

UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS



Prototipo centrado en una Infraestructura de

Nube Comunitaria Geo-distribuida

PRESENTADO POR:

Angulo García, Roxana Mercedes

Hernández Cortez, Jairo Antonio

Novoa Bernabé, Valeria Michelle

PARA OPTAR AL TÍTULO DE:

INGENIERO DE SISTEMAS INFORMÁTICOS

CIUDAD UNIVERSITARIA, NOVIEMBRE DE 2023

UNIVERSIDAD DE EL SALVADOR

RECTOR:

MSC. JUAN ROSA QUINTANILLA

SECRETARIO GENERAL:

LIC. PEDRO ROSALÍO ESCOBAR CASTANEDA

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DECANO:

ING. LUIS SALVADOR BARRERA MANCÍA

SECRETARIO:

ARQ. RAUL ALEXANDER FABIAN ORELLANA

ESCUELA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

DIRECTOR:

ING. CÉSAR AUGUSTO GONZÁLEZ

UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO DE SISTEMAS INFORMÁTICOS

Título:

Prototipo centrado en infraestructura de nube comunitaria geo-distribuida

Presentado por:

Angulo García, Roxana Mercedes

Hernández Cortez, Jairo Antonio

Novoa Bernabé, Valeria Michelle

Trabajo de Graduación Aprobado por:

Docente Asesor:

MSc. DAMIÁN MORALES

SAN SALVADOR, NOVIEMBRE DE 2023

Trabajo de Graduación Aprobado por:

Docente Asesor:

MSc. DAMIÁN MORALES

Agradecimientos

A Dios por caminar de la mano conmigo y ayudarme a superar obstáculos y llegar a culminar esta etapa de mi vida.

A mis padres por su apoyo constante en esta aventura y a lo largo de mi vida, por siempre estar presentes, tenerme paciencia y brindarme su cariño incondicional y valiosos consejos.

A mis mejores amigas por darme una mano siempre que lo he necesitado, sus palabras de aliento, tiempo y cariño a pesar de la vida tan apresurada y ocupada que todos vivimos hoy en día.

A los padres de mis mejores amigas por recibirme con brazos abiertos cada vez que teníamos trabajos en grupo y debíamos reunirnos.

A los docentes que me han animado a seguir adelante y que han sabido transmitir sus conocimientos a lo largo de la carrera.

Roxana Angulo

Quiero expresar mi profundo agradecimiento a Dios, quien ha sido mi guía y fortaleza a lo largo de este arduo pero gratificante viaje académico. Su infinita sabiduría y amor han sido la luz que ha iluminado mi camino, dándome la fuerza necesaria para superar los desafíos y alcanzar este logro.

A mis padres, Israel Hernández y Rosa Del Carmen Cortez, les debo mi más sincero agradecimiento. Su apoyo inquebrantable, sacrificios y amor incondicional han sido la base sobre la cual he construido mi educación y mi vida. Cada éxito que alcanzo es también suyo, y estoy eternamente agradecido por su dedicación y confianza en mí.

A mis queridos hermanos, Josemaría y Selvin, agradezco su constante aliento y comprensión. Su presencia ha sido un faro de apoyo emocional, haciéndome sentir respaldado en cada paso de

esta carrera. Su influencia positiva ha sido un recordatorio constante de la importancia de la unidad familiar, y estoy agradecido por tenerlos como cómplices en este viaje hacia el logro de mis metas.

A mi amada, Flor, quiero expresar mi más profundo agradecimiento. Su amor incondicional, paciencia y apoyo constante han sido pilares fundamentales en mi vida. Su presencia ha sido mi fuente de inspiración y motivación. Gracias por comprender mis compromisos, por celebrar mis logros y por ser mi compañera inquebrantable en este emocionante viaje. Tu presencia ha convertido cada desafío en una oportunidad para crecer juntos, y estoy agradecido por tenerte a mi lado.

Este logro no hubiera sido posible sin el apoyo de cada persona que ha cruzado mi camino. A todos ustedes, les dedico mi más sincero agradecimiento.

Con gratitud,

Jairo Hernández

Quiero aprovechar esta oportunidad para expresar mi más profundo agradecimiento a Dios, mi familia y amigos, cuyo apoyo ha sido un pilar fundamental en la realización de esta tesis.

A mis padres por su amor incondicional, su paciencia y su fe inquebrantable en mí. Su apoyo emocional y aliento constante han sido esenciales en cada paso de este camino. Su presencia y comprensión en los momentos más desafiantes de este proceso han sido una fuente de motivación y fortaleza.

También deseo agradecer sinceramente a mis amigos Gabriela, Roxana, Jaime, Daniela, Martha, Nicole, Milady y Lucia, por estar siempre allí para mí. Gracias por todas las palabras de aliento, los momentos de relajación y diversión que me ayudaron a mantener el equilibrio, y por ser una

fuente constante de alegría y apoyo. Vuestra amistad ha sido una luz brillante durante los tiempos de estudio y esfuerzo.

Este logro no sólo es un reflejo de mi esfuerzo, sino también de la amorosa presencia y el apoyo incondicional que he recibido de cada uno de ustedes. Gracias por ser parte de este viaje.

Valeria Novoa

Tabla de contenido

1.	Introducción	1
2.	Antecedentes	3
3.	Justificación	5
4.	Pregunta de investigación	7
5.	Objetivos	8
5.1.	Objetivo General.....	8
5.2.	Objetivos Específicos	8
6.	Alcances	10
7.	Limitaciones.....	12
8.	Marco teórico	13
8.1.	Computación en la Nube	13
8.1.1.	Modelos de Servicio en la Nube	15
8.1.2.	Modelo de despliegue en la nube	16
8.1.3.	Infraestructura de la Nube	20
8.1.4.	Arquitectura de la Nube	21
8.1.5.	Regiones en la nube.....	21
8.1.6.	Cluster	24
8.2.	Virtualización	25
8.2.1.	Hipervisor.....	26

8.3.	OpenStack como Plataforma de Nube.....	29
8.3.1.	Componentes Principales de OpenStack.....	30
8.3.2.	Mensajería y colas.....	36
8.4.	Orquestación.....	38
8.4.1.	Orquestación del almacenamiento de datos en la nube.....	39
8.4.2.	Almacenamiento en Caché.....	40
8.5.	Control y accesos en la nube.....	42
8.5.1.	Usuario.....	42
8.5.2.	Rol.....	42
8.5.3.	Usuarios Federados.....	43
8.5.4.	Lightweight Directory Access Protocol (LDAP).....	43
8.5.5.	OpenLDAP.....	44
9.	Metodología de la investigación.....	46
9.1.	Población y muestreo.....	46
9.1.1.	Instrumentos y Técnicas de Recopilación de Datos:.....	47
9.2.	Resultados.....	48
9.2.1.	Análisis de datos cuantitativos.....	48
10.	Desarrollo de la solución.....	54

10.1.	Requerimientos Funcionales:	54
10.2.	Arquitectura	56
10.3.	Configuración	59
10.4.	Topología de red	61
11.	Caso de estudio.....	62
12.	Factibilidad.....	64
12.1.	Factibilidad Técnica:	64
12.2.	Factibilidad Económica:	65
12.3.	Comparativa de Costos	67
12.4.	Factibilidad humana	68
13.	Conclusiones	70
14.	Recomendaciones.....	72
15.	Referencias	74
16.	Anexo	78
16.1.	Encuesta.....	78

TABLA DE ILUSTRACIONES

ILUSTRACIÓN 1: CLASIFICACIÓN DE LA COMPUTACIÓN EN NUBE EN DOS CONJUNTOS DISTINTOS DE MODELOS ...	14
ILUSTRACIÓN 2: MODELO DE SERVICIO EN LA NUBE	15
ILUSTRACIÓN 3: MODELO DE DESPLIEGUE EN LA NUBE	17
ILUSTRACIÓN 4: COMO SE COMUNICAN LOS SERVICIOS DE OPENSTACK.....	35
ILUSTRACIÓN 5: DIAGRAMA DE LA ARQUITECTURA DEL PROTOTIPO EN LA NUBE COMUNITARIA ENFOCADO EN LA INFRAESTRUCTURA	56
ILUSTRACIÓN 6: TOPOLOGÍA DE RED DE LA INFRAESTRUCTURA	61
ILUSTRACIÓN 7: DIAGRAMA DE NUBE COMUNITARIA.....	63

1. Introducción

En la era digital actual, la computación en la nube se ha establecido como un pilar esencial para el almacenamiento, procesamiento y acceso a datos, siendo una herramienta clave tanto para usuarios individuales como para organizaciones en todo el mundo. A pesar de su amplia adopción, el acceso a estos servicios de nube a menudo está restringido a entidades o regiones individuales, lo que plantea desafíos significativos en términos de colaboración y compartición de recursos a nivel global. Abordando esta problemática, nuestra investigación, realizada por un equipo multidisciplinario de expertos en tecnología de la información y desarrollo de software, se centra en el desarrollo de un prototipo de nube comunitaria. Esta nube, basada en la robusta infraestructura de OpenStack, busca superar las limitaciones de las nubes tradicionales, promoviendo una cooperación efectiva y un intercambio fluido de recursos entre diversas regiones geográficas.

El proyecto se concentra en diseñar la infraestructura necesaria para implementar una nube comunitaria, caracterizada por su capacidad de integrar un número ilimitado de regiones. Se presta especial atención a garantizar un funcionamiento eficiente y seguro, contemplando la federación de usuarios y la persistencia de sesión para proporcionar una experiencia de usuario coherente y continua. Una parte crucial del proyecto es la creación de una interfaz web intuitiva, diseñada para simplificar la gestión y el monitoreo de las distintas regiones de la nube comunitaria, asegurando así una administración eficaz de los servicios disponibles.

Además, el estudio se contextualiza en un caso práctico específico: una empresa de logística de transporte de vehículos operando entre Estados Unidos y El Salvador. La aplicación de la nube comunitaria en este entorno busca optimizar los procesos logísticos, con una infraestructura central en El Salvador y extensiones en varias regiones estadounidenses. Este caso de estudio provee un marco tangible para evaluar la implementación del prototipo de nube comunitaria, ilustrando su efectividad y los desafíos inherentes en un contexto empresarial real.

En resumen, esta investigación se enfoca en el desarrollo de un prototipo de nube comunitaria diseñado para impulsar la cooperación y el intercambio de recursos a través de fronteras regionales, utilizando OpenStack como su piedra angular. El objetivo es crear un modelo de nube que no solo atienda las necesidades actuales de los usuarios, sino que también establezca un marco para el futuro de la gestión eficiente de recursos en la nube comunitaria.

2. Antecedentes

Las infraestructuras de nube han revolucionado la forma en que las organizaciones despliegan y gestionan sus recursos digitales. En el sector de la logística y el transporte, la capacidad de escalar recursos rápidamente y gestionar grandes volúmenes de datos ha sido fundamental. La nube ha permitido a empresas como las de logística de vehículos mejorar su eficiencia operativa y capacidad de respuesta.

Las infraestructuras de nube han revolucionado la forma en que las organizaciones despliegan y gestionan sus recursos digitales. En el sector de la logística y el transporte, la capacidad de escalar recursos rápidamente y gestionar grandes volúmenes de datos ha sido fundamental. La nube ha permitido a empresas como las de logística de vehículos mejorar su eficiencia operativa y capacidad de respuesta.

Con el desarrollo de Microsoft Azure, Google Cloud Platform y otros servicios, la nube se ha convertido en una solución omnipresente para empresas de todos los tamaños, incluyendo aquellas en el sector logístico.

Desde su lanzamiento en el 2010, OpenStack se ha destacado por su adaptabilidad a diferentes entornos, convirtiéndose en una opción preferida para organizaciones que buscan soluciones de nube personalizadas. Su desarrollo continuo ha permitido adaptaciones específicas para industrias como la logística de vehículos.

Las nubes comunitarias ganaron tracción como una solución colaborativa para compartir recursos en sectores como la educación y la investigación. Programas como EGI

Federated Cloud han demostrado cómo diferentes organizaciones pueden beneficiarse de la colaboración en la nube.

La autenticación federada y el SSO, con tecnologías como OAuth y SAML, se han convertido en estándares de facto para la gestión de identidades en la última década. LDAP, implementado inicialmente en los 90, sigue siendo un componente crucial en muchos sistemas.

Las interfaces web para la gestión de sistemas en la nube han evolucionado continuamente, con ejemplos notables como el panel de control de AWS lanzado en los 2000 y las iteraciones sucesivas en plataformas como OpenStack Horizon.

Este panorama histórico y técnico ilustra cómo las tecnologías relacionadas con la computación en la nube han evolucionado para satisfacer las necesidades cambiantes de las organizaciones y cómo han influido en sectores críticos como la logística y el transporte de vehículos. La adaptabilidad y la innovación continua en estas áreas no solo han permitido una gestión más eficiente y segura de los recursos digitales, sino que también han abierto camino para nuevas formas de colaboración y operación en la era digital. Este contexto enriquece la base sobre la cual se desarrolla el prototipo de Nube Comunitaria para la empresa de logística de vehículos, resaltando la relevancia y la oportunidad que representa este proyecto en el marco actual de las tecnologías de nube.

3. Justificación

La implementación de una nube comunitaria basada en OpenStack para una empresa de logística de vehículos que opera entre Estados Unidos y El Salvador representa una estrategia crucial para enfrentar los desafíos contemporáneos y aprovechar las oportunidades en el ámbito de la tecnología de la información. Esta iniciativa se justifica por la necesidad de mejorar la eficiencia operativa y la gestión de datos en un entorno cada vez más digitalizado y competitivo.

La infraestructura de TI actual, limitada en términos de escalabilidad y flexibilidad, se verá mejorada con la adopción de una nube comunitaria. Esta nueva infraestructura promete no solo una gestión más eficiente de los recursos, sino también una reducción en los tiempos de procesamiento y una mejor coordinación entre las diferentes regiones operativas. La expectativa es que tales mejoras se reflejen en una reducción notable de los tiempos de procesamiento, aumentando la capacidad de respuesta de la empresa ante las demandas del mercado.

Además, la centralización de los datos en una nube comunitaria facilitará un acceso más rápido y seguro a la información crítica, algo esencial para la toma de decisiones estratégicas y para mantener la integridad de los datos a lo largo de toda la cadena logística. La implementación de una base de datos centralizada y un sistema de gestión de identidades robusto promete mejorar la coherencia y seguridad de los datos, aspectos fundamentales en el sector logístico.

