 192.168.122.6
bind_port = 6202
workers = 2
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

Configuración 107 Configuración general del anillo de cuenta de Swift

Agregar el módulo account-server a la lista.

```
[pipeline:main]
pipeline = healthcheck recon account-server
```

Configuración 108 Configuración de módulos del anillo de cuenta de Swift

Editar el archivo de configuración del anillo de contenedor de Swift

```
# nano /etc/swift/container-server.conf
```

Comando 164 Edición del archivo de configuración del anillo de contenedor de Swift

Se describe los parámetros de la sección DEFAULT a continuación respectivamente:

- Especificar la dirección IP de la interfaz de red por donde se expondrá el servicio.
- Especificar el puerto por donde se establecerá la conexión.
- Establecer 2 hilos para la ejecución del servicio.
- Establecer al usuario swift como ejecutor del servicio de cuenta.
- Indicar el directorio donde se encuentran los archivos de configuración.
- Indicar el directorio donde se encuentran los discos que pertenecen al clúster.
- Habilitar la opción para chequear si el disco está montado y así prevenir que se escriba en el disco primario.

```
[DEFAULT]
bind_ip = 192.168.122.6
bind_port = 6201
workers = 2
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

Configuración 109 Configuración general del anillo de contendor de Swift

Agregar el módulo account-server a la lista.

```
[pipeline:main]
pipeline = healthcheck recon container-server
```

Configuración 110 Configuración de módulos del anillo de contenedor de Swift

Editar el archivo de configuración del anillo de objeto de Swift

```
# nano /etc/swift/object-server.conf
```

Comando 165 Edición del archivo de configuración del anillo de objeto de Swift

Se describe los parámetros de la sección DEFAULT a continuación respectivamente:

- Especificar la dirección IP de la interfaz de red por donde se expondrá el servicio.
- Especificar el puerto por donde se establecerá la conexión.
- Establecer 2 hilos para la ejecución del servicio.
- Establecer al usuario swift como ejecutor del servicio de cuenta.
- Indicar el directorio donde se encuentran los archivos de configuración.
- Indicar el directorio donde se encuentran los discos que pertenecen al clúster.
- Habilitar la opción para chequear si el disco está montado y así prevenir que se escriba en el disco primario.

```
[DEFAULT]
bind_ip = 192.168.122.6
bind_port = 6200
workers = 2
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

Configuración 111 Configuración general del anillo de objeto de Swift

Agregar el módulo account-server a la lista.

```
[pipeline:main]
pipeline = healthcheck recon object-server
```

Configuración 112 Configuración de módulos del anillo de objeto de Swift

Establecer la propiedad del directorio donde se encuentran los discos del clúster al usuario swift y grupo swift de forma recursiva, para que procesos tengan los permisos necesarios para ejecutarse correctamente.

```
# chown -R swift:swift /srv/node
```

Comando 166 Cambio de propiedad del directorio /srv/node y sus respectivos archivos

Crear el directorio para cache, asignar la propiedad a root y el grupo de Swift y establecer permisos de escritura, lectura y ejecución para dichas identidades.

```
# mkdir -p /var/cache/swift
# chown -R root:swift /var/cache/swift
# chmod -R 775 /var/cache/swift
```

Comando 167 Creación y configuración del directorio para cache de Swift

Establecer la propiedad de los archivos del directorio de configuración a root y el grupo de swift, con el fin de proteger los datos de configuración confidenciales.

```
# chown -R root:swift /etc/swift
```

Comando 168 Cambio de propiedad del directorio /etc/swift para el usuario root y grupo swift

Inicializar todos los procesos que forman parte del servicio de object storare (Swift).

```
# swift-init all start
```

Comando 169 Inicialización de Swift

3.9. Exposición del servicio a Internet

3.9.1. Dominio

Compra del dominio **smartlab-epn.com** en el registrador de dominio GoDaddy, en la ilustración 45 se compra el dominio smartlab-epn.com en GoDaddy.

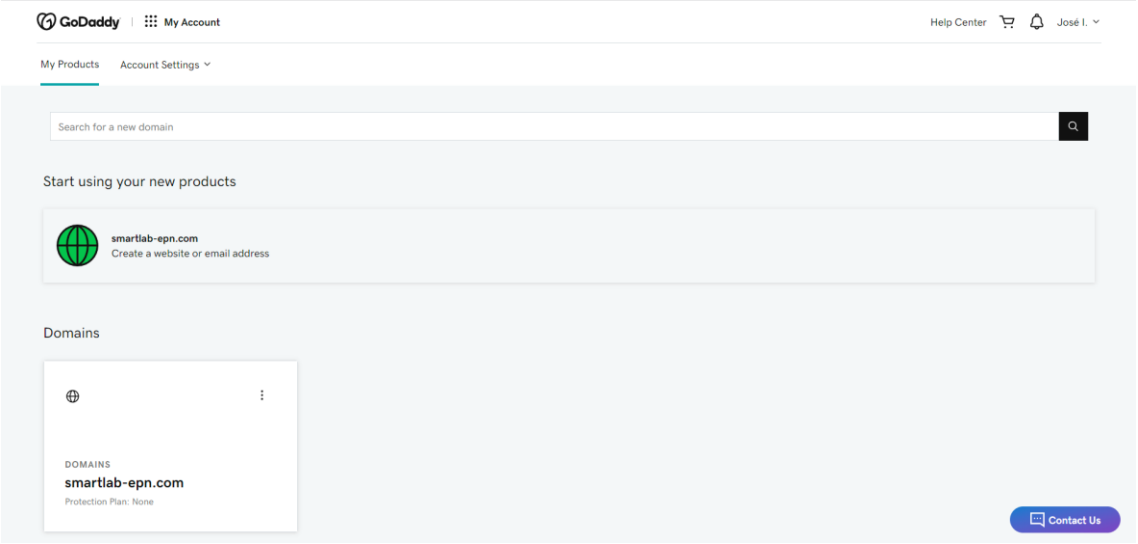


Ilustración 45 Compra del dominio smartlab-epn.com en el registrador de dominio GoDaddy

Establecer el récord DNS tipo A con la IP pública perteneciente al laboratorio Smartlab de la Escuela Politécnica Nacional, en la ilustración 46 se muestra el establecimiento del récord DNS tipo A con la IP pública perteneciente al laboratorio SmartLab de la EPN.

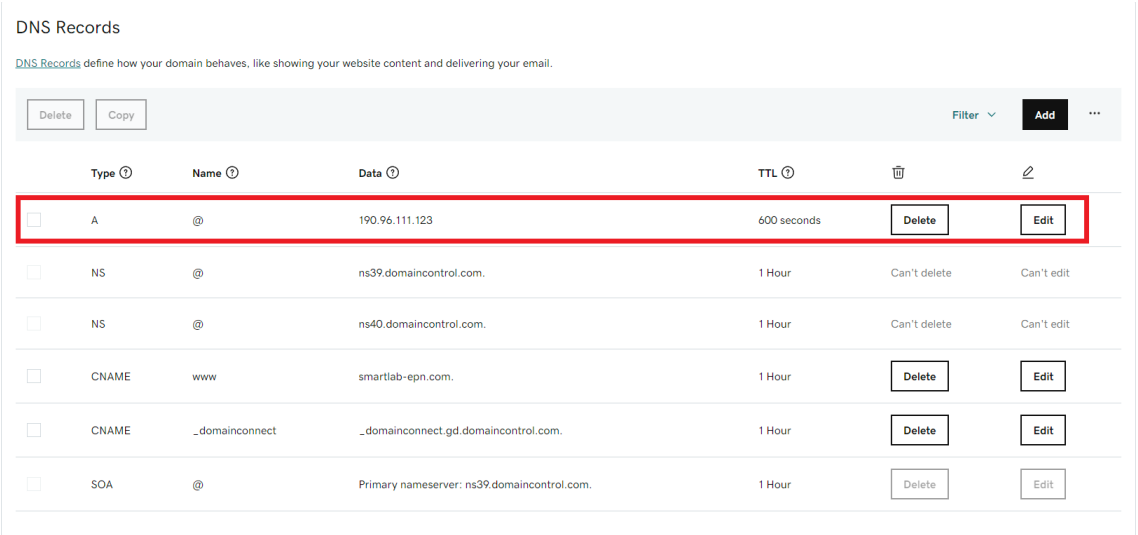


Ilustración 46 Establecimiento del récord DNS tipo A con la IP pública perteneciente al laboratorio Smartlab de la Escuela Politécnica Nacional.

El establecimiento del Nodo Gateway en la red DMZ se muestra en la ilustración 47, para reenviar todo el tráfico que proviene de internet al balanceador de carga y servidor web NGINX.

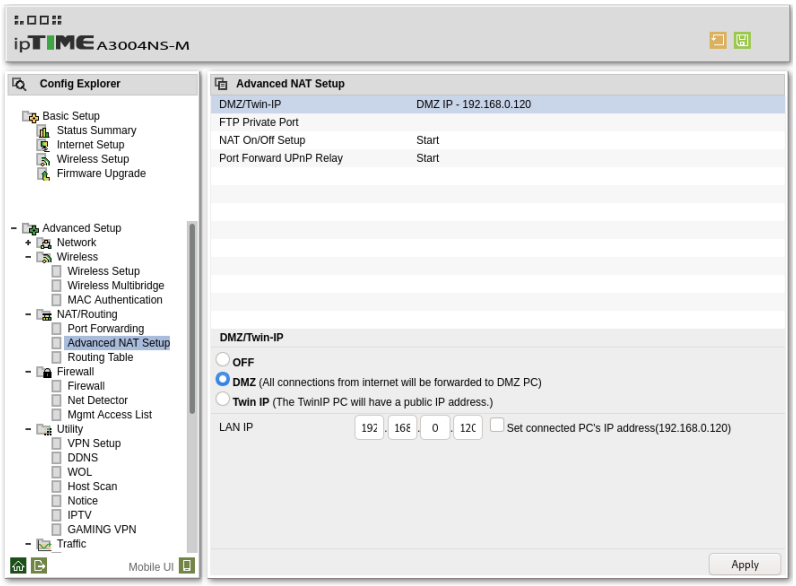


Ilustración 47 Establecimiento del Nodo Gateway en la red DMZ

Establecer el reenvío de los siguientes puertos (Port Forwarding) al balanceador de carga (Nodo Gateway):

- Puerto 80: Tráfico HTTP (Puerto 80).
- Puerto 443: Tráfico encriptado HTTPS (Puerto 443).
- Puerto 64002: Servicio de noVNC (Puerto 6080).
- Puerto 64001: Servicio SSH (Puerto 22). La ilustración 48 muestra el establecimiento del reenvío del Port Forwarding.