Por otro lado, se espera que la implementación de una interfaz web intuitiva y una autenticación federada simplifiquen el acceso y la gestión de los sistemas para los usuarios finales, mejorando su experiencia y reduciendo la curva de aprendizaje. La adopción de estas tecnologías avanzadas en la nube no solo mejorará las operaciones actuales, sino que también fomentará la innovación y mantendrá a la empresa competitiva en un mercado en constante cambio.

Desde el punto de vista de la factibilidad, OpenStack, como una plataforma de nube de código abierto probada y confiable, garantiza una base sólida para el desarrollo del proyecto. La infraestructura propuesta es viable, con numerosos casos de éxito en industrias similares que respaldan su aplicación y beneficios. Aunque la implementación inicial implica una inversión grande, el retorno a largo plazo en términos de eficiencia operativa y reducción de costos justifica este gasto. Además, la escalabilidad de la nube comunitaria permitirá un crecimiento sostenible con una inversión incremental, y la existencia de una amplia comunidad y soporte para OpenStack asegura asistencia continua y actualizaciones para mantener el sistema operativo y seguro.

En resumen, la nube comunitaria basada en OpenStack se presenta como una oportunidad estratégica no solo para responder a las necesidades actuales de la empresa, sino también para posicionarla de manera óptima para futuras innovaciones y mejoras en un sector altamente competitivo. La implementación de este proyecto promete transformar la manera en que la empresa gestiona sus operaciones logísticas, marcando un hito significativo en su evolución tecnológica y operativa.

4. Pregunta de investigación

¿Cómo puede implementarse de manera eficiente un prototipo de infraestructura de Nube Comunitaria basada en OpenStack que brinde a las entidades miembros una plataforma centralizada para compartir recursos tecnológicos, administrar múltiples regiones y garantizar la autenticación segura de usuarios y el acceso a servicios a través de un API?

5. Objetivos

5.1. Objetivo General

Desarrollar un prototipo de Infraestructura de nube comunitaria basada en OpenStack compuesta por múltiples regiones, ofreciendo una interfaz Web para la gestión de las regiones, servicio de autenticación, basado en usuarios servicios federados y persistente a servicios en la nube, garantizando la disponibilidad y seguridad de sus datos.

5.2. Objetivos Específicos

1. Diseñar y desarrollar un prototipo de infraestructura de nube comunitaria basada en OpenStack que permita el despliegue de múltiples regiones.
2. Validar la funcionalidad de usuarios federados y persistencia de sesión en la nube comunitaria.
3. Configurar la interfaz web para la gestión y supervisión de las diferentes regiones dentro de la nube.
4. Configurar una base de datos centralizada que respalde aspectos clave de la nube comunitaria y la operación logística.
5. Crear un servicio de identificación que proporcione autenticación de usuarios y descubrimiento de servicios para toda la nube comunitaria.
6. Establecer una API que permita a las diferentes regiones consumir los servicios proporcionados por la nube comunitaria.

7. Realizar pruebas exhaustivas del prototipo para validar su funcionalidad, escalabilidad y seguridad.
8. Crear documentación detallada sobre la infraestructura y configuración de la nube comunitaria.

6. Alcances

1. Diseño y desarrollo de la infraestructura basada en OpenStack para el despliegue de una Nube Comunitaria compuesta por un número ilimitado de regiones.
2. Desarrollo de un sistema de usuarios federados que permita a los usuarios acceder a diferentes regiones de la nube con una única identidad, facilitando la integración y simplificación de la autenticación.
3. Desarrollo de mecanismos de persistencia de sesión para garantizar que los usuarios mantengan su sesión activa al moverse entre regiones, evitando interrupciones y pérdida de datos.
4. Creación de una interfaz web intuitiva y fácil de usar que permita a los usuarios gestionar y administrar las diferentes regiones de la nube comunitaria, brindando control y visibilidad sobre los recursos y servicios disponibles.
5. Desarrollo de una base de datos centralizada que almacene y gestione los datos de los usuarios en todas las regiones, asegurando la consistencia y disponibilidad de la información.
6. Desarrollo de un servicio de identificación seguro y confiable que permita la autenticación de los usuarios en todas las regiones de la nube comunitaria, así como el descubrimiento de servicios disponibles, facilitando la integración de aplicaciones y servicios.

7. Creación de un API que permita la comunicación entre las diferentes regiones de la nube comunitaria, facilitando el consumo de los servicios proporcionados y asegurando la interoperabilidad, excluyendo la integración de un sistema de almacenamiento ya que esto va en un desarrollo más profundo de regiones.

7. Limitaciones

Coordinación entre Regiones: La gestión de múltiples regiones, cada una con sus propias características logísticas y requisitos específicos, puede requerir una coordinación efectiva. La necesidad de mantener una sincronización y coherencia en la operación entre las diferentes yardas y regiones se convierte en un desafío.

Dependencia de Proveedores Externos: Para ciertos componentes y servicios, puede haber una dependencia de proveedores externos, lo que podría generar riesgos en términos de costos, disponibilidad y control.

Dependencia de la Conectividad a Internet: Al estar basada en la nube, la infraestructura depende en gran medida de una conexión a internet estable y rápida, lo que podría ser un problema en regiones con infraestructura de internet limitada.

8. Marco teórico

En el presente marco teórico, exploraremos en detalle los elementos esenciales que componen esta Nube Comunitaria basada en OpenStack. Desde la infraestructura subyacente hasta la implementación de usuarios federados y la persistencia de sesiones, pasando por la interfaz web para la gestión de regiones, la base de datos y el servicio de identificación, cada componente desempeña un papel crucial en la habilitación de este entorno tecnológico colaborativo.

8.1. Computación en la Nube

La Nube es el conjunto “infinito” de servidores de información (computadoras) desplegados en centros de datos, a lo largo de todo el mundo donde se almacenan millones de aplicaciones Web y enormes cantidades de datos (big data) a disposición de miles de organizaciones y empresas, y cientos de miles de usuarios que se descargan y ejecutan directamente los programas y aplicaciones de software almacenados en dichos servidores tales como Google, Amazon, IBM o Microsoft. (Aguilar, 2012, pág. 19)

Computación en la nube se puede definir como un sistema de computación distribuido orientado al consumidor, que consiste en una colección de ordenadores virtualizado e interconectados que son suministrados dinámicamente y presentados como uno o más recursos computacionales unificados, conforme acuerdo de nivel de servicio negociado entre el proveedor de servicios y el consumidor. (Arias, 2015, pág. 13).

En resumen, la nube es un conjunto de servidores desplegados en centros de datos por todo el mundo, donde se almacenan aplicaciones web y grandes cantidades de datos a disposición de miles de organizaciones y usuarios. La computación en la nube es un sistema de computación distribuido enfocado en el consumidor, donde una colección de ordenadores conectados se presenta como uno o más recursos computacionales unificados, suministrados dinámicamente de acuerdo con un acuerdo de nivel de servicio negociado entre consumidor y proveedor de servicios.

El NIST (National Institute of Standards and Technology) junto con la mayoría de los usuarios y proveedores de la nube, clasifican la computación en nube en dos conjuntos distintos de modelos: los modelos de despliegue y los modelos de servicio.

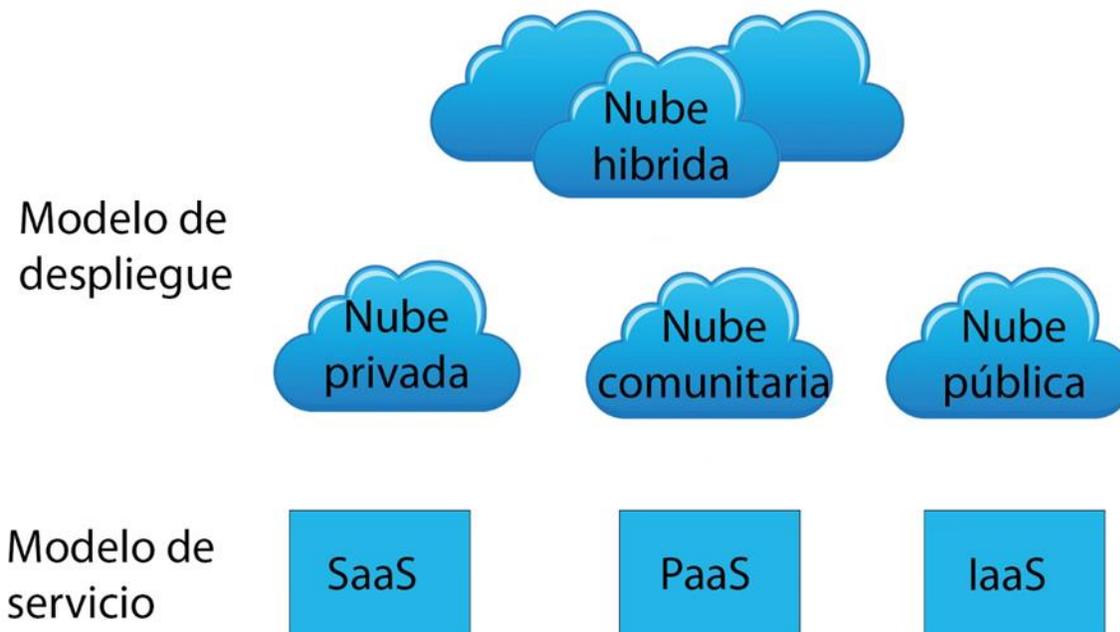


Ilustración 1: Clasificación de la computación en nube en dos conjuntos distintos de modelos

8.1.1. Modelos de Servicio en la Nube

Los modelos de servicio se refieren a los tipos específicos de servicios a los que se puede acceder en una plataforma de computación en la nube. Estos modelos son el software como servicio (SaaS), plataforma como servicio (PaaS) e infraestructura como servicio (IaaS).

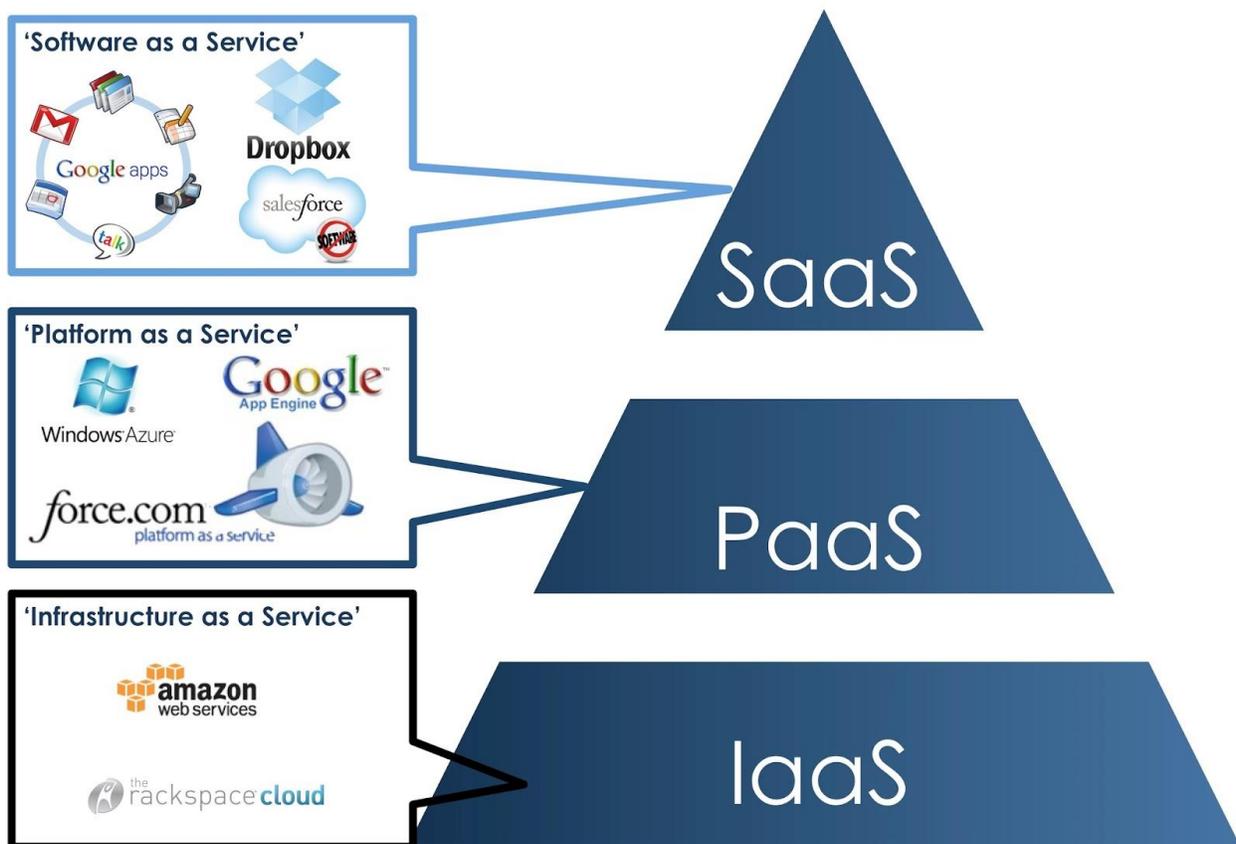


Ilustración 2: modelo de servicio en la nube

8.1.1.1. Infraestructura como Servicio (IaaS)

En este caso se contrata sólo capacidad de almacenamiento básico y de cálculo computacional (CPU), como servidores, discos duros, conexiones, enrutadores y otros sistemas para manejar tipos específicos de cargas de trabajo. Es decir, ofrece un hardware virtualizado bajo demanda, normalmente para empresas. El proveedor se encarga de su gestión y el cliente sólo paga por lo que usa. Ejemplos: Amazon Web Services, IBM SmartCloud, etc.

8.1.1.2. Plataforma como Servicio (PaaS)

En este caso la nube proporciona además un servidor de aplicaciones a modo de plataforma donde se podrán desarrollar y ejecutar aplicaciones. Es la nube para desarrolladores, proporcionando todo lo necesario para desarrollar y poner en marcha aplicaciones y servicios web en internet. Ejemplos: Microsoft Windows Azure, Google App Engine, etc.