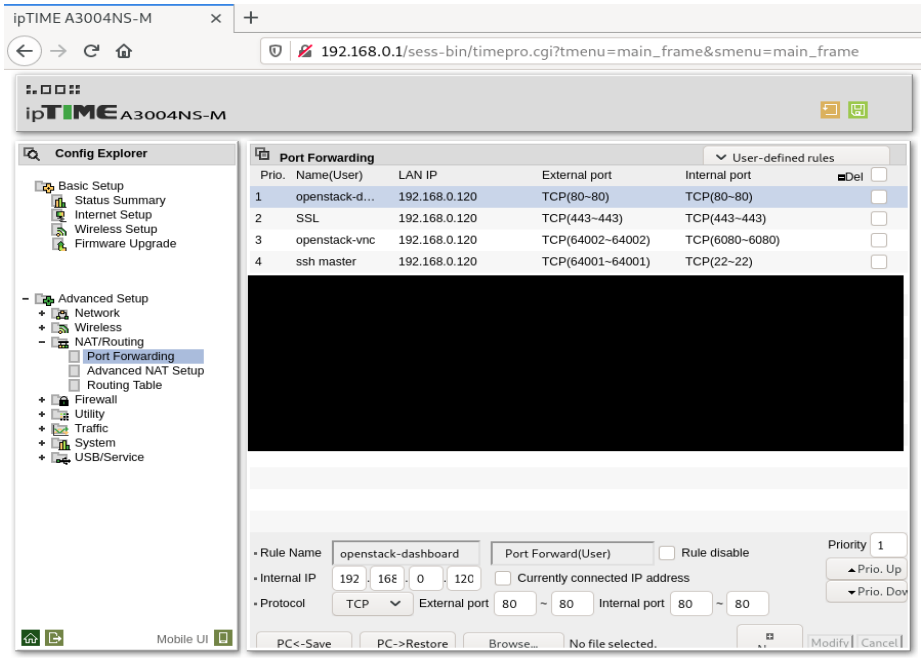


Ilustración 48 Establecer el reenvío de Port Forwarding

3.9.2. Proxy Reverso

Instalar el servidor web NGINX.

```
# apt install nginx
```

Comando 170 Instalación de Nginx

```
# nano /etc/nginx/sites-enabled/reverse-proxy.conf
```

Comando 171 Edición del archivo de configuración del Proxy Rerverso

Establecer la ruta de archivo donde se almacenarán los logs de acceso y error y reenviar el tráfico http hacia el nodo controlador.

```
server {
        access_log /var/log/nginx/reverse-access.log;
        error_log /var/log/nginx/reverse-error.log;
        server_name smartlab-epn.com;
        location / {
          proxy_pass http://192.168.0.201/;
        }
}
```

Comando 172 Configuración del Proxy reverso

Actualizar la lista de repositorios e instalar snapd.

```
# apt update && apt install snapd && snap install core && snap refresh core
```

Comando 173 Instalación de Snapd

Instalar certbot mediante snap y crear un enlace “suave” para agregar al PATH de ejecución.

```
# snap install --classic certbot && ln -s /snap/bin/certbot /usr/bin/certbot
```

Comando 174 Instalación de Certbot

Generar el certificado SSL para el dominio smartlab-epn.com y modificar las configuraciones actuales de nginx para redirigir todo el tráfico encryptado por el puerto 443.

```
# certbot –nginx
```

Comando 175 Ejecución de Certbot

3.10. Imágenes de sistemas operativos a virtualizar

Las imágenes de sistema operativo que estarán disponibles en el repositorio de imágenes del proyecto.

3.10.1. Debian GNU/Linux "bullseye"

La distribución estable y robusta que fue lanzada hace más de dos décadas (17 de junio de 1996). Soporta un gran número de paquetes y es la base para otras distribuciones como Ubuntu, SteamOS, Tails, entre otros.

3.10.2. Ubuntu 20.04 LTS (Focal Fossa)

Distribución con soporte a largo plazo LTS por sus siglas en inglés, está basada en Debian GNU / Linux. Tiene una comunidad robusta y brinda soporte empresarial.

3.11. Proyecto en OpenStack

El prototipo de Cloud computing aloja el proyecto llamado thesis-workspace. Este proyecto tiene el objetivo de brindar un espacio a los estudiantes y miembros del laboratorio Smartlab, en donde puedan implementar o probar proyectos que tengan en mente. Además, en el caso de que exista algún dispositivo de IOT o recurso conectado a la red LAN física conocida como red Proveedora, podrá ser utilizado de forma remota.

3.11.1. Restricciones de los recursos

Para permitir tener otros proyectos alojados en el prototipo se ha establecido cuotas a los recursos a nivel de servicio de cómputo y almacenamiento, dejando los valores predeterminados para las cuotas de red, en la tabla 5 y 6 se muestran las cuotas de red y los límites determinados para cada una de ellas.

Tabla 5 Cuota impuesta en el proyecto de los recursos de cómputo

Nombre de la Cuota	**Límite**
VCPUs	20
Archivo de contenido inyectado (Bytes)	10240
Longitud de la ruta del archivo inyectado	255
Archivos Inyectados	5
Par de llaves	100
Items de metadatos	128
RAM (Megabytes)	50000
Miembros del grupo de Servidores	10
Grupo de Servidores	10

Tabla 6 Cuota impuesta en el proyecto de los recursos de almacenamiento

Nombre de la Cuota	**Límite**
Volúmenes	10
Snapshots	5
Tamaño total de los volúmenes y Snapshots (Gigabytes)	400
Backups	10

3.11.2. Usuarios y Roles

Los usuarios pueden ser creados por un administrador de OpenStack. Cada usuario pertenece al proyecto thesis-workspace y puede tener los siguientes roles:

- admin: Tiene los permisos sobre recursos a nivel de sistema y proyecto.
- member: Tiene permisos sobre los propios recursos del usuario creado en el proyecto.
- reader: Tiene permisos de lectura sobre los recursos del proyecto, pero no puede realizar ningún cambio.

3.11.3. Topología de Red

La red self-service thesis-workspace-network del proyecto tiene la red privada clase A con notación CIDR 10.0.0.0/8. Esta red está conectada a un router llamado main-router virtual que tiene conexión con la red Proveedora. Las instancias tendrán la opción de reservar direcciones IP flotantes del pool reservado desde la dirección IP 192.168.0.216 hasta 192.168.0.254. En la ilustración 49 se muestra la topología de red del proyecto tesis-workspace.

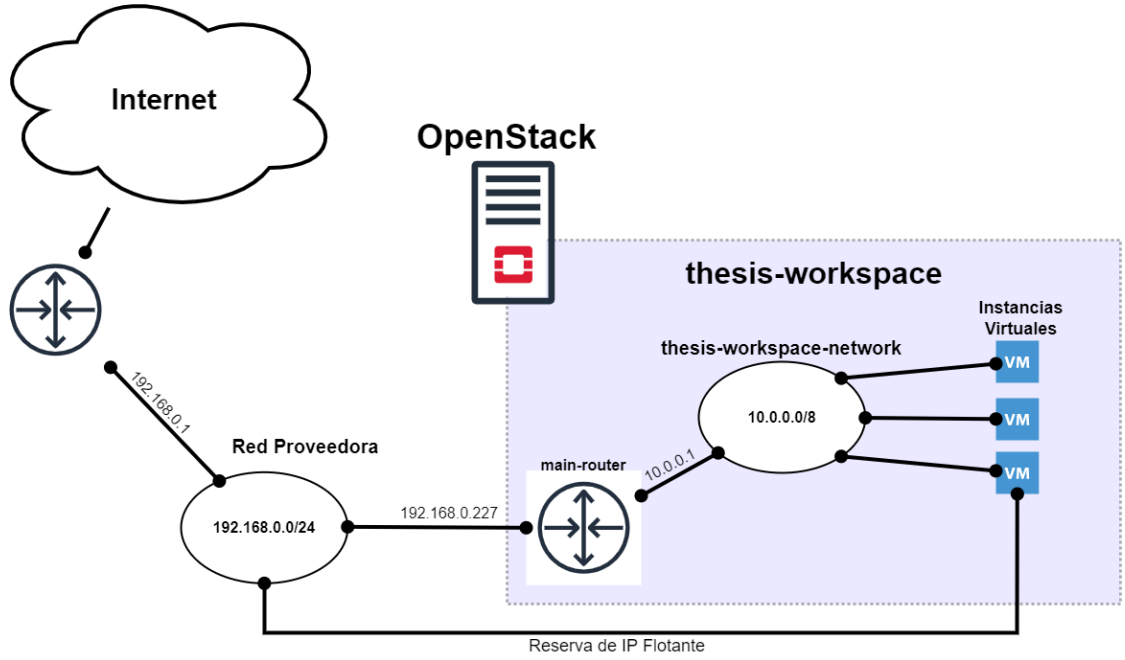


Ilustración 49 Topología de red del proyecto thesis-workspace

3.11.4. Flavors

El proyecto de thesis-workspace tiene 5 flavors que podrán ser utilizados por los usuarios del proyecto según sus necesidades. Las especificaciones de los flavors son descritas a continuación en la tabla 7:

Tabla 7 Flavors

Nombre	RAM (MB)	Disco E (GB)	vCPUs	Swap (MB)
m1.micro	1024	20	1	1024
m1.tiny	2048	40	1	2048
m1.small	2048	40	2	2048
m1.medium	4096	40	2	2048
m1.large	4096	80	4	2048

3.11.5. Implementación del Proyecto en OpenStack

3.11.5.1. *Proyecto y usuario*

Crear el proyecto thesis-workspace con su respectiva descripción.

```
# openstack project create \
  --description 'Environment for undergraduate students dissertation projects' \
  --domain default \
  thesis-workspace
```

Comando 176 Creación del proyecto thesis-workspace en OpenStack

Crear el usuario administrador del proyecto.

```
# openstack user create \
    --project thesis-workspace \
    --password-prompt \
    --email jose.garcia01@epn.edu.ec \
    --description "Undergraduate Student" \
    --or-show \
    tkhacker
```

Comando 177 Creación del usuario en OpenStack

Asignar el rol de administrador al usuario creado anteriormente.

```
# openstack role add \
    --user tkhacker \
    --project thesis-workpspace \
    admin
```

Comando 178 Asignación de rol administrador al usuario tkhacker en OpenStack

3.11.5.2. *Restricción de recursos*

Establecer las restricciones de los recursos de cómputo

```
# openstack quota set
    --cores 20 \
    --injected-file-size 10240 \
    --injected-files 10 \
    --instances 10 \
    --key-pairs 100 \
    --ram 50000 \
    --server-groups 10 \
    --server-group-members 10
```

Comando 179 Establecimiento de restricciones de recursos de cómputo en OpenStack

Establecer las restricciones de los recursos de almacenamiento

```
# openstack quota set
    --backups 10
    --gigabytes 400
    --snapshots 5
    --volumes 10
```

Comando 180 Establecimiento de restricciones de recursos de almacenamiento en OpenStack

3.11.5.3. *Redes*

Crear en OpenStack la red provider que reside en la LAN.

```
# openstack network create  --share --external \
  --provider-physical-network provider \
  --provider-network-type flat provider
```

Comando 181 Creación de la red proveedora en OpenStack

Crear en OpenStack la subred provider, se define el rango de IPs flotantes, servidores dns, el default Gateway y rango de la red LAN física.

```
# openstack subnet create --network provider \
  --allocation-pool start=192.168.0.216,end=192.168.0.254 \
  --dns-nameserver 8.8.8.8 --gateway 192.168.0.1 \
  --subnet-range 192.168.0.0/24 provider
```

Comando 182 Creación de la subred proveedora en OpenStack

Crear la red self-service thesis-workspace-network en el proyecto thesis-workspace.

```
# openstack network create thesis-workspace-network --project thesis-workspace
```

Comando 183 Creación de la red self-service thesis-workspace en OpenStack

Crear la subred thesis-workspace-subnet perteneciente a la red creada anteriormente con dirección de DNS 8.8.8.8, Gateway 10.0.0.1 y el rango de la red en formato CIDR 10.0.0.0/8.

```
# openstack subnet create \
  --network thesis-workspace-network \
  --dns-nameserver 8.8.8.8 --gateway 10.0.0.1 \
  --subnet-range 10.0.0.0/8 thesis-workspace-subnet \
  --project thesis-workspace
```

Comando 184 Creación de la subred thesis-workspace en OpenStack

Crear el router virtual que une a la red provider con la red self-service thesis-workspace network.

```
# openstack router create main-router
```

Comando 185 Creación del router virtual en OpenStack

Agregar la subred thesis-workspace-subnet al router para convertirlo en default Gateway.

```
# openstack router add subnet main-router thesis-workspace-subnet
```

Comando 186 Registro de la subred thesis-workspace en el router virtual en OpenStack

Agregar la subred provider al router y establecerlo como Gateway hacia Internet.

```
# openstack router set main-router --external-gateway provider
```

Comando 187 Configuración del gateway externo en OpenStack

3.11.5.4. *Grupos de seguridad de red*

Crear la regla para permitir el flujo de paquetes ICMP (ping) desde la red externa.

```
# openstack security group rule create \
  --proto icmp \
  2cebbbbd-bcdc-428a-9a90-51906202e7f6
```

Comando 188 Permiso de flujo de paquetes ICMP entre internet y OpenStack

Crear la regla para permitir el acceso remoto SSH desde la red externa.

```
# openstack security group rule create \
  --proto tcp \
  --dst-port 22 \
  2cebbbbd-bcdc-428a-9a90-51906202e7f6
```