8.1.1.3. Software como Servicio (SaaS)

El usuario final puede pagar un alquiler por el uso de software, o bien se le proporciona de forma gratuita. No es necesario adquirir un software en propiedad, instalarlo, configurarlo ni mantenerlo. Ejemplos: Dropbox, Gmail, etc. (Luna, 2014, pág. 6).

8.1.2. Modelo de despliegue en la nube

La ubicación de los servidores puede determinar distintos tipos de implementaciones en la nube, como la nube pública, la nube privada, la nube híbrida y la nube comunitaria.

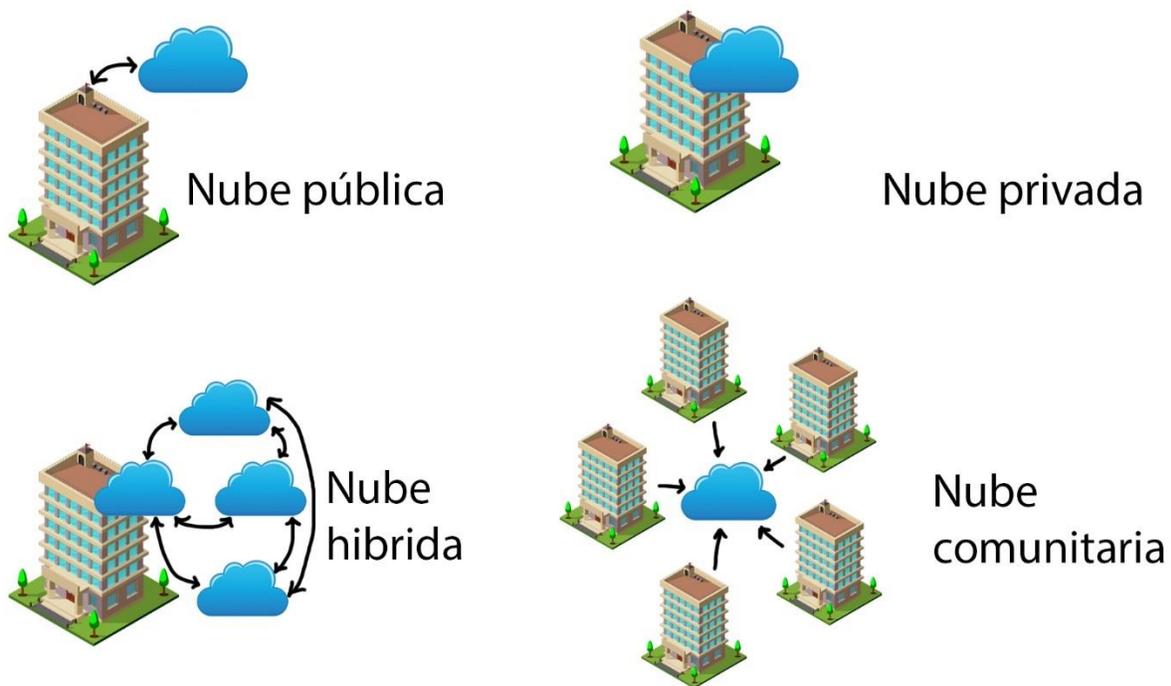


Ilustración 3: Modelo de despliegue en la nube

8.1.2.1. Nube Pública

La nube pública se establece como un método estándar y ampliamente adoptado para desplegar servicios de computación en la nube. En este esquema, los recursos computacionales como servidores y espacios de almacenamiento son proporcionados por entidades externas, separadas de la infraestructura interna de una empresa. Esto conlleva a que tanto la gestión de estos recursos como su accesibilidad se realicen exclusivamente a través de Internet. Este modelo permite a las organizaciones utilizar una red de recursos de nube, similar a un alojamiento compartido, maximizando así las capacidades de procesamiento y almacenamiento disponibles bajo su contrato de servicio.

Los servicios basados en la nube pública ofrecen ventajas notables, incluyendo una reducción de costos operativos, ya que las organizaciones pagan únicamente por los recursos que consumen. Además, simplifican la gestión al eliminar la necesidad de mantener infraestructuras físicas propias. Estos servicios son inherentemente escalables, permitiendo a las empresas ajustar los recursos a sus necesidades fluctuantes. La implementación de soluciones y aplicaciones es rápida y eficiente, y la alta disponibilidad está garantizada por una red distribuida de servidores situados en diversas localizaciones.

8.1.2.2. Nube Privada

La nube privada se configura como una variante específica de la computación en la nube, diseñada dentro de una red privada segura, destinada exclusivamente a clientes seleccionados, a diferencia de la nube pública accesible a un público más amplio. Esta modalidad proporciona un entorno controlado y personalizable, donde las empresas pueden beneficiarse de la escalabilidad y eficiencia operativa similares a las de la nube pública, pero con un mayor grado de personalización y control sobre su infraestructura dedicada.

Una característica distintiva de la nube privada es su enfoque en la privacidad y seguridad de datos. Al operar en un entorno cerrado y ser utilizada únicamente por clientes específicos, se pueden implementar medidas de seguridad más estrictas, como firewalls avanzados y redes privadas virtuales (VPN), asegurando así el aislamiento total de los recursos y la protección de las comunicaciones. A pesar de estas ventajas, la nube privada conlleva costes asociados con el mantenimiento, la actualización y la gestión de centros de datos tradicionales propios, lo que puede representar una inversión significativa.

Ejemplos notables de soluciones de nube privada incluyen Azure Stack de Microsoft, VMware Cloud Foundation y Red Hat OpenShift, cada uno ofreciendo distintas capacidades y niveles de personalización para atender las necesidades específicas de las organizaciones.

8.1.2.3. Nube Híbrida

La nube híbrida se define como una solución de servicios de nube que combina y aprovecha tanto las nubes privadas como las públicas. Esta modalidad ofrece una flexibilidad significativa, permitiendo a las organizaciones extender y adaptar su infraestructura para gestionar servidores dedicados y recursos de nube desde una única interfaz de control. Los servicios proporcionados en un entorno de nube híbrida están diseñados para sacar partido de las ventajas específicas de cada tipo de nube, optimizando la eficiencia, el ahorro de costos y la escalabilidad.

Los entornos de nube híbrida son particularmente adecuados para aplicaciones y proyectos que requieren infraestructuras flexibles y adaptativas. Por ejemplo, una empresa podría optar por utilizar una nube pública para manejar aspectos menos críticos, como la interfaz de usuario de un sitio web, mientras que los datos sensibles o críticos, como las bases de datos, se mantienen en una nube privada. Este enfoque permite una gestión más eficiente de los recursos, asegurando al mismo tiempo la seguridad y el control de los componentes críticos del sistema.

8.1.2.4. Nube Comunitaria

La nube comunitaria, enfocada en servir a un grupo de organizaciones con intereses y objetivos alineados, representa una solución de nube que fusiona los beneficios de las nubes

privadas y públicas, adaptándolos a un contexto colaborativo específico. Las entidades que forman parte de esta comunidad suelen tener requisitos comunes en términos de políticas de seguridad, cumplimiento normativo, y objetivos de misión, lo que hace que la nube comunitaria sea una opción ideal para abordar estas necesidades colectivas de manera eficiente.

Este modelo de nube ofrece una plataforma compartida que puede ser administrada tanto por las organizaciones participantes como por terceros especializados, ofreciendo así flexibilidad en la gestión y operación. La ventaja principal de la nube comunitaria radica en su capacidad para proporcionar una solución de nube altamente personalizada que se alinea estrechamente con los requisitos específicos del grupo, al mismo tiempo que se beneficia de la economía de escala y la eficiencia operativa. Esta configuración no solo facilita la colaboración y el intercambio de recursos entre las entidades miembros, sino que también permite una gestión de costos más efectiva, compartiendo la infraestructura y los gastos operativos relacionados. En consecuencia, la nube comunitaria se establece como una solución pragmática y estratégica para organizaciones con objetivos y desafíos comunes en el ámbito de la tecnología de la información.

8.1.3. Infraestructura de la Nube

La infraestructura de la nube es un término que describe los componentes necesarios para la computación en la nube, que incluyen hardware, recursos abstraídos, almacenamiento y recursos de red. Esta infraestructura es vista como las herramientas necesarias para construir una nube. Para alojar servicios y aplicaciones en la nube, se necesita esta infraestructura

8.1.4. Arquitectura de la Nube

La arquitectura de la nube, un término que encapsula la integración y la configuración de diversas tecnologías para formar entornos de computación en la nube, es una estructura compleja y multifacética. Esta arquitectura no se limita simplemente a la disposición física del hardware y los recursos virtuales, sino que también abarca la forma en que se interconectan y coordinan múltiples componentes esenciales, como las redes, los sistemas operativos, el middleware, las soluciones de automatización, la gestión de recursos, los contenedores y otros elementos cruciales para el funcionamiento eficiente de la nube.

En el corazón de la arquitectura de la nube se encuentra la idea de crear un entorno que sea a la vez escalable y eficiente, capaz de adaptarse a las cambiantes necesidades de los usuarios y las organizaciones. Esta arquitectura es fundamental para el diseño y la implementación de soluciones en la nube, ya que dicta cómo los recursos son asignados, gestionados y optimizados. Además, la arquitectura de la nube juega un papel crucial en la seguridad, el rendimiento y la disponibilidad de los servicios en la nube, siendo un componente clave en la estrategia de TI de las organizaciones que buscan aprovechar las ventajas de la computación en la nube.

8.1.5. Regiones en la nube

Las regiones en la nube son áreas geográficamente distintas que representan la presencia física de la infraestructura del proveedor de servicios en la nube. Cada región está configurada para funcionar como un centro independiente de recursos y servicios de computación en la nube, contribuyendo a la eficiencia y eficacia de la entrega de estos

servicios. La distribución geográfica de las regiones en la nube es un aspecto crucial en la arquitectura de la nube, ya que tiene un impacto significativo en varios factores clave como la latencia, el rendimiento, la redundancia de datos y el cumplimiento de las regulaciones locales.

Al seleccionar una región en la nube, las organizaciones pueden optimizar el rendimiento de sus aplicaciones basándose en la proximidad geográfica a los usuarios finales. Esto es especialmente relevante para aplicaciones que requieren tiempos de respuesta rápidos o que manejan grandes cantidades de datos. Además, la elección de la región puede ser determinada por consideraciones legales y de cumplimiento normativo, ya que diferentes regiones pueden estar sujetas a distintas leyes de protección de datos y privacidad. Así, las regiones en la nube no solo facilitan un rendimiento óptimo y una mejor experiencia del usuario, sino que también ayudan a las empresas a adherirse a las regulaciones locales y a gestionar mejor los riesgos relacionados con la seguridad de los datos.

8.1.5.1. Zona de Disponibilidad

Una zona de disponibilidad, en el contexto de la computación en la nube, es una unidad operativa compuesta por varios centros de datos interconectados, cada uno dotado de infraestructuras independientes en términos de suministro energético, sistemas de refrigeración y redes. Estos centros de datos se localizan en instalaciones físicamente distintas para garantizar redundancia y alta disponibilidad. Cada región de la nube puede contener múltiples zonas de disponibilidad, pero estas zonas son exclusivas de su región y no se extienden o comparten entre diferentes regiones.

La interconexión de centros de datos dentro de una zona de disponibilidad se realiza mediante enlaces de red privada. Estos enlaces son redundantes, de alta velocidad y baja latencia, asegurando así una comunicación eficiente y segura entre los centros de datos. Además, todas las zonas de disponibilidad dentro de una región específica están interconectadas a través de enlaces de red similares, lo que proporciona una base robusta para la continuidad del servicio y la resistencia ante fallos.

El diseño de zonas de disponibilidad es fundamental para ofrecer un alto nivel de fiabilidad y continuidad operativa en servicios en la nube. Al separar físicamente los centros de datos y dotarlos de infraestructuras independientes, las zonas de disponibilidad minimizan el riesgo de interrupciones del servicio causadas por fallos en infraestructuras físicas o desastres naturales. Este enfoque asegura que incluso si un centro de datos en una zona de disponibilidad experimenta un problema, los otros centros de datos dentro de la misma zona o en otras zonas de la misma región pueden continuar operando sin interrupciones significativas, garantizando así la disponibilidad constante de los servicios en la nube.

8.1.5.2. Dominio

Un dominio en el contexto de la computación en la nube se define como una subdivisión específica dentro de una región de la nube, distinguida por su infraestructura y servicios únicos. Estos dominios son esenciales para diferenciar y organizar diversos entornos operativos o zonas de servicio dentro de la misma región geográfica de la nube. Cada dominio posee una configuración particular y un conjunto de recursos diseñados para atender a requisitos específicos, ya sea para satisfacer necesidades de servicio únicas o para cumplir con determinados estándares regulatorios.

La implementación de dominios permite a los proveedores de servicios en la nube ofrecer una variedad de servicios altamente especializados bajo el mismo paraguas regional, facilitando así soluciones personalizadas para diferentes clientes o aplicaciones. Esta segregación no solo es crucial para la personalización de los servicios, sino que también juega un papel importante en la gestión y el aislamiento efectivo de estos servicios. Como resultado, se mejora la seguridad y la fiabilidad de los servicios en la nube, ofreciendo a los usuarios finales un entorno más seguro y estable para sus operaciones en la nube. La existencia de dominios dentro de las regiones de la nube refleja la creciente necesidad de servicios de computación en la nube adaptados y seguros, capaces de satisfacer una amplia gama de requerimientos operativos y normativos.

8.1.6. Cluster

Un clúster de computadoras es un grupo de máquinas, conocidas como nodos, que están interconectadas y diseñadas para trabajar colectivamente, funcionando efectivamente como una sola entidad computacional de mayor potencia. A diferencia de las redes de computadoras tradicionales donde cada nodo puede realizar tareas diferentes, en un clúster, todos los nodos se dedican a ejecutar la misma tarea o conjunto de tareas, colaborando entre sí para aumentar la capacidad de procesamiento y la eficiencia.

Los nodos dentro de un clúster están generalmente vinculados mediante redes de área local (LAN) de alta velocidad, lo que facilita una comunicación rápida y eficiente entre ellos. Esta configuración permite un procesamiento distribuido eficaz, donde las tareas se dividen y se ejecutan paralelamente en diferentes nodos. Cada nodo en un clúster opera con su propia

instancia del sistema operativo, aunque trabajan en conjunto para completar las tareas asignadas.

La arquitectura de un clúster ofrece varias ventajas, como una mayor potencia de procesamiento, confiabilidad mejorada y una mayor redundancia. Si un nodo en el clúster falla, otros nodos pueden asumir su carga de trabajo, minimizando así las interrupciones en el procesamiento. Los clústers son especialmente útiles en entornos que requieren un gran volumen de cálculos y procesamiento de datos, como la investigación científica, el análisis de grandes datos y las aplicaciones empresariales críticas. Su capacidad para proporcionar un rendimiento informático de alta capacidad de manera rentable los hace indispensables en muchas aplicaciones de computación modernas.

8.2. Virtualización

La virtualización es una tecnología avanzada que permite crear versiones virtuales de recursos físicos como servidores, dispositivos de almacenamiento, redes y otros elementos de hardware. Esta tecnología utiliza software especializado para emular las funciones del hardware físico, posibilitando así que varias máquinas virtuales operen simultáneamente en un único servidor físico.

El propósito principal de la virtualización es mejorar la eficiencia y la utilización de los recursos de hardware existentes. Permite a las empresas maximizar el rendimiento de sus inversiones en hardware, ya que pueden ejecutar múltiples sistemas y aplicaciones en un solo equipo físico, reduciendo la necesidad de adquirir hardware adicional. Además, la virtualización es un componente clave en los servicios de computación en la nube, ya que facilita la gestión flexible y eficiente de la infraestructura de TI. A través de la virtualización,

las organizaciones pueden crear y desplegar rápidamente entornos de TI virtualizados, lo que les permite responder con agilidad a las cambiantes necesidades empresariales.

Esta tecnología no solo proporciona beneficios económicos, sino que también contribuye a una mayor agilidad y escalabilidad en la gestión de TI. La virtualización también juega un papel crucial en la optimización de la gestión de centros de datos, la reducción del consumo de energía y la disminución del espacio físico necesario para el hardware, lo que la convierte en una solución tecnológica esencial para las empresas modernas que buscan eficiencia y sostenibilidad en sus operaciones de TI.

8.2.1. Hipervisor

Los hipervisores son componentes críticos en la tecnología de virtualización, especialmente en el contexto de la virtualización de servidores, una piedra angular de la computación en la nube. Un hipervisor es un tipo de software diseñado para crear y gestionar máquinas virtuales (MV) utilizando los recursos de un servidor físico. Actúa como una capa de abstracción entre el hardware físico y las máquinas virtuales, lo que permite un uso más eficiente de los recursos físicos, simplifica el mantenimiento y las operaciones, y contribuye a una reducción de costos.

Existen dos categorías principales de hipervisores, cada una con sus propias características y aplicaciones:

Los hipervisores son componentes críticos en la tecnología de virtualización, especialmente en el contexto de la virtualización de servidores, una piedra angular de la computación en la nube. Un hipervisor es un tipo de software diseñado para crear y gestionar

máquinas virtuales (MV) utilizando los recursos de un servidor físico. Actúa como una capa de abstracción entre el hardware físico y las máquinas virtuales, lo que permite un uso más eficiente de los recursos físicos, simplifica el mantenimiento y las operaciones, y contribuye a una reducción de costos.

Existen dos categorías principales de hipervisores, cada una con sus propias características y aplicaciones:

Hipervisores de Tipo 1:

Estos se instalan directamente sobre el hardware del servidor, conocidos comúnmente como hipervisores "bare metal". Este tipo de hipervisor ofrece eficiencia y seguridad mejoradas al minimizar la superficie de ataque. Son la opción preferida en entornos empresariales por su seguridad robusta, escalabilidad, estabilidad y alto rendimiento.

Ejemplos de hipervisores de Tipo 1 incluyen Nutanix AHV, VMware ESXi, Microsoft Hyper-V para entornos de servidor, Citrix Hypervisor y Kernel-based Virtual Machine (KVM).

Hipervisores de Tipo 2:

Estos funcionan como aplicaciones dentro de un sistema operativo existente, conocidos como hipervisores "alojados". Son fáciles de instalar y convenientes para usuarios individuales que requieren ejecutar múltiples sistemas operativos en una sola máquina.

A pesar de su facilidad de uso, los hipervisores de Tipo 2 pueden presentar una latencia mayor y riesgos de seguridad adicionales en comparación con los hipervisores de Tipo 1, lo que los hace menos idóneos para entornos basados en servidores.

Ejemplos incluyen Oracle VirtualBox, VMware Workstation, Parallels Desktop y Microsoft Hyper-V en entornos de escritorio.

En resumen, los hipervisores son esenciales para la virtualización efectiva de servidores, proporcionando la base necesaria para la creación y gestión de múltiples entornos virtuales en una sola infraestructura física. Su capacidad para optimizar el uso de recursos, junto con sus características de seguridad y estabilidad, los convierte en elementos fundamentales en la arquitectura de la computación en la nube moderna.

8.2.1.1. Kernel-based Virtual Machine

Kernel-based Virtual Machine (KVM) es una solución de virtualización de código abierto que se integra directamente en el sistema operativo Linux. Esta tecnología convierte al núcleo de Linux en un hipervisor de tipo 1, permitiendo que un servidor Linux actúe como una plataforma de alojamiento para múltiples entornos virtuales independientes, conocidos como máquinas virtuales (VM) o invitados.

Lo que distingue a KVM de otras soluciones de virtualización es su integración nativa con Linux, lo que le permite aprovechar directamente las características de seguridad, rendimiento y estabilidad del sistema operativo. Como hipervisor de tipo 1, KVM puede gestionar eficientemente los recursos del sistema y proporcionar un rendimiento optimizado para las máquinas virtuales que aloja.

KVM es una opción poderosa para la virtualización debido a su naturaleza open source y su compatibilidad con la amplia gama de herramientas y características que ofrece Linux. Esto incluye el soporte para una variedad de sistemas operativos invitados, redes virtualizadas, almacenamiento y otras funciones avanzadas de virtualización. Su capacidad

para operar eficientemente en un entorno Linux lo convierte en una solución ideal para organizaciones que buscan una plataforma de virtualización flexible, robusta y rentable.

8.3. OpenStack como Plataforma de Nube

OpenStack, según Rackspace en 2023, se define como una plataforma de software de código abierto que se utiliza tanto para la creación de nubes privadas como públicas. Esta plataforma funciona bajo el modelo de Infraestructura como Servicio (IaaS), brindando a las empresas la flexibilidad para expandir y mejorar sus recursos de servidores, redes y almacenamiento de manera sencilla y eficiente dentro de su infraestructura de nube.

Lanzado inicialmente en 2010 por Rackspace y la NASA, OpenStack es ahora supervisado y mantenido por la OpenStack Foundation, una entidad sin fines de lucro dedicada a su desarrollo y evolución. La plataforma se ha destacado por su robustez y escalabilidad, lo que ha llevado a su adopción por parte de grandes corporaciones y entidades. Ejemplos notables incluyen Walmart, que utiliza OpenStack para su plataforma de comercio electrónico, y el CERN, donde se implementa para gestionar su nube privada, reconocida como una de las mayores implementaciones de OpenStack en el mundo.

La popularidad y el éxito de OpenStack se atribuyen a su naturaleza de código abierto, lo que permite una personalización y adaptabilidad significativas para satisfacer las necesidades específicas de cada organización. Además, su comunidad activa y en constante crecimiento contribuye a la mejora continua de la plataforma, asegurando que siga siendo una solución de vanguardia para la gestión de infraestructuras en la nube.

8.3.1. Componentes Principales de OpenStack

8.3.1.1. Nova (Compute)

Nova, el componente de Compute en OpenStack, es esencial para la gestión y provisión de instancias en el entorno de OpenStack, donde una instancia se refiere a una máquina virtual aprovisionada en un nodo de computación físico. Este proyecto es responsable de administrar y escalar instancias a través de múltiples hosts físicos, permitiendo así a los usuarios de OpenStack manejar de forma eficiente sus recursos computacionales.

Uno de los aspectos destacados de Nova es su capacidad para proporcionar un método unificado y simplificado de acceso a las interfaces gráficas y consolas de todas las instancias, sin importar la tecnología de virtualización subyacente o los sistemas operativos que ejecuten las instancias. Esta característica hace de Nova una herramienta poderosa y flexible para la administración de entornos virtuales, ya que abstrae la complejidad asociada con la gestión de diversas tecnologías de virtualización y sistemas operativos.

La funcionalidad de Nova en OpenStack, por lo tanto, no solo facilita la gestión de recursos de computación a gran escala, sino que también garantiza una experiencia de usuario coherente y accesible. Esto es particularmente valioso en entornos de nube que requieren una alta disponibilidad, escalabilidad y eficiencia en la administración de sus recursos computacionales. La integración y el funcionamiento eficaz de Nova dentro del ecosistema de OpenStack subraya su importancia como un componente clave para la realización de operaciones de computación en la nube.

8.3.1.2. Neutron (Networking)

Neutron, el componente de Networking en OpenStack, juega un papel crucial al proporcionar y gestionar la conectividad de red tanto dentro de la nube de OpenStack como entre las instancias en la nube y el mundo exterior. Este servicio es fundamental para asegurar que las instancias y los servicios dentro de la nube de OpenStack se comuniquen eficientemente entre sí, así como con recursos externos.

Un aspecto clave de la red en OpenStack es la diferenciación entre las redes de proveedores y las redes de inquilinos. Las redes de proveedores son segmentos de la infraestructura de red física definidos dentro de OpenStack, cuya creación está reservada exclusivamente para los administradores. Estas redes reflejan la configuración de la infraestructura de red física subyacente y son esenciales para vincular la nube de OpenStack con la red física real. Por otro lado, las redes de inquilinos son creadas y utilizadas por los usuarios finales de la nube para segmentar y gestionar su tráfico de red virtual dentro de OpenStack.

La habilidad de Neutron para manejar estas dos categorías de redes es vital para la funcionalidad y la eficiencia de la red en OpenStack. Al permitir que los administradores mapeen y gestionen las conexiones entre las redes virtuales y las redes físicas, Neutron asegura que la infraestructura de la nube de OpenStack se integre y funcione armónicamente con la infraestructura de red existente. Esto no solo facilita una gestión eficiente del tráfico de red y la conectividad, sino que también permite una mayor flexibilidad y personalización en la configuración de la red para adaptarse a diversas necesidades y requerimientos de los usuarios.

8.3.1.3. Cinder (Block Storage)

Cinder, el servicio de almacenamiento de bloques de OpenStack, es un componente esencial para el manejo de almacenamiento adicional en entornos de nube. Cuando se trabaja con instancias de computación en OpenStack, a menudo surge la necesidad de contar con almacenamiento adicional que pueda ser montado y utilizado por estas instancias. Cinder se encarga de proporcionar y gestionar este tipo de almacenamiento, conocido como almacenamiento en bloque.

El almacenamiento en bloque se refiere a unidades de almacenamiento individuales, conocidas como 'bloques', que se pueden adjuntar a las instancias como volúmenes independientes. Cada bloque es tratado por la instancia como un disco duro externo, proporcionando almacenamiento flexible y ampliable que es esencial para ciertas aplicaciones y necesidades de datos.

Cinder facilita la creación, el aprovisionamiento y la gestión de estos bloques de almacenamiento, permitiendo a los usuarios de OpenStack acceder y montar fácilmente los volúmenes necesarios en sus instancias. Esta capacidad de Cinder para integrar sin problemas el almacenamiento adicional en el entorno de nube es clave para ofrecer una experiencia de usuario fluida y eficiente en OpenStack, especialmente en aplicaciones que requieren una gran cantidad de datos o un almacenamiento altamente disponible y escalable.

8.3.1.4. Swift (Object Storage)

Swift, el servicio de almacenamiento de objetos de OpenStack, es un componente crucial que ofrece una solución de almacenamiento escalable y duradero dentro del

ecosistema de OpenStack. Este servicio está diseñado para almacenar y gestionar grandes cantidades de datos no estructurados, como archivos, imágenes y backups, a través de una interfaz de API REST HTTP.

La API REST de Swift permite realizar operaciones comunes sobre objetos almacenados utilizando el protocolo HTTP estándar, lo que facilita la integración y el acceso a los datos almacenados. Las operaciones comunes incluyen la creación, recuperación, modificación y eliminación de objetos, así como la gestión de metadatos asociados y la organización de los datos en contenedores.

Lo que distingue a Swift de otras soluciones de almacenamiento es su enfoque en la simplicidad y la escalabilidad, junto con su capacidad para manejar grandes volúmenes de datos de manera eficiente. Esto lo hace ideal para escenarios donde se requiere un almacenamiento masivo y duradero, como el almacenamiento de backups, archivos multimedia y otros tipos de contenido digital. Además, la arquitectura de Swift está optimizada para garantizar la alta disponibilidad y durabilidad de los datos, lo que es esencial en aplicaciones críticas donde la pérdida de datos es inaceptable.

8.3.1.5. Glance (Image Service)

Glance, conocido como el servicio de imágenes en OpenStack, es una pieza fundamental del ecosistema de OpenStack que se centra en la gestión de imágenes de máquinas virtuales. Este servicio permite a los usuarios descubrir, registrar y acceder a imágenes de máquinas virtuales, que son esenciales para la creación y despliegue de instancias en un entorno de nube.

La funcionalidad principal de Glance incluye una API REST que facilita la interacción con los metadatos de las imágenes de máquinas virtuales y permite recuperar las imágenes reales. Esta API es una herramienta crucial para los usuarios de OpenStack, ya que proporciona un método accesible y eficiente para gestionar imágenes de máquinas virtuales.

Glance ofrece flexibilidad en términos de almacenamiento de estas imágenes. Las imágenes pueden ser almacenadas en una variedad de ubicaciones, desde sistemas de archivos sencillos hasta soluciones más complejas de almacenamiento de objetos, como el servicio OpenStack Object Storage (Swift). Esta versatilidad asegura que Glance pueda adaptarse a diferentes necesidades y configuraciones de infraestructura, permitiendo a los usuarios almacenar y acceder a las imágenes de las máquinas virtuales de manera eficiente y segura.

8.3.1.6. Keystone (Identity Service)

Keystone, el servicio de identidad en OpenStack, desempeña un rol fundamental en la gestión de autenticación y autorización dentro del ecosistema de OpenStack. Su función principal es procesar y validar solicitudes de autenticación de usuarios y servicios, y luego emitir un "token de autenticación". Este token actúa como una credencial que permite a los usuarios y servicios autenticarse frente a las distintas API de OpenStack y acceder a otros servicios en la infraestructura de la nube.