Comando 189 Permiso de flujo de tráfico SSH

3.11.5.5. *Flavors*

Crear el flavor m1.micro en el proyecto con sus respectivas especificaciones

```
# openstack flavor create m1.micro \
    --ram 1024 --disk 20 --vcpus 1 --swap 1024 \
    --private --project thesis-workspace
```

Comando 190 Creación del flavor m1.micro en OpenStack

Crear el flavor m1.tiny en el proyecto con sus respectivas especificaciones

```
# openstack flavor create m1.tiny \
    --ram 2048 --disk 40 --vcpus 1 --swap 2048 \
    --private --project thesis-workspace
```

Comando 191 Creación del flavor m1.tiny en OpenStack

Crear el flavor m1.small en el proyecto con sus respectivas especificaciones

```
# openstack flavor create m1.small \
    --ram 2048 --disk 40 --vcpus 2  --swap 2048 \
    --private --project thesis-workspace
```

Comando 192 Creación del flavor m1.small en OpenStack

Crear el flavor m1.medium en el proyecto con sus respectivas especificaciones

```
# openstack flavor create m1.medium \
    --ram 4096 --disk 40 --vcpus 2  --swap 2048 \
    --private --project thesis-workspace
```

Comando 193 Creación del flavor m1.medium en OpenStack

Crear el flavor m1.large en el proyecto con sus respectivas especificaciones

```
# openstack flavor create m1.large \
    --ram 4096 --disk 80 --vcpus 4  --swap 2048 \
    --private  --project thesis-workspace
```

Comando 194 Creación del flavor m1.large en OpenStack

3.11.5.6. *Imágenes de sistemas operativos*

Cargar la imagen descargada del repositorio oficial de Ubuntu al servidor Glance con nombre de focal-server

```
# openstack image create --public \
  --disk-format qcow2 --container-format bare \
  --file /home/controller/images/focal-server-cloudimg-amd64.img focal-server
```

Comando 195 Carga de la imagen de Sistema Ubuntu a OpenStack

Cargar la imagen descargada del repositorio oficial de Debian al servidor Glance con nombre de debian-10

```
# openstack image create --public \
  --disk-format qcow2 --container-format bare \
  --file /home/controller/images/debian-10.11.2-20211129-openstack-amd64.qcow2
debian-10
```

Comando 196 Carga de la imagen de Sistema Debian a OpenStack

3.12. Metodología de Desarrollo

El proyecto consta de 7 etapas, en donde cada etapa está relacionada con cada actividad del cronograma de actividades, las cuales se describen en la ilustración 50. Las actividades son secuenciales, lo que significa que para avanzar a la siguiente actividad se debe completar primero la actividad activa.

	Mes 1				Mes 2				Mes 3				Mes 4				Mes 5				Mes 6			
	Semana				Semana				Semana				Semana				Semana				Semana			
Actividades	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Definición de requisitos	10	20																						
Diseño de la arquitectura			10	20																				
Instalación de los componentes de OpenStack					10	20	20	20																
Configuración de los componentes de OpenStack									10	20	20	20												
Integración de los componentes de OpenStack													10	10	20	20								
Configuración del prototipo de cloud computing																	10	20	10	20				
Adaptación y ejecución de TAM al prototipo de cloud computing																					20	20	20	40
TOTAL, DE HORAS:	420																							

Ilustración 50 Cronograma de actividades para el desarrollo del proyecto

El desarrollo del proyecto se gestionó mediante la Pizarra Kanban implementada con la herramienta Trello. Gracias a la flexibilidad de diseño la pizarra consta de 6 columnas descritas a continuación:

- Backlog: Aloja todas las tareas que pertenece a una etapa. No está permitido agregar tareas de otras etapas hasta que todas las tareas de la etapa actual estén en la columna done.
- Ready: Recibe un grupo de tareas de la columna Backlog que forman un entregable. Las tareas son ordenadas por prioridad y es posible crear tareas adicionales que no se hayan previsto anteriormente.
- TO-DO: Acepta las tareas que están listas para realizarse.
- Doing: Se alojan las tareas que se estén realizando en ese momento. Se ha definido un WIP de 2 tareas, por lo que esta columna tiene el límite de alojar máximo 2 tareas al mismo tiempo.
- Done: Recibe todas las tareas de la columna Doing que se han completado.
- Hold: Por la falta de experiencia en la planificación de proyectos y falta de conocimientos sobre el tema, la planificación de tareas no es perfecta, las tareas que se crean en el Backlog son muy generales y no representan el trabajo real que debe realizarse. Cuando se presentan dificultades que demandan la creación de tareas adicionales, la tarea es transferida a esta columna y las nuevas tareas siguen el flujo normal desde la columna TO-DO.

En el caso de que estas tareas adicionales, a su vez tengan más tareas adicionales se repite el proceso y se ordenan las tareas en la columna Hold, en la ilustración 51 se muestra el flujo de tareas de Kanban, y en la ilustración 52 y 53 se muestra la pizarra de tareas con las tareas pendientes y finalizadas.

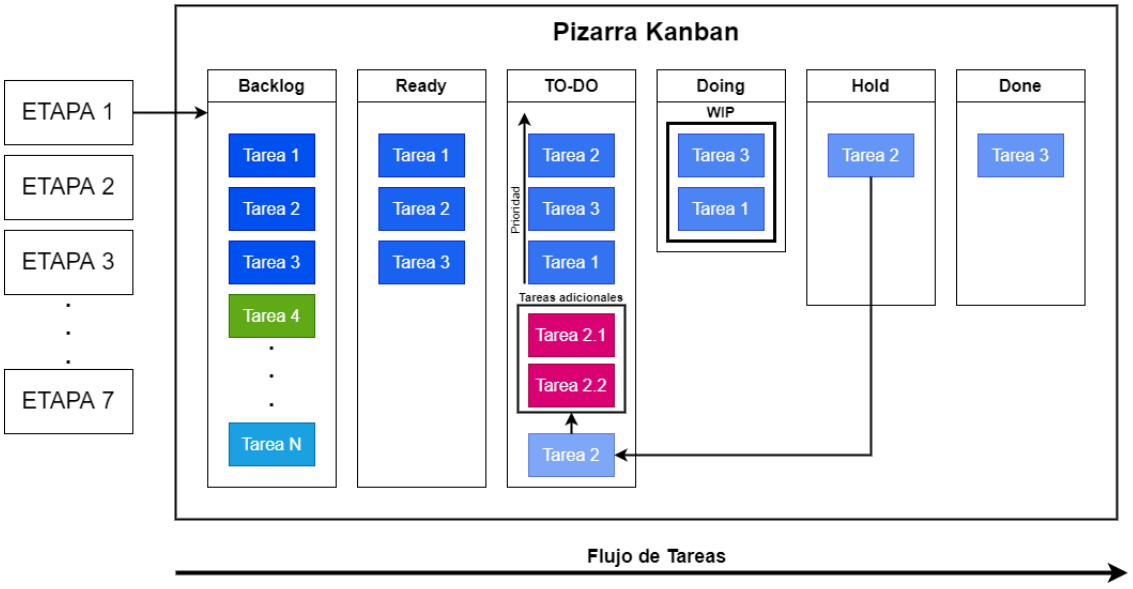


Ilustración 51 Flujo de Tareas en la Pizarra Kanban

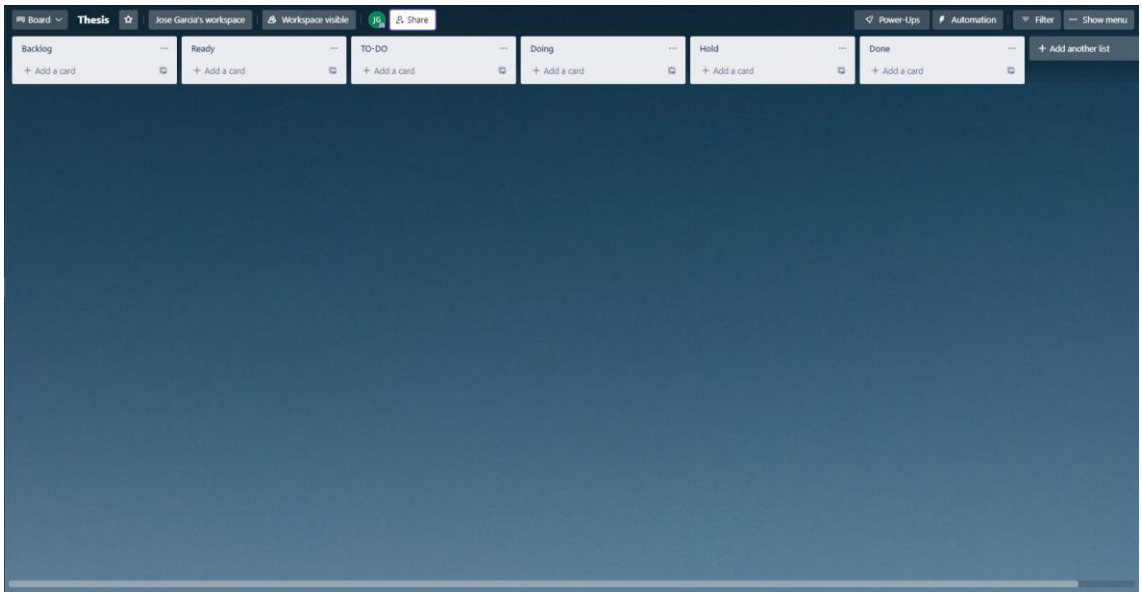


Ilustración 52 Pizarra Kanban utilizada para desarrollar el proyecto

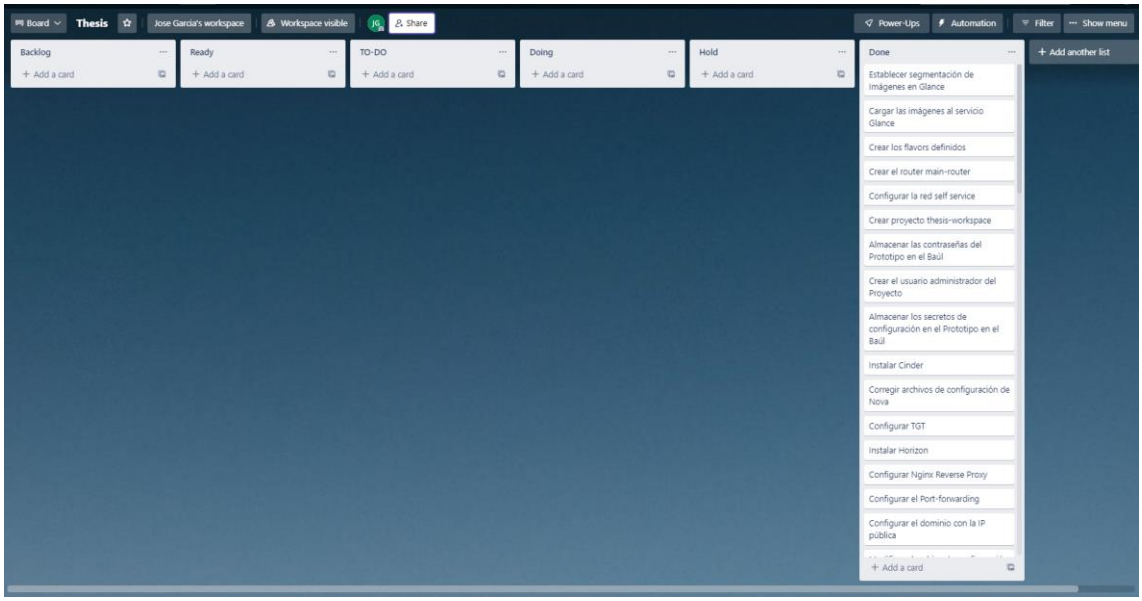


Ilustración 53 Pizarra Kanban con todas las tareas involucradas finalizadas

4. CAPÍTULO CUARTO: RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

4.1. Resultados

4.1.1. Desarrollo del caso de prueba

El administrador del proyecto debe crear 3 máquinas virtuales que forman parte de un clúster utilizando el cliente Web Horizon, en las tablas 8, 9 y 10 se muestran las especificaciones de las máquinas virtuales main node, secondary node 1 y secondary node 2 respectivamente:

Tabla 8 Máquina virtual main-node

Máquina virtual main-node	
vCPUs	2
RAM (GB)	8
Almacenamiento (GB)	100
Swap (GB)	2
Sistema Operativo	Ubuntu Focal Fossa 20.04 LTS
Número de Interfaces de Red	2

Tabla 9 Máquina virtual secondary-node-1

Máquina virtual secondary-node-1	
vCPUs	4
RAM (GB)	16
Almacenamiento (GB)	120
Swap (GB)	2
Sistema Operativo	Ubuntu Focal Fossa 20.04 LTS
Número de Interfaces de Red	1

Tabla 10 Máquina virtual secondary-node-2

Máquina virtual secondary-node-2	
vCPUs	4
RAM (GB)	16
Almacenamiento (GB)	120
Swap (GB)	2
Sistema Operativo	Ubuntu Focal Fossa 20.04 LTS
Número de Interfaces de Red	1

Nota: Ya que no existen Flavors que soporten las especificaciones enlistadas anteriormente, el administrador del proyecto creará los Flavors apropiados con anterioridad.