El sistema de tokens de Keystone es un componente crítico para garantizar la seguridad y el control de acceso en OpenStack. Permite un manejo centralizado de identidades y políticas de acceso, asegurando que solo los usuarios y servicios autorizados puedan acceder a recursos y funciones específicos en la infraestructura de la nube. Esto no

solo facilita una administración de acceso segura y eficiente, sino que también simplifica el proceso de autenticación para los usuarios finales.

Además, Keystone proporciona una interfaz API para la administración de usuarios, roles, permisos y servicios, lo que lo convierte en una herramienta integral para la administración de identidades y acceso en OpenStack. Su capacidad para integrarse con otros servicios de OpenStack y con sistemas de identidad externos lo hace esencial para operar y gestionar de manera efectiva y segura una infraestructura de nube OpenStack.

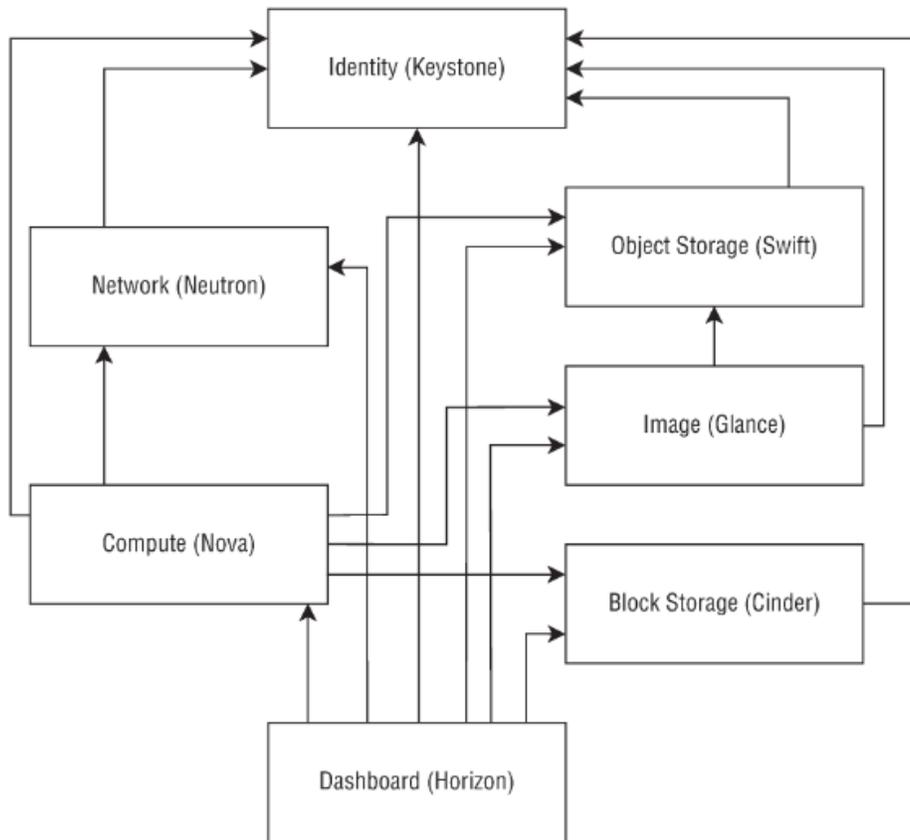


Ilustración 4: Como se comunican los servicios de openstack

8.3.1.7. Horizon (Dashboard)

Horizon, el proyecto de panel o dashboard en OpenStack, es una interfaz web crucial para la interacción con los diversos componentes y servicios de la plataforma OpenStack. Construida sobre el marco Django y utilizando las distintas API de OpenStack, Horizon ofrece una interfaz gráfica de usuario (GUI) intuitiva y fácil de usar que facilita la administración y el monitoreo de los recursos y servicios en la nube de OpenStack.

La GUI de Horizon está diseñada para ser accesible tanto para usuarios novatos como para administradores experimentados, proporcionando una visión clara y organizada de los recursos de la nube, como instancias, redes, almacenamiento y otros servicios críticos. Esta interfaz permite a los usuarios realizar una variedad de tareas administrativas de manera sencilla y eficiente, sin la necesidad de interactuar directamente con las API de OpenStack o utilizar la línea de comandos.

El dashboard de Horizon se ha convertido en una herramienta esencial para la gestión de infraestructuras en la nube OpenStack, ya que ofrece un punto de acceso centralizado y coherente para la gestión de la nube. Su diseño y funcionalidad facilitan el aprendizaje y la adopción de OpenStack, al tiempo que proporcionan una plataforma poderosa para el manejo eficiente de operaciones complejas en la nube. En resumen, Horizon juega un papel vital en la mejora de la experiencia del usuario y la administración eficiente de la nube OpenStack.

8.3.2. Mensajería y colas

La mensajería en un entorno de nube se refiere al intercambio de información a través de mensajes entre diferentes componentes de una aplicación o entre distintas aplicaciones.

Esta comunicación se realiza mediante protocolos de mensajería y APIs que permiten a las aplicaciones enviar y recibir datos de forma confiable y segura.

Las colas son estructuras de datos utilizadas en procesos de mensajería para almacenar temporalmente mensajes hasta que pueden ser procesados. En un sistema de colas, los mensajes son enviados por un productor y almacenados en la cola hasta que un consumidor los procesa, permitiendo la comunicación asíncrona entre ellos. (Dawood, 2022)

La mensajería y las colas proporcionan comunicación y coordinación entre aplicaciones distribuidas, simplificando la codificación de aplicaciones desacopladas y mejorando la fiabilidad, el rendimiento y la escalabilidad.

Permiten a los servidores web responder rápidamente a las solicitudes en lugar de tener que realizar procedimientos que consumen muchos recursos en el momento, lo que podría retrasar el tiempo de respuesta. Además, son útiles para distribuir mensajes a múltiples consumidores o equilibrar cargas entre trabajadores.

8.3.2.1. RabbitMQ

En el ecosistema de OpenStack, RabbitMQ desempeña un papel crucial como un sistema de gestión de colas de mensajes o intermediario de mensajes. Este software es esencial para manejar la comunicación entre diversos componentes y servicios dentro de una infraestructura de nube OpenStack. Su principal función es actuar como un intermediario en el flujo de mensajes, asegurando que la información se transmita de manera eficiente y confiable entre diferentes partes del sistema.

RabbitMQ facilita la creación de colas, a las cuales las aplicaciones se pueden conectar para enviar y recibir mensajes. Esta arquitectura de mensajería basada en colas

permite a las aplicaciones y servicios de OpenStack comunicarse de forma asincrónica, lo que significa que pueden intercambiar mensajes sin necesidad de una interacción directa o simultánea. Esta capacidad es particularmente valiosa en entornos de nube complejos, donde los componentes pueden estar distribuidos y necesitan coordinarse eficientemente para realizar tareas o procesos.

La implementación de RabbitMQ en OpenStack contribuye significativamente a la escalabilidad y eficiencia de la plataforma, permitiendo el manejo de grandes volúmenes de mensajes y la coordinación fluida entre servicios y componentes. Al proveer un mecanismo confiable y escalable para la gestión de la comunicación, RabbitMQ es un componente integral que refuerza la robustez y la capacidad operativa de la infraestructura de nube OpenStack.

8.4. Orquestación

La orquestación, en el ámbito de la tecnología de la información, se refiere a la coordinación y gestión sistemática de diversos sistemas informáticos, aplicaciones y servicios. Su objetivo principal es integrar y automatizar múltiples tareas para formar un flujo de trabajo o proceso más complejo y cohesivo. Esta práctica es fundamental en entornos donde se requiere que distintas operaciones y procesos interactúen y funcionen de manera armónica y eficiente.

En la orquestación, las tareas individuales, que pueden ser diversas y ejecutadas por sistemas diferentes, son secuenciadas y coordinadas para lograr un objetivo específico o para completar un proceso integral. Esta metodología es especialmente útil en entornos de

computación en la nube, donde la integración y automatización de procesos a través de diferentes servicios y aplicaciones son cruciales para la eficiencia operativa.

Además de mejorar la eficiencia, la orquestación permite una gestión más efectiva de los recursos, reduciendo la redundancia y optimizando el uso de la infraestructura existente. Al automatizar y coordinar tareas, los sistemas pueden funcionar con mayor independencia y menor intervención humana, lo que reduce el margen de error y aumenta la fiabilidad de los procesos empresariales. En resumen, la orquestación es un componente clave en la gestión de sistemas y operaciones complejas, proporcionando una estructura esencial para la ejecución eficaz y coherente de tareas y procesos en entornos informáticos modernos.

8.4.1. Orquestación del almacenamiento de datos en la nube

La orquestación de datos es “El proceso de recopilar, transformar, integrar y administrar datos de múltiples fuentes de manera eficiente se conoce como orquestación de datos.” (Pathak, 2023). La orquestación de datos tiene como objetivo principal optimizar los datos de diferentes fuentes de manera efectiva y eficiente para que las empresas puedan aprovechar al máximo su potencial. Este proceso es fundamental en el mundo moderno basado en datos ya que proporciona información clara sobre el negocio, clientes, mercado y competidores, lo que ayuda a tomar decisiones informadas y lograr los resultados deseados. En resumen, la orquestación de datos es como un conductor que lee y recopila datos de múltiples fuentes para obtener una visión general del rendimiento de la empresa

8.4.2. Almacenamiento en Caché

El almacenamiento en caché en un entorno de nube es una estrategia crucial para optimizar el acceso y el procesamiento de datos. Esta técnica consiste en duplicar los datos que se acceden con frecuencia y almacenarlos en el sistema de almacenamiento más rápido disponible, idealmente situado lo más cerca posible de la aplicación que los utiliza. Al hacerlo, se acelera significativamente el tiempo de acceso a estos datos, mejorando así el rendimiento de las aplicaciones.

Esta forma de almacenamiento en caché es especialmente beneficiosa en operaciones que requieren una alta velocidad de procesamiento de datos. Al almacenar datos en caché, las aplicaciones pueden acceder y procesar la información mucho más rápido en comparación con la recuperación de datos desde el almacenamiento regular. Esto es especialmente relevante para tareas que requieren lecturas repetitivas de los mismos datos o para sistemas que necesitan responder rápidamente a las solicitudes de los usuarios.

En el ámbito de los servicios de nube, existen varias soluciones de almacenamiento en caché disponibles que ayudan a implementar esta estrategia de manera efectiva. Ejemplos notables incluyen Azure Cache para Redis de Microsoft, Amazon ElastiCache de Amazon Web Services y Google App Engine Memcache de Google Cloud. Estos servicios proporcionan plataformas robustas y escalables para implementar el almacenamiento en caché, ayudando a las aplicaciones a funcionar de manera más eficiente y confiable.

El uso del almacenamiento en caché en la nube es un aspecto fundamental para mejorar la eficiencia y el rendimiento de las aplicaciones en entornos de nube. Al reducir la dependencia del almacenamiento tradicional y optimizar el acceso a los datos, las

organizaciones pueden garantizar un rendimiento superior de sus aplicaciones y una mejor experiencia para los usuarios finales.

8.4.2.1. Memcached

Memcached es un sistema de almacenamiento en memoria ampliamente utilizado para optimizar el rendimiento de aplicaciones web dinámicas que interactúan con bases de datos. Este sistema de almacenamiento en memoria se utiliza para cachear datos y objetos frecuentemente accedidos, lo que permite a las aplicaciones recuperar esta información de manera rápida y eficiente, evitando así el acceso repetido al disco duro para obtener los mismos datos.

La principal ventaja de Memcached es su capacidad para reducir la carga en los sistemas de back-end y disminuir significativamente la latencia en la recuperación de datos. Al almacenar datos en la memoria RAM, Memcached facilita un acceso mucho más rápido a estos en comparación con la lectura desde un sistema de almacenamiento basado en disco. Esto resulta en una mejora notable en la velocidad de respuesta de las aplicaciones, especialmente en aquellas que requieren el procesamiento de grandes volúmenes de datos o que manejan un número alto de consultas a bases de datos.

Sitios web de alto tráfico como YouTube, Facebook, Twitter y Wikipedia han implementado Memcached para aprovechar sus beneficios. Gracias a esta solución de código abierto, estos sitios pueden ofrecer a los usuarios una experiencia de navegación más fluida y rápida, mejorando significativamente el rendimiento y la eficiencia de sus plataformas en línea.

8.5. Control y accesos en la nube

8.5.1. Usuario

es una persona o entidad que accede y utiliza los recursos y servicios proporcionados por la infraestructura de nube. Los usuarios pueden variar desde individuos que utilizan aplicaciones basadas en la nube para tareas personales o laborales, hasta organizaciones que implementan soluciones de nube para operaciones empresariales.

8.5.2. Rol

Rol se refiere a un conjunto de permisos y políticas que definen lo que un usuario o grupo de usuarios puede hacer dentro del sistema de nube. Los roles son fundamentales para la gestión de la seguridad y el control de accesos en entornos de nube, ya que permiten a los administradores asignar y gestionar los privilegios de los usuarios de manera eficiente y segura.

Los roles pueden incluir permisos para realizar acciones específicas como el despliegue de recursos, la administración de bases de datos, la configuración de redes y la gestión de aplicaciones. También pueden definir límites en el uso de recursos y acceso a datos sensibles. Esto ayuda a prevenir el acceso no autorizado y garantiza que cada usuario tenga los permisos necesarios para realizar su trabajo sin exponer el sistema a riesgos de seguridad innecesarios.

8.5.3. Usuarios Federados

El patrón de identidad federada es utilizado en la nube para permitir que la autenticación de los usuarios sea manejada por un proveedor de identidad externo. Este patrón es útil cuando se requiere proporcionar una experiencia de inicio de sesión único (SSO) a los usuarios, especialmente en casos donde las aplicaciones necesitan autenticar a los usuarios con múltiples socios o cuando los inquilinos de una aplicación de software como servicio (SaaS) necesitan autenticarse utilizando un proveedor de identidad adecuado.

8.5.4. Lightweight Directory Access Protocol (LDAP)

El Lightweight Directory Access Protocol (LDAP) es un protocolo ampliamente utilizado que facilita la búsqueda y gestión de información sobre organizaciones, individuos y otros recursos en una red. LDAP cumple dos funciones esenciales: por un lado, permite almacenar datos en directorios LDAP y, por otro, autentica a los usuarios para que puedan acceder a estos directorios.

LDAP proporciona un lenguaje de comunicación estándar que permite a las aplicaciones interactuar con los servicios de directorio. Estos servicios de directorio son sistemas que almacenan y organizan información sobre los recursos de la red, como usuarios, grupos, dispositivos y políticas. La capacidad de LDAP para ofrecer un acceso estructurado y eficiente a esta información lo convierte en una herramienta esencial en la administración de sistemas y redes.

Uno de los usos más comunes de LDAP es en la autenticación y autorización de usuarios. Al autenticar a los usuarios contra el directorio LDAP, las organizaciones pueden gestionar de manera centralizada el acceso a recursos y aplicaciones, mejorando así la seguridad y la eficacia operativa. Además, LDAP se utiliza para gestionar de manera eficiente la información sobre usuarios y recursos, lo que permite una administración simplificada y una mejor integración de los servicios en la red.

8.5.5. OpenLDAP

OpenLDAP es una implementación de código abierto del protocolo Lightweight Directory Access Protocol (LDAP), ampliamente reconocida y utilizada desde su lanzamiento en 1998. Este software proporciona a los administradores de TI las herramientas necesarias para construir y gestionar directorios LDAP, ofreciendo una solución eficaz para la administración de información sobre usuarios, grupos y otros recursos en una red.

Una de las características distintivas de OpenLDAP es su enfoque exclusivo en LDAP, a diferencia de otras soluciones de software LDAP que pueden incluir una gama más amplia de características y funcionalidades. Esta especialización hace de OpenLDAP una herramienta poderosa y eficiente para aquellos que buscan una implementación pura de LDAP.

OpenLDAP se maneja principalmente a través de la línea de comandos, lo que implica que su uso requiere un conocimiento técnico profundo del protocolo LDAP y de la estructura de directorios. Aunque esta característica puede representar una curva de aprendizaje más pronunciada, también ofrece una mayor flexibilidad y capacidad de personalización, lo que lo

convierte en una opción atractiva para casos de uso más técnicos y para profesionales con experiencia en TI.

Esta plataforma es particularmente útil en entornos Linux, donde su capacidad para integrarse y soportar servidores y aplicaciones basadas en Linux es muy valorada. La naturaleza de código abierto de OpenLDAP también contribuye a su popularidad, permitiendo a los usuarios modificar y adaptar el software a sus necesidades específicas. En resumen, OpenLDAP es una solución robusta y versátil para la gestión de directorios LDAP, especialmente adecuada para entornos técnicos y usuarios con un conocimiento avanzado de LDAP y administración de sistemas.

9. Metodología de la investigación

Reconociendo la investigación científica como una respuesta esencial a las interrogantes de la vida diaria, se destaca la importancia de establecer un método investigativo que sea no solo adecuado, sino también eficiente y eficaz. El objetivo es conseguir resultados que faciliten la interpretación efectiva de los fenómenos analizados.

Para el estudio en cuestión, se eligió el método de investigación cuantitativa. Esta metodología ha sido crucial para reunir información vital sobre el desarrollo de una infraestructura de nube comunitaria multi-región geodistribuida, asegurando que la investigación se alinee con el problema identificado y los objetivos propuestos.

La adopción de un enfoque cuantitativo fue fundamental para abordar los desafíos tecnológicos y empresariales en el ámbito de la computación en la nube. Este enfoque permitió la recopilación de datos específicos y el análisis estadístico, lo cual proporciona una base firme para determinar si dicha solución es viable o no.

9.1. Población y muestreo

Para esta investigación, se seleccionó cuidadosamente la población objetivo, centrándose en profesionales del departamento de TI en los Estados Unidos, con un énfasis particular en aquellos que operan en diversas instalaciones de almacenamiento de vehículos, conocidas como 'yardas'. Estas yardas desempeñan un papel crítico en el transporte de automóviles desde Estados Unidos hacia El Salvador y tienen operaciones en varios estados estadounidenses, incluyendo Florida, Texas, California, Nueva York y Georgia. Este enfoque

se eligió con el propósito de garantizar la recolección de datos altamente relevantes y directamente aplicables al caso de estudio. La metodología de selección se basó en la realización de una encuesta a desarrolladores de TI en estas yardas. Esta elección se justificó por la necesidad de obtener información detallada y específica sobre el conocimiento y uso de la tecnología OpenStack y la nube comunitaria en este grupo de interés.

9.1.1. Instrumentos y Técnicas de Recopilación de Datos:

Para llevar a cabo la recopilación de datos necesaria en esta investigación, se empleó una encuesta como la principal técnica de recopilación de datos. La encuesta se diseñó y distribuyó entre desarrolladores de TI que operan en diversas yardas en Estados Unidos, específicamente en estados como Florida, Texas, California, Nueva York y Georgia.

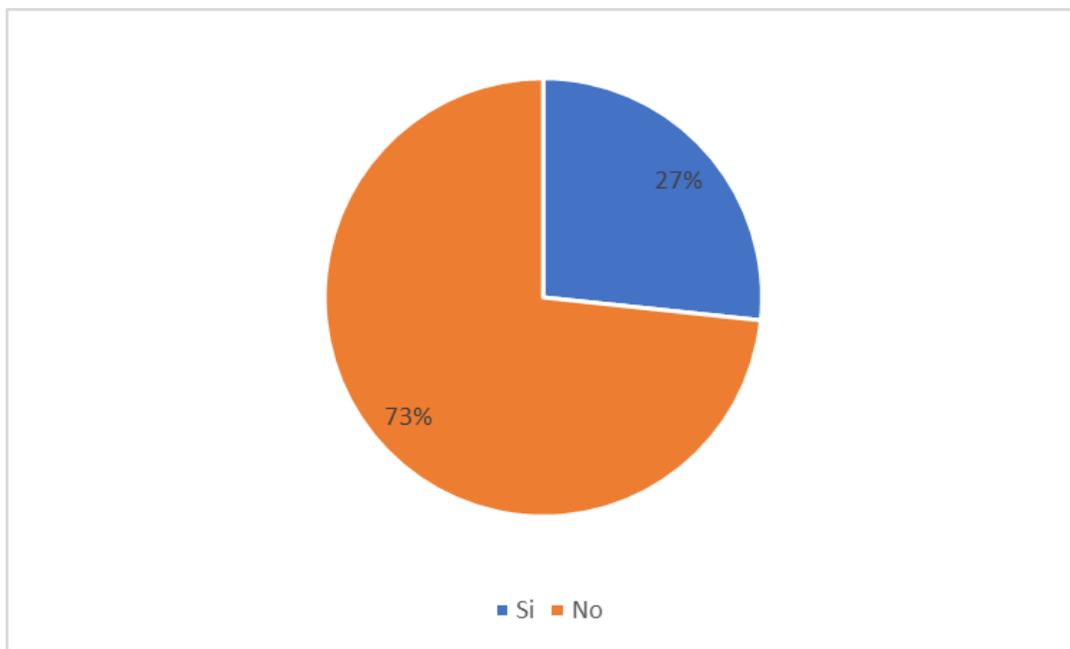
La encuesta se estructuró en torno a cuatro preguntas clave (ver 16. Anexo, 16.1 Encuesta) relacionadas con el conocimiento de la tecnología OpenStack y la nube comunitaria, así como la disposición a adoptar esta tecnología en sus proyectos. Esta técnica de recopilación de datos permitió obtener tanto datos cuantitativos como cualitativos, proporcionando una visión completa de la población objetivo y sus perspectivas en relación con el caso de estudio.

La elección de la encuesta como instrumento de recopilación de datos se basó en su capacidad para llegar a un número significativo de participantes de manera eficiente y en su flexibilidad para recopilar datos cuantitativos y cualitativos. Además, la encuesta permitió obtener información directamente aplicable al diseño y desarrollo del prototipo de nube comunitaria en el contexto de la empresa de logística de transporte de vehículos.

9.2. Resultados

9.2.1. Análisis de datos cuantitativos

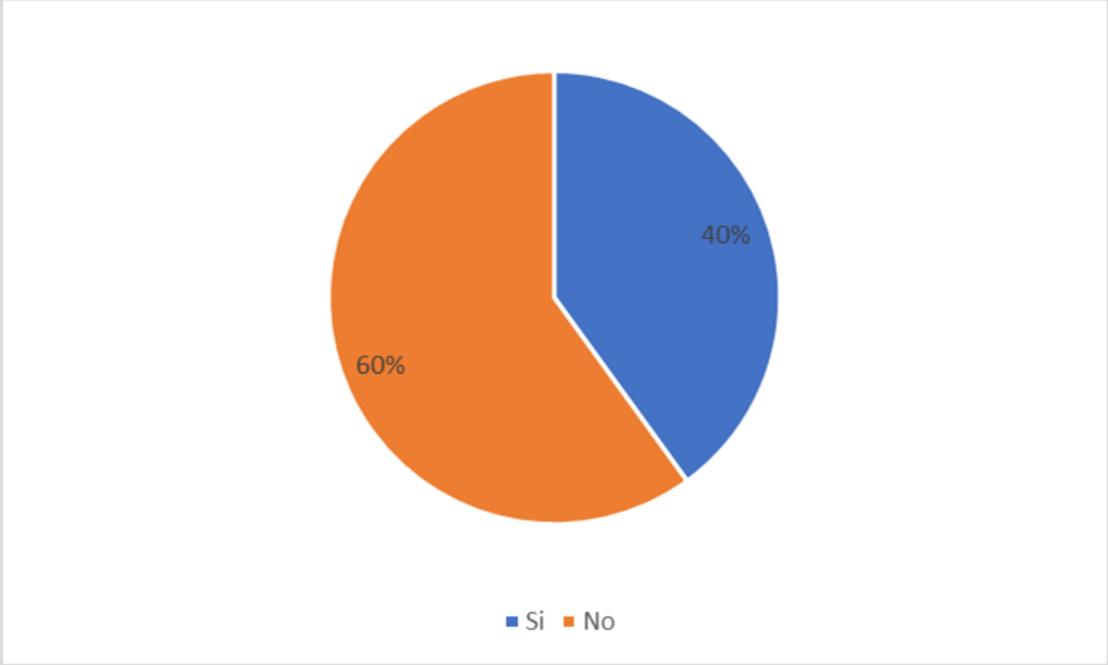
Se llevó a cabo una encuesta dirigida a desarrolladores de software en el sector de Tecnologías de la Información (TI) en Estados Unidos. La encuesta se realizó originalmente en inglés y, para fines de análisis, se tradujo. Un total de 15 trabajadores completaron esta encuesta, y sus respuestas conforman la muestra seleccionada para el análisis.



Pregunta 1: ¿Conoce la tecnología Openstack como solución para ofrecer IaaS (Infraestructura como Servicio)?

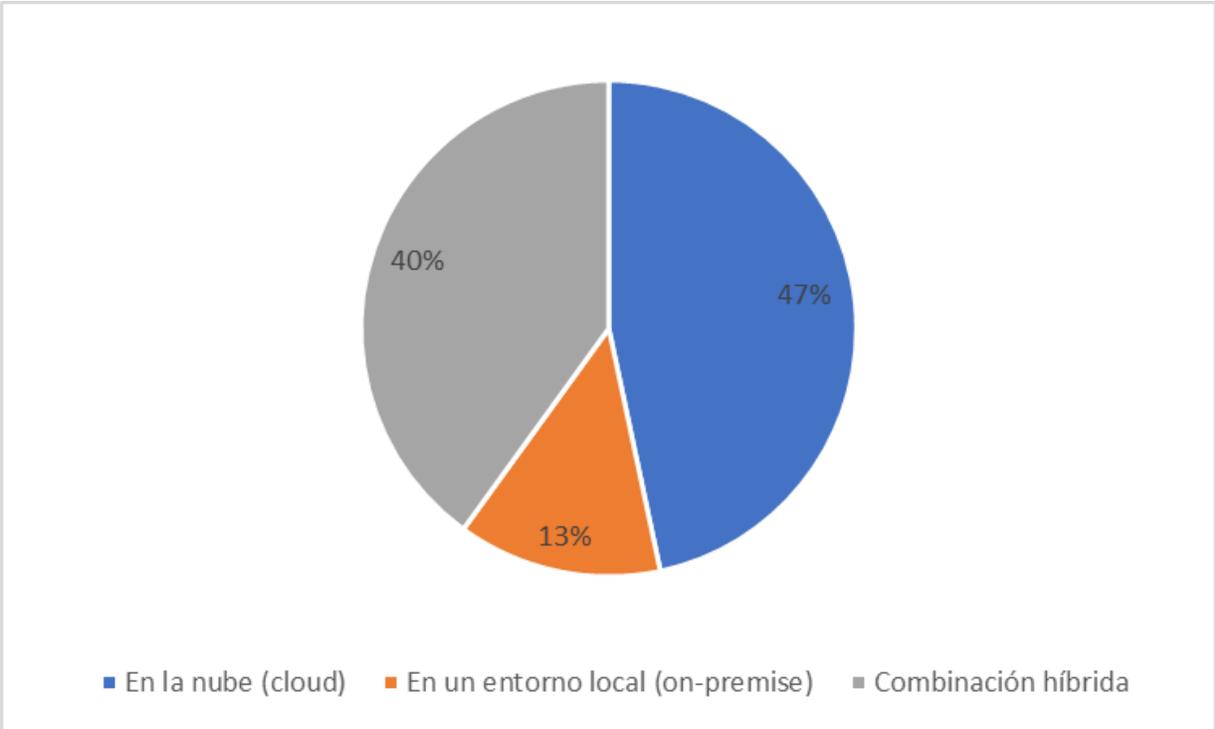
Según los resultados obtenidos, el 27% de los encuestados están familiarizados con la tecnología Openstack para IaaS, lo que indica que existe un nivel de conocimiento sobre esta plataforma dentro de la comunidad encuestada. Sin embargo, una mayoría del 73% no conoce Openstack como una solución para IaaS, lo que señala una oportunidad significativa para la educación y la formación en esta tecnología. Este desconocimiento puede ser un indicativo de

que Openstack no ha alcanzado una penetración amplia en el mercado o que existe una preferencia por otras soluciones de IaaS entre los profesionales del sector.



Pregunta 2: ¿Alguna vez a desarrollado algún proyecto en una nube comunitaria?