Las 3 máquinas virtuales pertenecerán a la red self-service del proyecto y la máquina virtual main-node tendrá una interfaz de red adicional en la red proveedora. Las 2 máquinas restantes tendrán la opción de reservar una IP flotante más adelante. En la ilustración 54 se muestra el diagrama de red del escenario de pruebas.

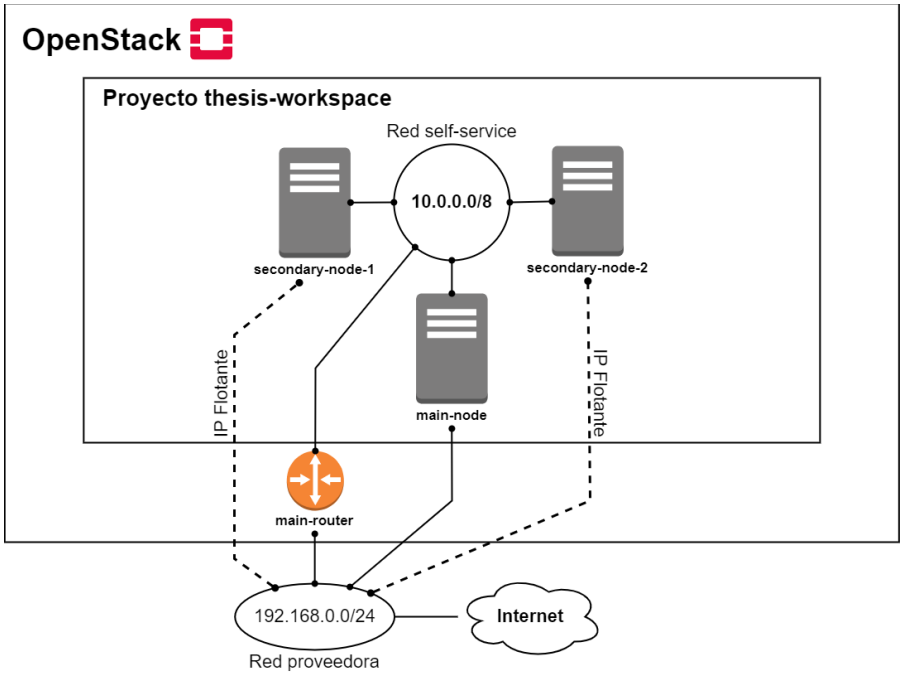


Ilustración 54 Diagrama de red del escenario de pruebas

4.1.2. Proceso de Medición

El proceso de medición permitirá saber cómo se desenvuelven las máquinas virtuales en el prototipo de cloud computing. Se realizará la medición de las 3 máquinas virtuales del caso de prueba con los parámetros de medición de conectividad, estabilidad y recursos.

4.1.2.1. **Conectividad**

Se medirá la latencia que existe entre las máquinas virtuales dentro de la red self-service y el tiempo que toma realizar la conexión SSH desde internet hasta la máquina virtual main-node.

4.1.2.1.1. *Envío de mensajes ICMP (ping)*

Permite obtener el tiempo que toma en recibir una respuesta de una instancia a otra. Se enviará 100 mensajes ICMP desde cada máquina virtual con el comando ping y se obtendrá el resultado promedio por 2 escenarios:

- Entre máquinas virtuales (red self-service)
- Desde cada máquina virtual hacia Internet (servidor DNS de Google)

4.1.2.1.2. *Conexión SSH*

Se medirá el tiempo promedio que toma establecer una conexión SSH mediante la herramienta OpenSSH. El cliente accederá desde Internet hacia la máquina virtual main-node, se autenticará utilizando su propia llave privada y ejecutará el comando true. La dirección IP de la interfaz de red estará expuesta a internet por medio de Port-Forwarding. El usuario establecerá 10 conexiones SSH y se obtendrá el resultado promedio. Además, se utilizará el dominio como dirección del host.

4.1.2.2. Estabilidad

4.1.2.2.1. *Ejecución de un programa*

Para poder medir el tiempo de actividad (up-time), cada máquina virtual ejecutará un script por 24 horas. El script consiste en un lazo que crea un archivo de texto y añade líneas de texto con la información del tiempo actual por cada segundo transcurrido.

4.1.2.3. Recursos

Los recursos de la máquina virtual y del nodo compute que se medirán son memoria y CPU. Para poder realizar esta prueba se ejecutará en OpenStack únicamente las máquinas virtuales del caso de prueba.

4.1.2.3.1. *Visualización de recursos*

Se utilizará la herramienta htop para visualizar los recursos mencionados anteriormente.

4.1.3. Resultados de cada prueba

En la ilustración 55 se muestran los resultados del proyecto, con la lista de las máquinas virtuales del caso de pruebas ejecutándose.

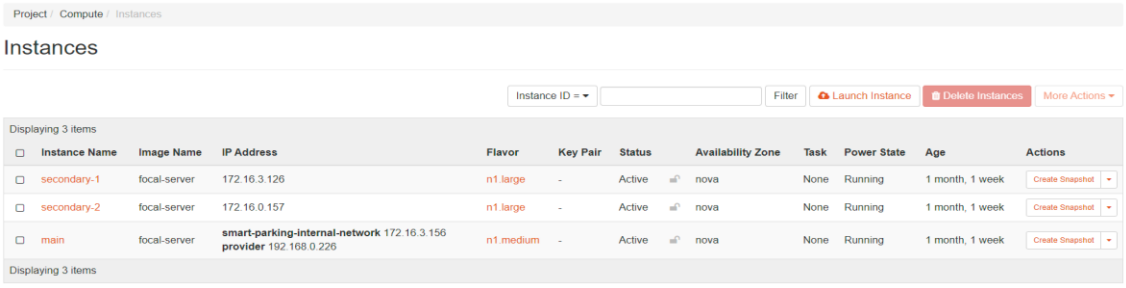


Ilustración 55 Lista de máquinas virtuales del caso de pruebas ejecutándose.

4.1.3.1. Conectividad

4.1.3.1.1. *Envío de mensajes ICMP (ping)*

4.1.3.1.1.1. Máquina virtual main

4.1.3.1.1.1.1. Entre máquinas virtuales (red self-service):

En la tabla 11 se detallan los resultados del tiempo promedio de envío de mensajes ICMP a las otras dos máquinas virtuales, mientras que en la ilustración 56 se detalla el comando para enviar 100 pings desde la máquina virtual main a las otras dos máquinas de manera simultánea y guardar los resultados en un archivo de texto por cada host; además, en la ilustración 57 se muestra la ejecución de 2 comandos para calcular el tiempo promedio de los resultados de cada archivo generado en la ilustración anterior.

Tabla 11 Resultados del tiempo promedio de envío de mensajes ICMP a las otras dos máquinas virtuales

Máquina virtual	Tiempo promedio en microsegundos
Secondary-1 (172.16.3.126)	0.58005
Secondary-2 (172.16.0.157)	0.58831

```
ubuntu@main:~$ pings=100; host=172.16.3.126; date; ping -c $pings $host > $host.txt & host=172.16.0.157; pin
g -c $pings $host > $host.txt &
Fri Apr 22 05:16:01 UTC 2022
[1] 1529
[2] 1530
ubuntu@main:~$ fg
-bash: fg: job has terminated
[1]-  Done                    ping -c $pings $host > $host.txt
[2]+  Done                    ping -c $pings $host > $host.txt
ubuntu@main:~$
```

Ilustración 56 Comando para enviar 100 pings desde la máquina virtual main a las otras dos máquinas de manera simultánea y guardar los resultados en un archivo de texto por cada host.

```
ubuntu@main:~$ echo ""; sum=$(tail -n +2 172.16.0.157.txt | head -n $pings | awk '{print $7}' | cut -d "=" -
f2 | paste -sd+ | bc); printf "Tiempo promedio (ms): "; echo "$sum / $pings" | bc -l;

Tiempo promedio (ms): .58005000000000000000
ubuntu@main:~$ echo ""; sum=$(tail -n +2 172.16.3.126.txt | head -n $pings | awk '{print $7}' | cut -d "=" -
f2 | paste -sd+ | bc); printf "Tiempo promedio (ms): "; echo "$sum / $pings" | bc -l;

Tiempo promedio (ms): .58831000000000000000
ubuntu@main:~$
```

Ilustración 57 Ejecución de 2 comandos para calcular el tiempo promedio de los resultados de cada archivo generado en la ilustración anterior.

4.1.3.1.1.1.2. Envío de mensajes ICMP hacia el servidor DNS de Google

La tabla 12 resume los resultados del tiempo promedio de envío de 100 mensajes de ICMP hacia el servidor DNS de Google, mientras que la ilustración 58 indica el comando para enviar 100 pings hacia el servidor DNS de Google y calcular el valor promedio.

Tabla 12 Resultados del tiempo promedio de envío de 100 mensajes ICMP hacia el servidor DNS de Google.

Servidor DNS de Google	Tiempo promedio en microsegundos
8.8.8.8	65.35

```
ubuntu@main:~$ echo "" && date; pings=100; sum=$(ping -c $pings 8.8.8.8 | tail -n +2 | head -n $pings | awk
'{print $7}' | cut -d "=" -f2 | paste -sd+ | bc); printf "Tiempo promedio (ms): "; echo "$sum / $pings" | bc
 -l; date

Fri Apr 22 04:43:33 UTC 2022
Tiempo promedio (ms): 65.35000000000000000000
Fri Apr 22 04:45:12 UTC 2022
ubuntu@main:~$
```

Ilustración 58 Comando para enviar 100 pings hacia el servidor DNS de Google y calcular el valor promedio.

4.1.3.1.1.2. Máquina virtual secondary-1

4.1.3.1.1.2.1. Entre máquinas virtuales (red self-service)

La tabla 13 indica los resultados del tiempo promedio de envío de mensajes ICMP a las otras dos máquinas virtuales, mientras que la ilustración 59 muestra el comando para enviar 100 pings desde la máquina virtual secondary-1 a las otras dos máquinas de forma simultánea y guardar los resultados en un archivo de texto; la ilustración 60 muestra la ejecución de dos comandos para calcular el tiempo promedio de los resultados de cada archivo generado en la ilustración 59.