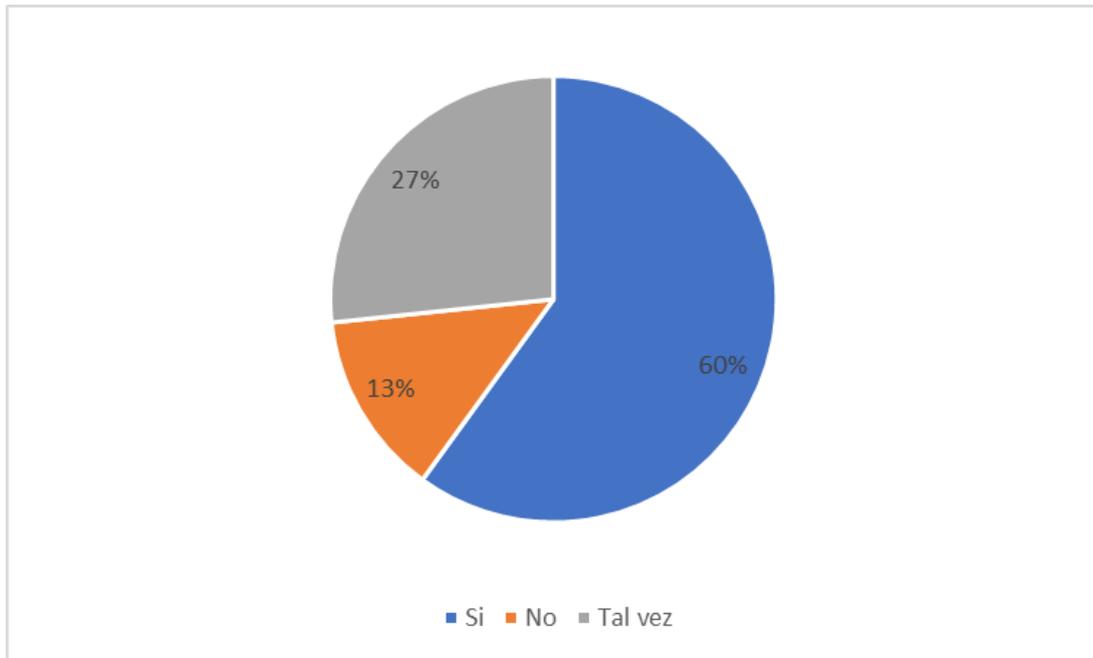
De los encuestados, un 40% afirmó haber desarrollado un proyecto en una nube comunitaria, lo que indica una experiencia significativa con esta tecnología dentro del grupo. Por otro lado, la mayoría, un 60%, indicó que no han tenido la oportunidad de trabajar con nubes comunitarias. Esto podría reflejar una tendencia en la que una proporción considerable de profesionales en el campo de TI aún no ha explorado o adoptado nubes comunitarias para el desarrollo de proyectos, lo que a su vez sugiere una posible área de crecimiento y capacitación en la industria.



Pregunta 3: En su trabajo como desarrollador de software, ¿qué tipo de tecnología utiliza principalmente para alojar tus aplicaciones y datos?

El gráfico muestra las preferencias de los desarrolladores de software en cuanto al alojamiento de aplicaciones y datos. Un 47% de los encuestados utiliza principalmente tecnologías basadas en la nube, lo que refleja una adopción significativa de soluciones en la nube en el ámbito del desarrollo de software. Por otro lado, un 40% prefiere una combinación híbrida, indicando que hay una valoración por la flexibilidad y los beneficios de combinar recursos en la nube con infraestructuras locales. Un menor porcentaje, el 13%, sigue confiando en entornos locales (on-premise) exclusivamente. Este dato sugiere que, mientras la tendencia general se inclina hacia soluciones en la nube o híbridas, aún hay una presencia

notable de desarrolladores que optan por mantener sus aplicaciones y datos en infraestructuras tradicionales.



Pregunta 4: Como desarrollador de software, si se te ofreciera la oportunidad de utilizar una nube comunitaria para tus proyectos, ¿considerarías su uso?

El gráfico representa las respuestas de los desarrolladores de software a la pregunta sobre si considerarían utilizar una nube comunitaria para sus proyectos. La mayoría, el 60%, respondió afirmativamente, mostrando una disposición considerable hacia la exploración de esta tecnología. Un 27% de los encuestados no está seguro y respondió con un "Tal vez", lo que indica cierta indecisión que podría deberse a la falta de información o experiencia con nubes comunitarias. Por otro lado, solo un 13% rechazó la idea, lo que sugiere que hay una menor resistencia al cambio hacia nuevas soluciones de nube. Estos resultados podrían

reflejar un creciente interés en las nubes comunitarias como una alternativa viable para el alojamiento de proyectos de desarrollo de software.

El análisis integral de los resultados de la encuesta entre desarrolladores de software en el sector de Tecnologías de la Información (TI) en Estados Unidos arroja una imagen matizada sobre la percepción y disposición hacia la tecnología OpenStack y las nubes comunitarias. En primer lugar, se observa que aproximadamente el 27% de los encuestados está familiarizado con OpenStack como solución para Infraestructura como Servicio (IaaS), lo que sugiere un conocimiento limitado dentro de la comunidad encuestada. Este hallazgo señala la necesidad de un esfuerzo educativo y de formación para aumentar la conciencia sobre OpenStack. Por otro lado, el 40% de los encuestados afirma haber desarrollado proyectos en nubes comunitarias, lo que indica una experiencia significativa en este ámbito. Sin embargo, el 60% que no ha tenido la oportunidad de trabajar con nubes comunitarias refleja un área de potencial crecimiento y desarrollo en la industria de TI. En cuanto a las preferencias de alojamiento de aplicaciones y datos, el hecho de que el 47% utilice principalmente tecnologías basadas en la nube muestra una fuerte adopción de soluciones en la nube en el desarrollo de software, mientras que el 40% que prefiere una combinación híbrida destaca la valoración de la flexibilidad en la elección de infraestructuras. Por último, la disposición positiva hacia la exploración de nubes comunitarias, evidenciada por el 60% que consideraría su uso, indica un creciente interés en esta tecnología como una alternativa viable para alojar proyectos de desarrollo de software. En conjunto, estos resultados sugieren que, a pesar de los desafíos de conocimiento y adopción iniciales, existe un potencial significativo para la integración exitosa de nubes comunitarias en el ámbito de TI en Estados

Unidos, respaldado por la disposición de los desarrolladores a explorar nuevas soluciones en la nube.

10. Desarrollo de la solución

Se ha desarrollado un prototipo de infraestructura de nube comunitaria geo-distribuida basada en OpenStack. Este prototipo se destaca por su capacidad para admitir múltiples regiones, y actúa como el núcleo central de alojamiento para estas regiones, que utilizan los servicios ofrecidos. Es importante mencionar que la construcción del prototipo de infraestructura para nube comunitaria se basó en documentos de Alto Nivel (HLD) y Bajo Nivel (LLD), que se pueden encontrar en los Anexos.

A continuación, se detallan los distintos requisitos que se precisaron para el desarrollo de este proyecto.

10.1. Requerimientos Funcionales:

1. Creación de Regiones: El sistema permite la creación de un número ilimitado de regiones dentro de la Nube Comunitaria. Cada Región puede configurarse de manera independiente.
2. Usuarios Federados: El sistema admite la autenticación de usuarios de manera federada, lo que significa que los usuarios pueden acceder a las diferentes regiones de la Nube Comunitaria con un solo conjunto de credenciales.
3. Persistencia de Sesión: Existe la funcionalidad de persistencia de sesión para garantizar que los usuarios puedan mantener su sesión activa al cambiar de una región a otra sin necesidad de volver a autenticarse.

4. Interfaz Web de Gestión: El sistema proporciona una interfaz web que permite a los administradores gestionar y supervisar las diferentes regiones de la Nube Comunitaria.
5. Base de Datos: Existe una base de datos centralizada que almacene información esencial y datos relacionados con el funcionamiento de la Nube Comunitaria. Esto incluye configuraciones y registros.
6. Servicio de Identificación: El sistema ofrece un servicio de identificación centralizado que permite la autenticación de usuarios en todas las regiones de la Nube Comunitaria.
7. API para Integración: Todos los servicios y funcionalidades de la Nube Comunitaria están disponibles a través de un API. Este API es accesible para las diferentes regiones y permitir la integración de servicios de manera eficiente.

10.2. Arquitectura

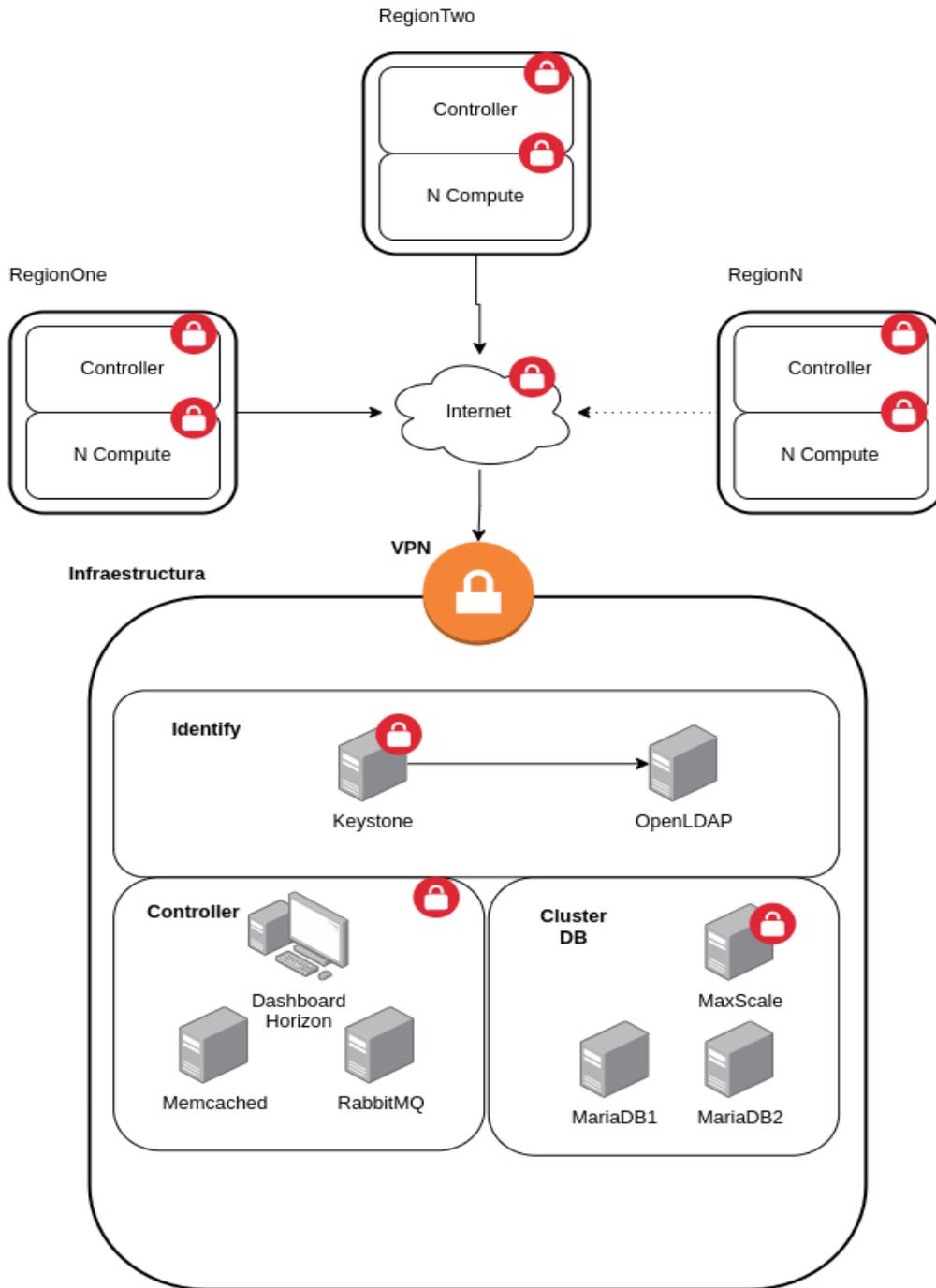


Ilustración 5: diagrama de la arquitectura del prototipo en la nube comunitaria enfocado en la infraestructura

1. RegionOne, RegionTwo, RegionN: Estas cajas representan diferentes regiones donde se despliegan los recursos de computación en la nube. Cada región puede representar un centro de datos separado o un clúster de recursos en una ubicación específica.

Controller: En cada región, hay un "Controller" que maneja las operaciones de gestión y orquestación para esa región.

N Compute: Junto a cada controlador, hay un número no especificado de nodos de computación ("N Compute") que representan los servidores virtuales que proporcionan capacidad de computación para ejecutar cargas de trabajo e instancias de máquinas virtuales.

2. Internet: Esta es la conexión general a Internet que permite la comunicación entre las diferentes regiones y el acceso remoto a la infraestructura de la nube.

3. VPN: Un icono de un candado representa una Red Privada Virtual (VPN), donde la comunicación entre las regiones o con el Internet está protegida y cifrada para la seguridad y privacidad de los datos.

4. Infraestructura: En la parte inferior, se muestra una infraestructura más detallada que incluye varios servicios de gestión de la nube:

Identify:

Keystone: Un servicio de identidad para la autenticación y autorización de usuarios y servicios.

OpenLDAP: Un sistema de directorio que puede estar integrado con Keystone para manejar la información de los usuarios y políticas de seguridad.

Controller:

Dashboard Horizon: La interfaz de usuario web para la gestión de la nube.

Memcached: Un sistema de almacenamiento en caché de memoria distribuida para mejorar el rendimiento de la base de datos.

RabbitMQ: Un sistema de mensajería para la comunicación entre los servicios de la nube.

5. Cluster DB:

MaxScale: Un proxy de base de datos para la escalabilidad y la alta disponibilidad.

MariaDB1, MariaDB2: Instancias de base de datos que probablemente están configuradas para la redundancia o replicación para la tolerancia a fallos.

Los círculos con el signo de prohibido indican que estos componentes están bloqueados o restringidos, lo que puede significar que están protegidos contra el acceso no autorizado o que están actualmente fuera de servicio.

Este diagrama se utiliza para mostrar cómo se distribuyen y se protegen los recursos en una infraestructura de nube comunitaria, detallando las regiones de implementación y los componentes de software que proporcionan las funcionalidades clave para la operación de la nube.