Tabla 13 Resultados del tiempo promedio de envío de mensajes ICMP a las otras dos máquinas virtuales

Máquina virtual	**Tiempo promedio en microsegundos**
main (172.16.3.156)	0.58203
Secondary-2 (172.16.0.157)	0.76180

```
ubuntu@secondary-1:~$ pings=100; host=172.16.0.157; date; ping -c $pings $host > $host.txt & host=172.16.3.156; pin
g -c $pings $host > $host.txt &
Fri Apr 22 05:16:03 UTC 2022
[1] 1607
[2] 1608
ubuntu@secondary-1:~$ fg
-bash: fg: job has terminated
[1]-  Done                    ping -c $pings $host > $host.txt
[2]+  Done                    ping -c $pings $host > $host.txt
ubuntu@secondary-1:~$
```

Ilustración 59 Comando para enviar 100 pings desde la máquina virtual secondary-1 a las otras dos máquinas de manera simultánea y guardar los resultados en un archivo de texto por cada host.

```
ubuntu@secondary-1:~$ echo ""; sum=$(tail -n +2 172.16.0.157.txt | head -n $pings | awk '{print $7}' | cut -d "=" -
f2 | paste -sd+ | bc); printf "Tiempo promedio (ms): "; echo "$sum / $pings" | bc -l;

Tiempo promedio (ms): .58203000000000000000
ubuntu@secondary-1:~$ echo ""; sum=$(tail -n +2 172.16.3.156.txt | head -n $pings | awk '{print $7}' | cut -d "=" -
f2 | paste -sd+ | bc); printf "Tiempo promedio (ms): "; echo "$sum / $pings" | bc -l;

Tiempo promedio (ms): .76180000000000000000
ubuntu@secondary-1:~$
```

Ilustración 60 Ejecución de 2 comandos para calcular el tiempo promedio de los resultados de cada archivo generado en la figura anterior.

4.1.3.1.1.2.2. Envío de mensajes ICMP hacia el servidor DNS de Google

La tabla 14 indica los resultados del tiempo promedio de envío de 100 mensajes ICMP hacia el servidor DNS de Google, mientras que la ilustración 61 muestra el comando para enviar 100 pings hacia el servidor DNS de Google y calcular el valor promedio.

Tabla 14 Resultados del tiempo promedio de envío de 100 mensajes ICMP hacia el servidor DNS de Google.

Servidor DNS de Google	**Tiempo promedio en microsegundos**
8.8.8.8	65.726

```
ubuntu@secondary-1:~$ echo "" && date; pings=100; sum=$(ping -c $pings 8.8.8.8 | tail -n +2 | head -n $pings | awk
'{print $7}' | cut -d "=" -f2 | paste -sd+ | bc); printf "Tiempo promedio (ms): "; echo "$sum / $pings" | bc -l; da
te

Fri Apr 22 04:43:36 UTC 2022
Tiempo promedio (ms): 65.72600000000000000000
Fri Apr 22 04:45:15 UTC 2022
ubuntu@secondary-1:~$
```

Ilustración 61 Comando para enviar 100 pings hacia el servidor DNS de Google y calcular el valor promedio.

4.1.3.1.1.3. Máquina virtual secondary-2

4.1.3.1.1.3.1. Entre máquinas virtuales (red self-service)

La tabla 15 indica los resultados del tiempo promedio de envío de mensajes ICMP a las otras dos máquinas virtuales mientras que la ilustración 62 muestra el comando para enviar 100 pings desde la máquina virtual secondary-2 a las otras 2 máquinas de manera simultánea y guardar los resultados en un archivo de texto por cada host; y la ilustración 63 muestra la ejecución de dos comandos para calcular el tiempo promedio de los resultados de cada archivo generado en la ilustración 62.

Tabla 15 Resultados del tiempo promedio de envío de mensajes ICMP a las otras dos máquinas virtuales.

Máquina virtual	**Tiempo promedio en microsegundos**
main (172.16.3.156)	0.5981
Secondary-1 (172.16.3.126)	0.7233

```
ubuntu@secondary-2:~$ pings=100; host=172.16.3.126; date; ping -c $pings $host > $host.txt & host=172.16.3.156; pin
g -c $pings $host > $host.txt &
Fri Apr 22 05:15:59 UTC 2022
[1] 1701
[2] 1702
ubuntu@secondary-2:~$ fg
-bash: fg: job has terminated
[1]-  Done                    ping -c $pings $host > $host.txt
[2]+  Done                    ping -c $pings $host > $host.txt
ubuntu@secondary-2:~$
```

Ilustración 62 Comando para enviar 100 pings desde la máquina virtual secondary-2 a las otras dos máquinas de manera simultánea y guardar los resultados en un archivo de texto por cada host.

```
ubuntu@secondary-2:~$ echo ""; sum=$(tail -n +2 172.16.3.126.txt | head -n $pings | awk '{print $7}' | cut -d "=" -
f2 | paste -sd+ | bc); printf "Tiempo promedio (ms): "; echo "$sum / $pings" | bc -l;

Tiempo promedio (ms): .598100000000000000
ubuntu@secondary-2:~$ echo ""; sum=$(tail -n +2 172.16.3.156.txt | head -n $pings | awk '{print $7}' | cut -d "=" -
f2 | paste -sd+ | bc); printf "Tiempo promedio (ms): "; echo "$sum / $pings" | bc -l;

Tiempo promedio (ms): .723320000000000000
ubuntu@secondary-2:~$
```

Ilustración 63 Ejecución de 2 comandos para calcular el tiempo promedio de los resultados de cada archivo generado en la figura anterior.

4.1.3.1.1.3.2. Envío de mensajes ICMP hacia el servidor DNS de Google

La tabla 16 indica los resultados del tiempo promedio del envío de 100 mensajes ICMP hacia el servidor DNS de Google, mientras que la ilustración 64 muestra el comando para enviar 100 pings hacia el servidor DNS de Google y calcular el valor promedio.

Tabla 16 Resultados del tiempo promedio de envío de 100 mensajes ICMP hacia el servidor DNS de Google.

Servidor DNS de Google	**Tiempo promedio en microsegundos**
8.8.8.8	65.747

```
ubuntu@secondary-2:~$ echo "" && date; pings=100; sum=$(ping -c $pings 8.8.8.8 | tail -n +2 | head -n $pings | awk
'{print $7}' | cut -d "=" -f2 | paste -sd+ | bc); printf "Tiempo promedio (ms): "; echo "$sum / $pings" | bc -l; da
te

Fri Apr 22 04:43:35 UTC 2022
Tiempo promedio (ms): 65.747000000000000000
Fri Apr 22 04:45:14 UTC 2022
ubuntu@secondary-2:~$
```

Ilustración 64 Comando para enviar 100 pings hacia el servidor DNS de Google y calcular el valor promedio.

4.1.3.1.2. *Conexión SSH*

La tabla 17 indica el tiempo en segundos de cada conexión SSH establecida en la máquina virtual main y su valor promedio, mientras que la ilustración 65 muestra el comando para realizar 10 conexiones SSH secuenciales y obtener el tiempo en segundos que toma establecer una conexión SSH y ejecutar el comando true.

Tabla 17 Tiempo en segundos de cada conexión SSH establecida en la máquina virtual main y su valor promedio.

Tiempo en segundos	
Intento 1	2.582
Intento 2	1.906
Intento 3	1.800
Intento 4	1.840
Intento 5	1.797
Intento 6	1.823
Intento 7	1.979
Intento 8	1.790
Intento 9	1.903
Intento 10	1.865
Valor del tiempo promedio:	1.9285

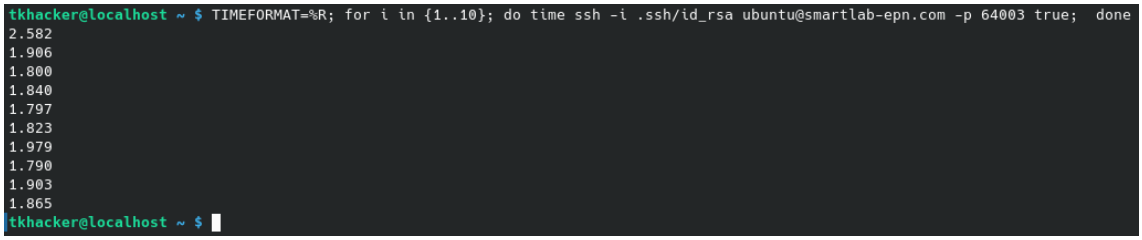


Ilustración 65 Comando para realizar 10 conexiones SSH secuenciales y obtener el tiempo en segundos que toma establecer una conexión SSH y ejecutar el comando true.

4.1.3.2. Estabilidad

4.1.3.2.1. *Ejecución de un programa*

4.1.3.2.1.1. Máquina virtual main

La ejecución del script generó un archivo de 86400 líneas (1 línea cada segundo) que equivale a alrededor de 24 horas. El primer Timestamp es del viernes 15 de abril del año 2022 a las 03:54:18 UTC y el último Timpestamp es del día sábado 16 de abril del año 2022 a las 04:01:13, por lo que al script le tomó ejecutarse 6 minutos con 55

segundos más de lo esperado. Considerando la iteración de cada sentencia y el tiempo de ejecución del comando date, al script le tomó 0.477% más del tiempo esperado; en la ilustración 66 se muestra el comando para leer las primeras y últimas 4 líneas del archivo generado (output.txt) por el script y contar su número de líneas, mientras que la ilustración 67 muestra el timestamp de la última modificación del archivo output.txt.

```
ubuntu@main:~$ head -n 4 output.txt && echo -e "\t.\n\t.  Output Omitido\n\t." && tail -n 4 output.txt
Fri Apr 15 03:54:18 UTC 2022
Fri Apr 15 03:54:19 UTC 2022
Fri Apr 15 03:54:20 UTC 2022
Fri Apr 15 03:54:21 UTC 2022
        .
        .  Output Omitido
        .
Sat Apr 16 04:01:10 UTC 2022
Sat Apr 16 04:01:11 UTC 2022
Sat Apr 16 04:01:12 UTC 2022
Sat Apr 16 04:01:13 UTC 2022
ubuntu@main:~$ wc -l output.txt
86400 output.txt
ubuntu@main:~$
```

Ilustración 66 Comando para leer las primeras y últimas 4 líneas del archivo generado (output.txt) por el script, y contar el número de líneas de este.

```
ubuntu@main:~$ ls -la
total 2496
drwxr-xr-x 6 ubuntu ubuntu    4096 Apr 15 03:54 .
drwxr-xr-x 4 root   root      4096 Mar 29 23:21 ..
-rw------- 1 ubuntu ubuntu     983 Apr 15 03:39 .bash_history
-rw-r--r-- 1 ubuntu ubuntu     220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu    3771 Feb 25  2020 .bashrc
drwx------ 2 ubuntu ubuntu    4096 Mar 16 22:36 .cache
drwx------ 4 ubuntu ubuntu    4096 Apr 14 00:17 .config
drwxrwxr-x 3 ubuntu ubuntu    4096 Apr 15 03:52 .local
-rw-r--r-- 1 ubuntu ubuntu     807 Feb 25  2020 .profile
drwx------ 2 ubuntu ubuntu    4096 Mar 16 22:42 .ssh
-rw-r--r-- 1 ubuntu ubuntu       0 Mar 25 17:29 .sudo_as_admin_successful
-rw-rw-r-- 1 ubuntu ubuntu 2505600 Apr 16 04:01 output.txt
-rwxrwxr-x 1 ubuntu ubuntu      77 Apr 15 03:53 script.sh
ubuntu@main:~$
```

Ilustración 67 Timestamp de última modificación del archivo output.txt

4.1.3.2.1.2. Máquina virtual secondary-1

La ejecución del script generó un archivo de 86400 líneas (1 línea cada segundo) que equivale a aproximadamente a 24 horas. El primer Timestamp es del viernes 15 de abril del año 2022 a las 03:54:25 UTC y el último Timpestamp es del sábado 16 de abril del año 2022 a las 04:01:38, por lo que al script le tomó ejecutarse 7 minutos con 13 segundos más de lo esperado. Considerando la iteración de cada sentencia y el tiempo de ejecución del comando date, al script le tomó 0.501% más del tiempo esperado; en la ilustración 68 se ve el comando para leer las primeras y últimas 4 líneas del archivo generado (output.txt) por el script y contar su número de líneas, mientras que en la ilustración 69 se ve el timestamp de última modificación del archivo.

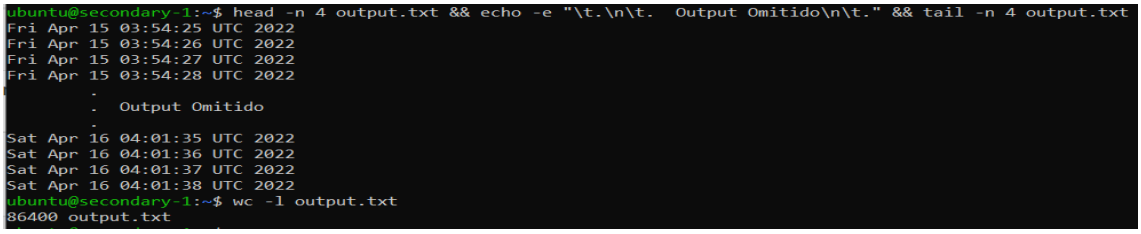


Ilustración 68 Comando para leer las primeras y últimas 4 líneas del archivo generado (output.txt) por el script, y contar el número de líneas de este.

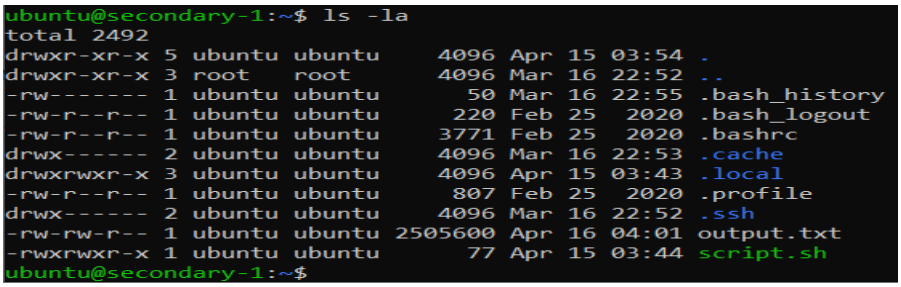


Ilustración 69 Timestamp de última modificación del archivo output.txt

4.1.3.2.1.3. Máquina virtual secondary-2

La ejecución del script generó un archivo de 86400 líneas (1 línea cada segundo) que equivale a alrededor de 24 horas. El primer Timestamp es del viernes 15 de abril del año 2022 a las 03:54:31 UTC y el último Timpestamp es del día sábado 16 de abril del año 2022 a las 04:01:41, por lo que al script le tomó ejecutarse 7 minutos con 10 segundos más de lo esperado. Considerando la iteración de cada sentencia y el tiempo de ejecución del comando date, al script le tomó 0.498% más del tiempo esperado; en la ilustración 70 se muestra el comando para leer las primeras y últimas 4 líneas del archivo generado (output.txt) por el script y contar su número de líneas, mientras que en la ilustración 71 se muestra el timestamp de última modificación del archivo output.txtRecursos.

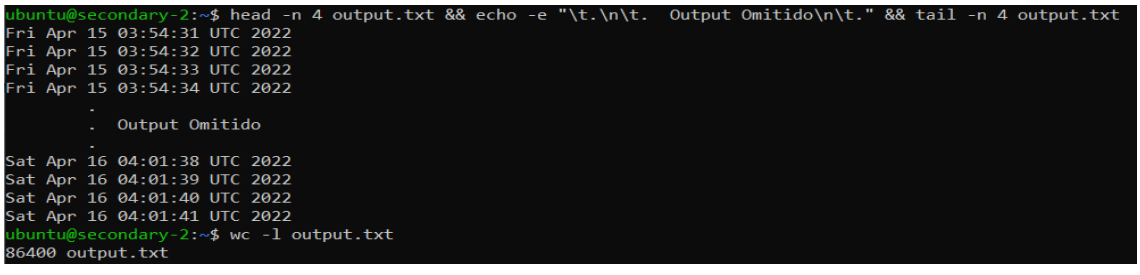


Ilustración 70 Comando para leer las primeras y últimas 4 líneas del archivo generado (output.txt) por el script, y contar el número de líneas de este.

```
ubuntu@secondary-2:~$ ls -la
total 2492
drwxr-xr-x 5 ubuntu ubuntu    4096 Apr 15 03:54 .
drwxr-xr-x 3 root   root      4096 Mar 16 22:41 ..
-rw------- 1 ubuntu ubuntu     133 Mar 16 22:52 .bash_history
-rw-r--r-- 1 ubuntu ubuntu     220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu    3771 Feb 25  2020 .bashrc
drwx------ 2 ubuntu ubuntu    4096 Mar 16 22:42 .cache
drwxrwxr-x 3 ubuntu ubuntu    4096 Apr 15 03:46 .local
-rw-r--r-- 1 ubuntu ubuntu     807 Feb 25  2020 .profile
drwx------ 2 ubuntu ubuntu    4096 Mar 16 22:41 .ssh
-rw-rw-r-- 1 ubuntu ubuntu 2505600 Apr 16 04:01 output.txt
-rwxrwxr-x 1 ubuntu ubuntu      77 Apr 15 03:47 script.sh
ubuntu@secondary-2:~$
```

Ilustración 71 Timestamp de última modificación del archivo output.txtRecursos

4.1.3.3. *Visualización de recursos*

En la ilustración 72 se muestra la visualización del uso del CPU y memoria, además de la ejecución de las máquinas virtuales como procesos y threads en el nodo de cómputo, mientras que la ilustración 73, 74 y 75 muestran la visualización del uso de CPU, Memoria y procesos mediante el comando htop de la máquina virtual main, secondary-1 y secondary-2.

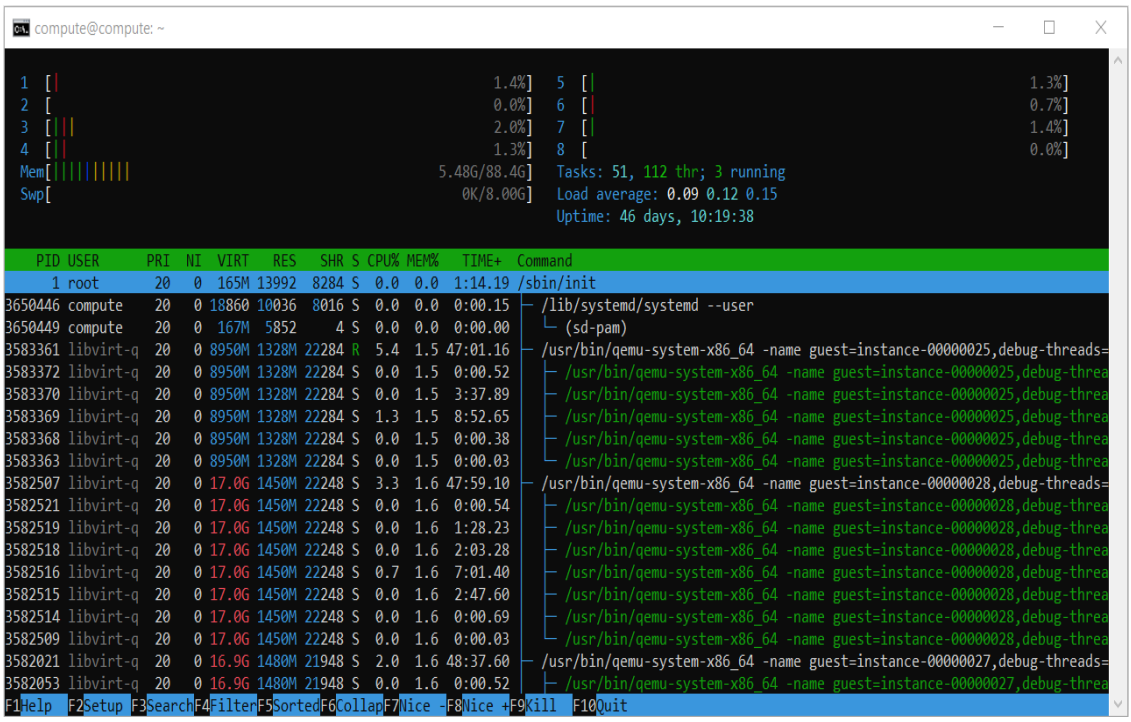


Ilustración 72 Visualización del uso de CPU y Memoria y la ejecución de las máquinas virtuales como procesos y threads en el Nodo de Cómputo

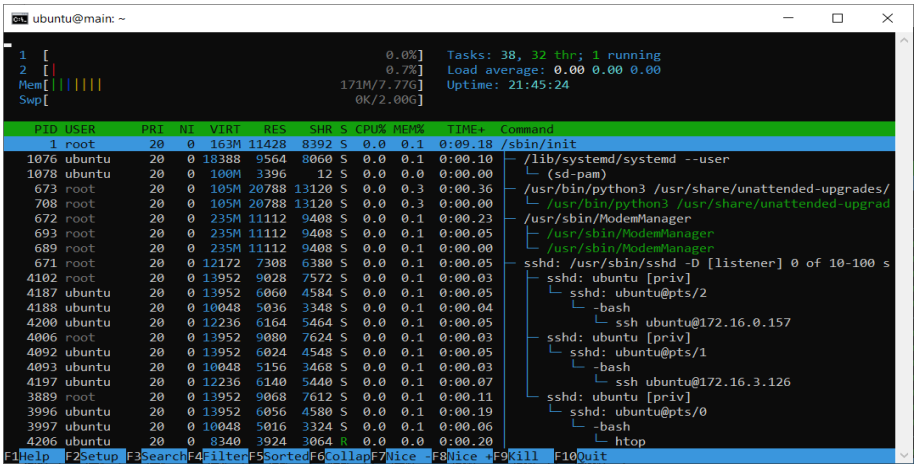


Ilustración 73 Visualización del uso de CPU, Memoria y procesos mediante el comando htop de la máquina virtual main.

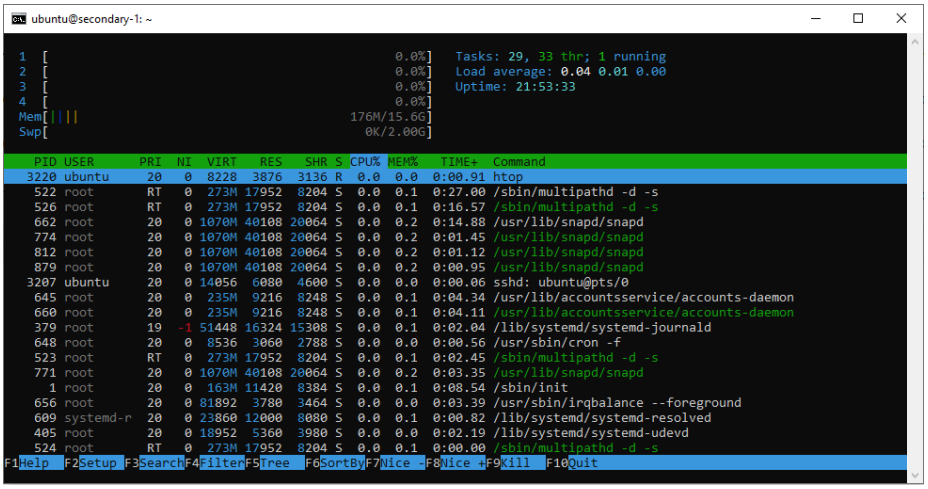


Ilustración 74 Visualización del uso de CPU, Memoria y procesos mediante el comando htop de la máquina virtual secondary-1

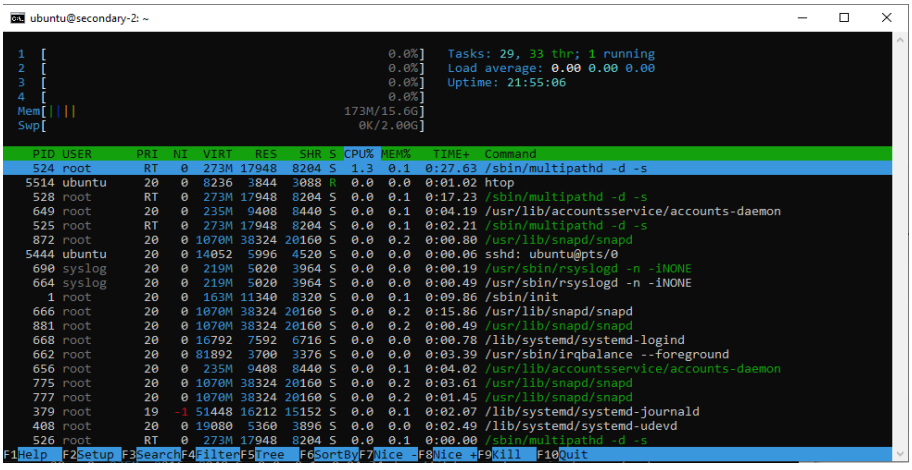


Ilustración 75 Visualización del uso de CPU, Memoria y procesos mediante el comando htop de la máquina virtual secondary-2

4.1.4. Evaluación del usuario

La tabla 18 muestra la evaluación de resultados mediante un cuestionario TAM que se aplica al usuario.

Tabla 18 Evaluación de resultados del cuestionario TAM

Respuesta	
Escala: 1 = Estoy de acuerdo ---- 5 = Estoy en desacuerdo	
Timestamp	2022/04/28 11:10:08 AM AST
Preguntas orientadas a la Utilidad percibida (PU)	
¿Las máquinas virtuales son lo suficientemente veloces para implementar mi proyecto?	1
¿Las máquinas virtuales tienen los recursos computacionales necesarios para implementar mi proyecto?	1
¿Las máquinas virtuales cumplen con la disponibilidad necesaria para implementar mi proyecto?	1
¿El tiempo acceso a las máquinas es lo suficientemente rápido?	3
¿El sistema operativo proporcionado facilita el desarrollo de mi proyecto?	1
¿La rapidez de aprovisionamiento de las máquinas virtuales ahorra el tiempo de desarrollo de mi proyecto?	1
Preguntas orientadas a la Percepción de facilidad de Uso (EOU)	
¿Es fácil conectarse a las máquinas virtuales?	1
¿Es fácil de usar el sistema operativo proporcionado?	1
Preguntas orientadas a la Aceptación de Usuario (UA)	
¿Utilizaría las máquinas virtuales para implementar otros proyectos?	1
¿Utilizaría las máquinas virtuales en el futuro?	1
¿Utilizaría las máquinas virtuales de forma recurrente?	1
Preguntas orientadas a la Seguridad	
Me preocupa la existencia de otras máquinas virtuales en la misma red interna	4
Me preocupa el acceso no autorizado a las máquinas virtuales desde el host	4

La retroalimentación de la evaluación del usuario acerca del uso del grupo de máquinas virtuales del caso de prueba fue en general positiva. Las máquinas virtuales cumplen con las expectativas del usuario para poder implementar su propio proyecto. Los dominios que sobresalió el prototipo fue la Percepción de facilidad de Uso (EOU) y la Aceptación de Usuario (UA) en donde recibieron una puntuación perfecta. El siguiente mejor dominio fue la Utilidad Percibida (PU), en donde se recibió una puntuación perfecta a excepción de la pregunta de tiempo de acceso a las máquinas virtuales. Por último, en el dominio de la Seguridad, el usuario no consideró inseguro tener sus máquinas virtuales alojadas en el prototipo.

4.2. Conclusiones

El desarrollo del prototipo de cloud computing se ha completado exitosamente y está listo para brindar infraestructura como servicio (IaaS). Esta plataforma eliminará la necesidad de trámites engorrosos y por tanto agilitará la asignación de recursos computacionales para el desarrollo de nuevos proyectos que los miembros del laboratorio o estudiantes de la facultad de sistemas deseen implementar.

El diseño de la asignación de recursos computacionales a partir de las guías proporcionadas por OpenStack, permitieron minimizar el riesgo e impacto generado a partir de un mal diseño. Esto promovió un desarrollo más fluido en donde todos los nodos cumplieron con los requisitos mínimos de hardware para operar normalmente en conjunto.

La selección del sistema operativo al principio del proyecto influyó en el desarrollo y resultado del proyecto. El cambio de sistema operativo que se tuvo que realizar en fases avanzadas del proyecto, consumió tiempo y recursos extras que resultó en atrasos a los tiempos establecidos en el cronograma de desarrollo. Para este proyecto, las características del sistema operativo ideal fueron: un repositorio con configuraciones y software actualizados, configuraciones válidas, y dependencias de software bien estructuradas.

Esta plataforma de código abierto permitirá a la facultad de Ingeniería en Sistemas brindar IaaS a los estudiantes y profesores para sus proyectos. Además, por ser altamente configurable y expandible, facilitará la integración y el manejo de nuevas tecnologías y dispositivos que la Escuela Politécnica Nacional adquiera o desarrolle.

Este puede ser el primer paso para implementar una Nube Privada a nivel universitario, que con el debido apoyo podría generar un espacio colaborativo entre facultades para el desarrollo de proyectos académicos más grandes.

La aplicación del modelo de aceptación de Tecnología TAM permitió evaluar la satisfacción del usuario frente al uso de las máquinas virtuales del caso de prueba. Los resultados demuestran que el proyecto cumple con los estándares y expectativas del usuario y que es fácil de utilizarlo lo cual demuestra que se ha cumplido con la razón y objetivo principal de este proyecto.

4.3. Recomendaciones

Para mejorar el rendimiento general del prototipo cada nodo debería trasladarse a su propia máquina física y dejar el nodo de cómputo en el servidor actual. De esta manera, el nodo aprovecharía mejor los recursos del servidor, permitiendo alojar más instancias y daría paso a la eliminación de UnRaid en un futuro para aprovechar los recursos de hardware directamente.

Mantener la disciplina al momento de utilizar Kanban. Es decir, no acumular tareas incompletas y enfocarse en el trabajo que necesita ser realizado para agregar valor al proyecto; la pizarra Kanban soporta cualquier número de columnas, por lo que se debe evitar la tentación de crear columnas innecesarias con el fin de mantener el orden de la visualización de trabajo.

Utilizar herramientas de automatización de configuración e implementación de software como Ansible, Chef o Puppet o herramientas de Infraestructura como Código (IaC). De este modo se puede automatizar el trabajo manual y gestionar la configuración de cada componente de manera consistente y ordenada que resulta en una cantidad menor de errores.

La mejorar la experiencia a medida que el número de usuarios aumenta se debe mejorar los recursos de red para soportar más conexiones y tráfico, modificar la red proveedora a una red de clase A para abarcar más máquinas virtuales con IP flotantes y por último cambiar la forma de acceso por medio de Port Forwarding a un túnel VPN para tener una mejor gestión de acceso, mayor seguridad y experiencia de usuario.

5. BIBLIOGRAFÍA

Benson, J. (2011). *Personal Kanban: Mapping Work .* Modus Cooperandi Press.

COE, N. (2020). *Resoluciones COE Nacional 17 de marzo 2020*. Obtenido de https://www.gestionderiesgos.gob.ec/resoluciones-coe-nacional-17-de-marzo-2020/

Davis, F. D. (1985). *A Technology Acceptance Model for Empirically Testing New End-user Information Systems.* Massachusetts Institute of Technology.

Devassy, H., Mukhedkar, P., & Vettathu, A. (2016). Mastering KVM Virtualization. Packt Publishing Ltd.

Docker. (2022). *Use containers to Build, Share and Run your applications*. Obtenido de https://www.docker.com/resources/what-container/

EPN. (2010). *Sistema de Investigación, Desarrollo, Innovación, y Vinculación (SIDiV) de la Escuela Politécnica Nacional.* Obtenido de https://atenea.epn.edu.ec/bitstream/25000/476/1/Sistema_Investigaci%C3%B3n_Desarrollo_Innovaci%C3%B3n_Vinculaci%C3%B3n.pdf

Intro, C. (2021). *WSI tutorial*. Obtenido de http://wsgi.tutorial.codepoint.net/intro

Jigsaw. (2021). *Best Open Source Cloud Software And Solutions* . Obtenido de https://www.jigsawacademy.com/blogs/cloud-computing/open-source-cloud-software

Kumar, N., Pramod, C., & Acken, J. (2020). Cloud Computing With Security. Santa Clara, California: Springer.

Lee, Y., Kozar, K. A., & Larsen, K. R. (2003). The Technology Acceptance Model: Past, Present,. *Communications of the Association for Information Systems*.

MariaDB Foundation. (2021). *MariaDB Server: The open source relational database*. Obtenido de https://mariadb.org/

Martinelli, S., Nash, H., & Topol, B. (2016). *Identity, Authentication, and Access Management in OpenStack.* O'Rilley.

Mell, & Grance, T. (2011). *The NIST Definition of Cloud Computing*. Obtenido de https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Memached. (2021). *What is Memcached?* Obtenido de https://memcached.org/

Mendoza, C. (2020). Tecnología en la educación ecuatoriana logros, problemas y debilidades. *Dominio de las Ciencias*, 17-21.

National Institute of Standards and Tecnology. (2011). Obtenido de The NIST Definition of Cloud: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf

Openinfra. (2019). *About Glance*. Obtenido de https://docs.openstack.org/glance/latest/

Openinfra. (2019). *Gnocchi*. Obtenido de https://wiki.openstack.org/wiki/Gnocchi

Openinfra. (2019). *OpenStack Wallaby Enhances Security and Cross-Project Integration with Other Open Source Technologies*. Obtenido de https://www.openstack.org/software/wallaby/

Openinfra. (2021). *About the OpenINfra Foundation*. Obtenido de https://openinfra.dev/about/

Openinfra. (2021). *Open Infrastructure*. Obtenido de https://openinfra.dev/

Openinfra. (2021). *OpenSUSE Contributors*. Obtenido de https://get.opensuse.org/leap/

Openinfra. (2021). *Red Hat, Inc. and others*. Obtenido de https://getfedora.org/

Openinfra. (2021). *The Most Widely Deployed Open Source Cloud Software in the World*. Obtenido de https://www.openstack.org/

Openstack. (2020). *Cinder block storage*. Obtenido de https://www.openstack.org/software/releases/mitaka/components/cinder

Openstack. (2020). *Object Storage API overview*. Obtenido de https://docs.openstack.org/swift/xena/api/object_api_v1_overview.html

Openstack. (2020). *Object Storage service overview*. Obtenido de https://docs.openstack.org/swift/xena/install/get_started.html

Openstack. (2020). *OpenStack Compute (nova)*. Obtenido de https://docs.openstack.org/nova/latest/

Openstack. (2021). *Celiometer*. Obtenido de https://docs.openstack.org/ceilometer/latest/

Openstack. (2021). *Cinder block storage service overview*. Obtenido de https://docs.openstack.org/cinder/latest/install/get-started-block-storage.html

Openstack. (2021). *DevStack*. Obtenido de https://docs.openstack.org/devstack/latest/

Openstack. (2021). *Horizon: the Openstack Dashboard project*. Obtenido de https://docs.openstack.org/horizon/latest/

Openstack. (2021). *https://docs.openstack.org/ceilometer/latest/install/get_started.html*. Obtenido de https://docs.openstack.org/ceilometer/latest/install/get_started.html

Openstack. (2021). *Identity service overview*. Obtenido de https://docs.openstack.org/keystone/latest/install/get-started-obs.html

Openstack. (2021). *Installation Guide*. Obtenido de https://docs.openstack.org/install-guide/

Openstack. (2021). *Keystone, the OpenStack Identity Service*. Obtenido de https://docs.openstack.org/keystone/latest/

Openstack. (2021). *ML2 plug-in Neutron*. Obtenido de https://docs.openstack.org/neutron/pike/admin/config-ml2.html#:~:text=The%20mechanism%20driver%20is%20responsible,with%20external%20devices%20or%20controllers.

Openstack. (2021). *Neutron*. Obtenido de https://docs.openstack.org/neutron/pike/admin/intro.html

Openstack. (2021). *OpenStack Architecture Design Guide*. Obtenido de OpenStack Architecture Design Guide: https://docs.openstack.org/arch-design/

OpenStack. (2021). *OpenStack Architecture Design Guide*. Recuperado el 08 de 03 de 2021, de OpenStack Architecture Design Guide: https://docs.openstack.org/arch-design/

Openstack. (2021). *Openstack client*. Obtenido de https://docs.openstack.org/python-openstackclient/pike/

OpenStack. (10 de 03 de 2021). *OpenStack Installation Guide*. Recuperado el 13 de 03 de 2021, de https://docs.openstack.org/install-guide/

Openstack. (2021). *Openstack Placement*. Obtenido de https://docs.openstack.org/placement/latest/

Openstack. (2021). *OpenStack Roadmap*. Obtenido de https://www.openstack.org/software/roadmap/

Openstack. (2021). *OpenStack Services*. Obtenido de https://www.openstack.org/software/project-navigator/openstack-components#openstack-services

Openstack. (2021). *OpenStack Virtual Machine Image Guide*. Obtenido de OpenStack Virtual Machine Image Guide: https://docs.openstack.org/image-guide/

Openstack. (2021). *Supported Software*. Obtenido de https://docs.openstack.org/horizon/latest/contributor/policies/supported-software.html

Openstack. (2022). *OpenStack API Documentation*. Obtenido de https://docs.openstack.org/api-ref/image/v2/index.html#general-information

Openstack. (2022). *OpenStack's 24th Release, Xena, Wields Powerful Hardware Support*. Obtenido de https://www.openstack.org/software/xena/

Pepple, K. (2011). *Deploying OpenStack.* O'Reilly.

Pepple, K. (2015). *Deploying OpenStack.* O'Reilly.

Pip. (2021). *Pip Project*. Obtenido de https://pypi.org/project/pip/

RabbitMQ. (2021). *RabbitMQ*. Obtenido de https://www.rabbitmq.com/

Red Hat. (2021). *LVM (Logical Volume Manager)*. Obtenido de https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/deployment_guide/ch-lvm

Singh, P. K., & Kumari, M. (2017). *Containers in OpenStack.* Packt Publishing.

Solberg, M., & Silverman, B. (2017). *OpenStack for Architects.* Packt.

Srinivasan, A., & Quadir, A. (2015). Era of Cloud Computing: A New Insight to Hybrid Cloud. *Elsevier*.

Statista. (2022). *Statista*. Obtenido de Cloud Infrastructure Market : https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/

Sunilkumar Manvi, G. S. (2021). *Cloud Computing, Concepts and Tecnologies .* CRCpress.

Ubuntu. (2021). *HTTPD - Apache2 Web Server*. Obtenido de https://ubuntu.com/server/docs/web-servers-apache

Unraid. (2021). *Unraid overview*. Obtenido de https://wiki.unraid.net/UnRAID_6/Overview

Vault. (2020). *What is Vault?* Obtenido de https://www.vaultproject.io/docs/what-is-vault

WSI. (2021). *WSI*. Obtenido de https://wsgi.readthedocs.io/en/latest/what.html

6. ANEXOS

6.1. Anexo 1: lista de ilustraciones

6.2. Anexo 2: Lista de tablas

6.3. Anexo 3: Lista de Comandos

6.4. Anexo 4: Lista de configuraciones

6.5. Anexo 5: Lista de Endpoints

6.6. Anexo 6: Lista de Endpoints

```
root@controller:~# openstack endpoint list
+----------------------------------+-----------+--------------+--------------+---------+-----------+-----------------------------------------------+
| ID                               | Region    | Service Name | Service Type | Enabled | Interface | URL                                           |
+----------------------------------+-----------+--------------+--------------+---------+-----------+-----------------------------------------------+
| 27d35df24a964514a1f1dcd94261f305 | RegionOne | neutron      | network      | True    | public    | http://controller:9696                        |
| 42b9e32200e446298d57e907f31d8d41 | RegionOne | placement    | placement    | True    | admin     | http://controller:8778                        |
| 4392dbe9db6945dbaa3bf1d535263656 | RegionOne | gnocchi      | metric       | True    | internal  | http://controller:8041                        |
| 4dc1b623413d4266aa2557bb3367df94 | RegionOne | swift        | object-store | True    | admin     | http://controller:8080/v1                     |
| 4e3a01f113be4879a0c7305f86c6dee1 | RegionOne | nova         | compute      | True    | admin     | http://controller:8774/v2.1                   |
| 4e8da997cb414ebdbb8a9ddd6a9b7256 | RegionOne | neutron      | network      | True    | admin     | http://controller:9696                        |
| 5e12f0f24ab8431a8381d0cd43341b2e | RegionOne | nova         | compute      | True    | public    | http://controller:8774/v2.1                   |
| 61635640fc084686868ebf8ef9d71b15 | RegionOne | swift        | object-store | True    | internal  | http://controller:8080/v1/AUTH_%(project_id)s |
| 631649a35ee44fd8906793af45e322c3 | RegionOne | glance       | image        | True    | internal  | http://controller:9292                        |
| 68d0cd963d9d4919813ecd8e5194eccb | RegionOne | gnocchi      | metric       | True    | admin     | http://controller:8041                        |
| 6bd5f9f2369f406db4d5745919711da9 | RegionOne | swift        | object-store | True    | public    | http://controller:8080/v1/AUTH_%(project_id)s |
| 7d7a918252c64477907cea1eba25c51d | RegionOne | keystone     | identity     | True    | public    | http://controller:5000/v3/                    |
| 82dbe4c8eb9d48da8b493730bd513ab1 | RegionOne | glance       | image        | True    | admin     | http://controller:9292                        |
| 85aa45e893d848529e68510bd359ea9c | RegionOne | keystone     | identity     | True    | internal  | http://controller:5000/v3/                    |
| 92e0eaf18e0f4725b014a8a32552dfbc | RegionOne | neutron      | network      | True    | internal  | http://controller:9696                        |
| 9e115c0ec8a84025a6d6f210941a1f22 | RegionOne | nova         | compute      | True    | internal  | http://controller:8774/v2.1                   |
| a6ae28e126304ab2aaed822f225f7483 | RegionOne | cinderv3     | volumev3     | True    | public    | http://controller:8776/v3/%(project_id)s      |
| b89307dd43fd4ddc8993d99ddea39d24 | RegionOne | gnocchi      | metric       | True    | public    | http://controller:8041                        |
| d78c9591ccd0499c94c5f5c6d5d34d56 | RegionOne | placement    | placement    | True    | internal  | http://controller:8778                        |
| dc9e6631682746be8e446a6a79b6291c | RegionOne | placement    | placement    | True    | public    | http://controller:8778                        |
| e374cc326919484fa18261d3b052a29e | RegionOne | cinderv3     | volumev3     | True    | admin     | http://controller:8776/v3/%(project_id)s      |
| e77100b012e240618a82bd2d7d7fa89b | RegionOne | glance       | image        | True    | public    | http://controller:9292                        |
| e8bcb4b12b3c4ae199586cc469de0171 | RegionOne | keystone     | identity     | True    | admin     | http://controller:5000/v3/                    |
| f2b67274170c45e6ba71871d9ba91644 | RegionOne | cinderv3     | volumev3     | True    | internal  | http://controller:8776/v3/%(project_id)s      |
+----------------------------------+-----------+--------------+--------------+---------+-----------+-----------------------------------------------+
root@controller:~#
```

6.7. Anexo 7: Lista de Servicios OpenStack

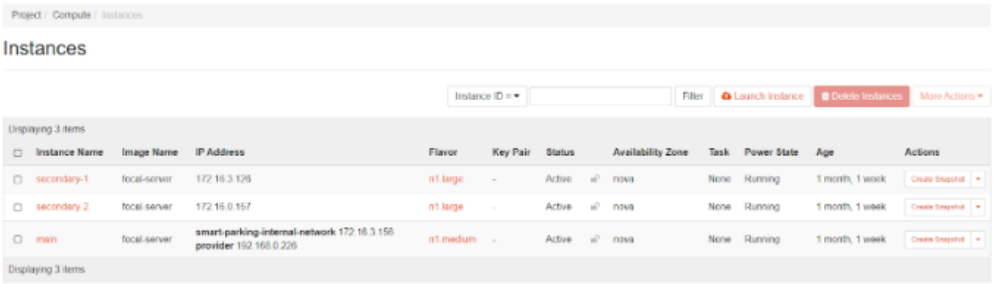
```
root@controller:~# openstack service list
+----------------------------------+------------+--------------+
| ID                               | Name       | Type         |
+----------------------------------+------------+--------------+
| 07a336cd8736443bbdce0824ed27f644 | neutron    | network      |
| 20e93aca7fd9492588b104d9623839a7 | nova       | compute      |
| 38f7c0590956424591cc46b189a24cf6 | gnocchi    | metric       |
| 4ed68c006531408ebe1570ae07942f30 | keystone   | identity     |
| 4ffcbc7dc76545b387070d3ba85359a6 | placement  | placement    |
| 7c6da928cb3b4b6bb2e59cf8ae696caa | cinderv3   | volumev3     |
| 8191ee4c076642dca92510f13b3570a2 | glance     | image        |
| 893ae1b0cbee44c18162a36eae219243 | swift      | object-store |
| aeb4d333465b4f6296495e410ed7bb9f | ceilometer | metering     |
+----------------------------------+------------+--------------+
root@controller:~#
```

6.8. Anexo 8: Cuestionario TAM

Technology Access Model (TAM)

El siguiente cuestionario tiene como objetivo evaluar la respuesta conductual del usuario respecto al uso de

Lista de máquinas virtuales en el prototipo de Cloud Computing



Explicación:

El usuario puede ingresar desde Internet a la máquina virtual principal (main) por medio de SSH y acceder a las otras dos máquinas por medio de la red interna que las conecta.

Nota:

Se recomienda utilizar el cliente OpenSSH para conectarse a la máquina virtual principal (main)

Preguntas orientadas a la Utilidad percibida (PU)

Description (optional)

¿Las máquinas virtuales son lo suficientemente veloces para implementar mi proyecto? *

	1	2	3	4	5	
Totalmente de acuerdo	○	○	○	○	○	Totalmente en desacuerdo

¿Las máquinas virtuales tienen los recursos computacionales necesarios para implementar mi proyecto? *

	1	2	3	4	5	
Totalmente de acuerdo	○	○	○	○	○	Totalmente en desacuerdo

¿Las máquinas virtuales cumplen con la disponibilidad necesaria para implementar mi proyecto? *

	1	2	3	4	5	
Totalmente de acuerdo	○	○	○	○	○	Totalmente en desacuerdo

¿El tiempo accesso a las máquinas es lo suficientemente rápido? *

	1	2	3	4	5	
Totalmente de acuerdo	○	○	○	○	○	Totalmente en desacuerdo

¿El sistema operativo proporcionado facilita el desarrollo de mi proyecto? *

	1	2	3	4	5	
Totalmente de acuerdo	○	○	○	○	○	Totalmente en desacuerdo

¿La rapidez de aprovisionamiento de las máquinas virtuales ahorra el tiempo de desarrllo de mi proyecto? *

	1	2	3	4	5	
Totalmente de acuerdo	○	○	○	○	○	Totalmente en desacuerdo

Preguntas orientadas a la Percepción de facilidad de Uso (EOU)

Description (optional)

¿Es fácil conectarse a las máquinas virtuales?

	1	2	3	4	5	
Totalmente de acuerdo	○	○	○	○	○	Totalmente en desacuerdo

¿Es fácil de usar el sistema operativo proporcionado?

	1	2	3	4	5	
Totalmente de acuerdo	○	○	○	○	○	Totalmente en desacuerdo

Preguntas orientadas a la Aceptación de Usuario (UA)

Description (optional)

¿Utilizaría las máquinas virtuales para implementar otros proyectos?

	1	2	3	4	5	
Totalmente de acuerdo	○	○	○	○	○	Totalmente en desacuerdo

¿Utilizaría las máquinas virtuales en el futuro?

	1	2	3	4	5	
Totalmente de acuerdo	○	○	○	○	○	Totalmente en desacuerdo

¿Utilizaría las máquinas virtuales de forma recurrente?

	1	2	3	4	5	
Totalmente de acuerdo	○	○	○	○	○	Totalmente en desacuerdo

Preguntas orientadas a la Seguridad

Description (optional)

Me preocupa la existencia de otras máquinas virtuales en la misma red interna

	1	2	3	4	5	
Totalmente de acuerdo	○	○	○	○	○	Totalmente en desacuerdo

Me preocupa el accesso no autorizado a las máquinas virtuales desde el host

	1	2	3	4	5	
Totalmente de acuerdo	○	○	○	○	○	Totalmente en desacuerdo