10.3. Configuración

A continuación, se especifica el hardware empleado según las necesidades del prototipo, detallando en términos de máquinas virtuales y sus respectivos componentes.

1. Nodo OpenLDAP sirve como un pilar fundamental en la gestión de identidades y control de acceso, ofreciendo una solución cohesiva para autenticar, autorizar y administrar usuarios en un contexto distribuido y de fácil escalabilidad.

RAM	CPU (Núcleos)	Almacenamiento	Sistema operativo
1 GB	1	50 GB	Ubuntu server

2. Nodo Keystone desempeña el papel de centinela, garantizando que únicamente los usuarios con autorización tengan acceso y puedan alterar determinados recursos. Además, ofrece un marco organizado para gestionar de manera óptima usuarios, roles y permisos en un ambiente de nube ampliable y descentralizado.

RAM	CPU (Núcleos)	Almacenamiento	Sistema operativo
4 GB	2	100 GB	Ubuntu server

3. Nodo controller desempeña el papel central en una implementación de OpenStack, alojando los servicios clave y las APIs encargadas de administrar la infraestructura de la nube. En un modelo prototípico de nube comunitaria, este nodo es esencial para la organización y gestión eficiente de los recursos y servicios brindados a usuarios y aplicaciones.

RAM	CPU (Núcleos)	Almacenamiento	Sistema operativo
4 GB	2	100 GB	Ubuntu server

4. Cluster DB desempeña un papel esencial en la conservación, estructuración y administración de la información crucial que asegura el funcionamiento armónico y eficaz de la nube. Actúa como el eje central de datos para prácticamente todos los servicios de OpenStack.

RAM	CPU (Núcleos)	Almacenamiento	Sistema operativo
4 GB	2	100 GB	Ubuntu server

5. Nodo MaxScale ofrece una administración sofisticada de conexiones y consultas para las bases de datos MariaDB, optimizando la eficiencia, fortaleciendo la seguridad y elevando la disponibilidad del sistema de base de datos principal.

RAM	CPU (Núcleos)	Almacenamiento	Sistema operativo
4 GB	2	100 GB	Ubuntu server

10.4. Topología de red

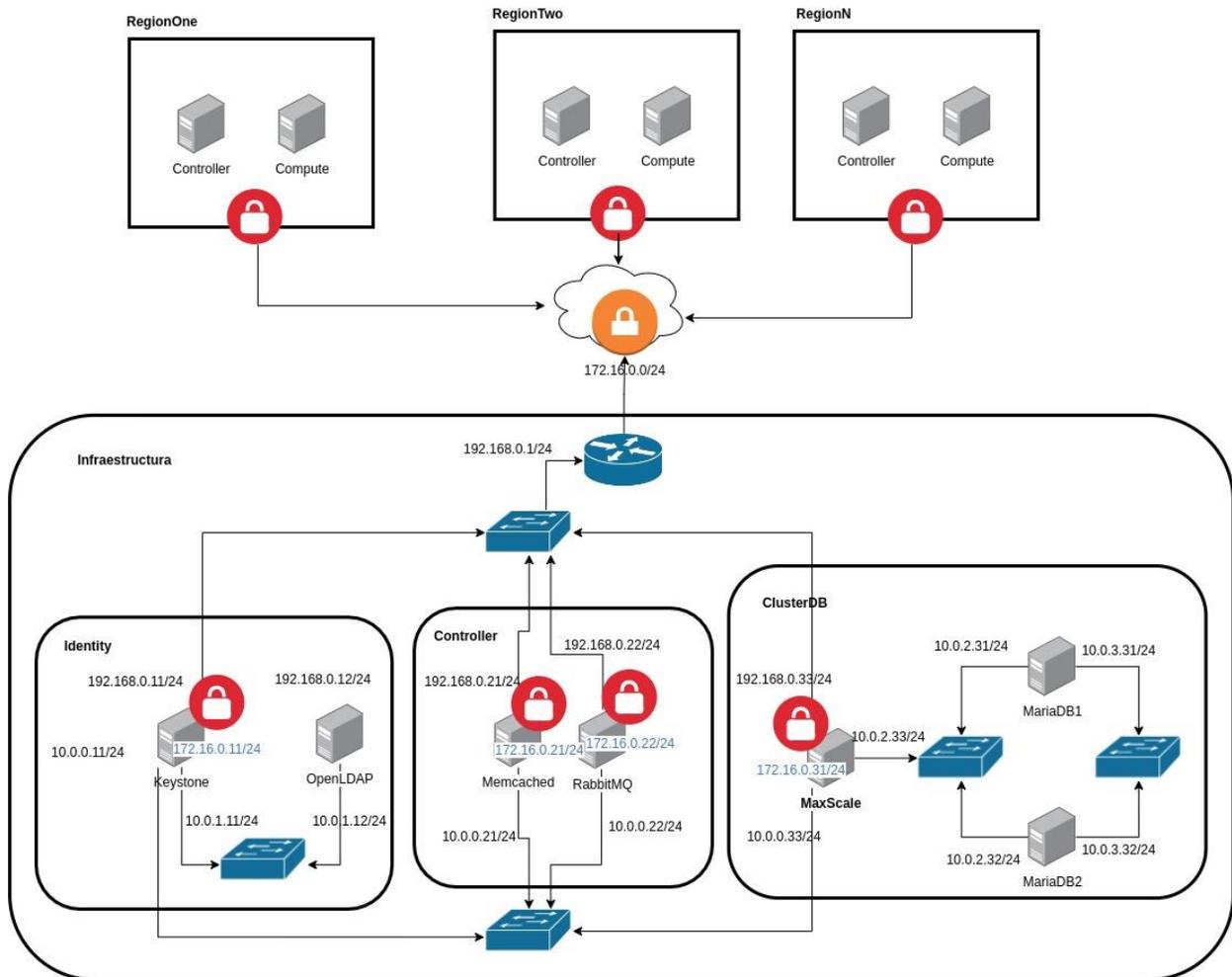


Ilustración 6: topología de red de la infraestructura

11. Caso de estudio

Prototipo centrado en Infraestructura de Nube Geo-distribuida para empresa de logística en importación de vehículos desde Estados Unidos a El Salvador.

Se plantea desarrollar una infraestructura de nube geo-distribuida utilizando OpenStack para responder a las necesidades de computación, almacenamiento y redes de comunicación de una empresa de logística especializada en la importación de vehículos usados desde Estados Unidos a El Salvador. Esta empresa gestiona diversos centros de almacenamientos de vehículos (Yardas) ubicadas en Florida, Texas, California, Nueva York, Nueva Jersey y Georgia, donde se registra información de los vehículos y se prepara para su transportación.

La sede principal de la empresa de logística está en El Salvador y es la que contará con la infraestructura central de la nube que incluye los servicios necesarios que permitan incorporar nuevos almacenamientos de vehículos (Yardas) garantizando la operabilidad de estos, aunque físicamente se encuentren en otro país.

El prototipo de infraestructura de la nube geo-distribuida hace uso del servicio Keystone para la gestión y autenticación de usuarios que a su vez están registrado en LDAP para garantizar que el usuario correcto acceda a la información y funcionalidades para las que ha sido creado.

Para el manejo de los mensajes en cola en tiempo real se hace uso de RabbitMQ. También hacemos uso MaxScale para balancear la carga y procesamiento de información. Por

otro lado, pero no menos importante la infraestructura cuenta con Memcached como servicio de chache.

Los servicios antes mencionados se integran con facilidad con OpenStack, lo cual, permite a su vez la integración de múltiples regiones que representan los almacenamientos de vehículos en este caso de estudio, brindando escalabilidad a la nube y crecimiento a la empresa. Para el almacenamiento las regiones harían uso de Ceph sin embargo esto está contemplado a un desarrollo más profundo de las regiones lo que está fuera del prototipo en cuestión ya que este se basa en la infraestructura.

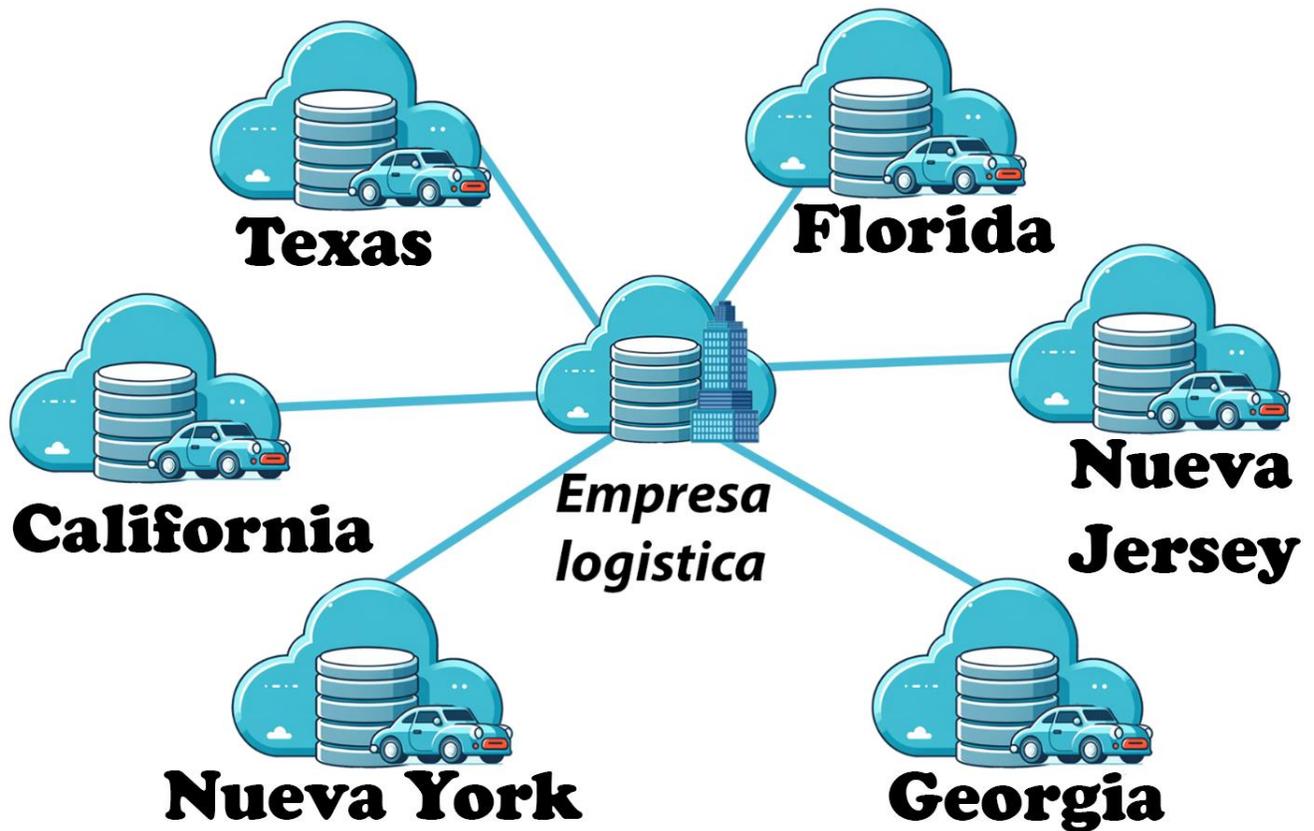


Ilustración 7: diagrama de nube comunitaria

12. Factibilidad

Al analizar la viabilidad de las nubes comunitarias desde una perspectiva de investigación, es imprescindible tener en cuenta múltiples factores. Estos no solo comprenden la viabilidad técnica, económica y operativa, sino también elementos como la demanda del mercado, la recepción por parte del colectivo de desarrolladores y las entidades, y el posible impacto en el sector y la sociedad.

12.1. Factibilidad Técnica:

Presencia y desarrollo de la tecnología: Soluciones de nubes comunitarias, tales como OpenStack y Cloud Foundry, se caracterizan por su madurez y robusto soporte, convirtiéndolas en opciones técnicamente viables.

Aspecto	OpenStack	Cloud Foundry
Madurez Tecnológica	Alta - Establecido en 2010, madurez demostrada	Moderada - Menos madura que OpenStack
Soporte y Documentación	Amplio - Gran comunidad, documentación extensa	Fuerte - Bien soportada con buena documentación
Flexibilidad de Despliegue	Alta - Personalizable para varios entornos	Moderada - Orientada principalmente a aplicaciones PaaS
Seguridad	Robusta - Opciones de seguridad avanzadas	Fuerte - Enfoque sólido en seguridad
Escalabilidad	Alta - Escala bien en grandes entornos	Moderada - Buena para aplicaciones escalables
Comunidad y Ecosistema	Extensa - Amplia adopción y comunidad activa	Activa - Comunidad sólida, pero más enfocada en PaaS

Este cuadro destaca las fortalezas y limitaciones de ambas plataformas en áreas cruciales como la madurez tecnológica, el soporte, la flexibilidad, la seguridad, la escalabilidad y la comunidad. OpenStack muestra una ventaja en términos de madurez tecnológica, flexibilidad de despliegue y escalabilidad, lo que lo hace ideal para

infraestructuras de nube comunitaria a gran escala. Por otro lado, Cloud Foundry, con su enfoque en PaaS y una comunidad activa, es fuerte en seguridad y tiene un buen soporte, pero está más orientado a aplicaciones específicas en lugar de a infraestructuras de nube a gran escala

Integración y compatibilidad: La viabilidad técnica también está condicionada por la sencillez con la que estas soluciones pueden fusionarse con tecnologías ya presentes y su capacidad para ajustarse a variados entornos y necesidades.

12.2. Factibilidad Económica:

Para evaluar si es factible crear un prototipo, comparamos los costos entre utilizar los servicios de Google Cloud Platform, proveedor de nube pública, y la construcción de nuestra propia infraestructura física. Debido a que crear una nube privada implica un gasto inicial grande, queremos determinar cuánto tiempo mínimo llevaría igualar o reducir los costos de la nube privada en comparación con los costos de usar la nube pública. Hemos elegido Google Cloud Platform para este análisis no porque haya una razón específica, sino porque los precios en las nubes públicas tienden a ser bastantes estándares entre diferentes proveedores para servicios similares.

Se necesitará los siguientes recursos de máquinas a desplegar:

	CPU (Nucleos)	Almacenamiento	RAM (GB)
Keystone	8	100	16
LDAP	8	100	16
Controller	4	100	8
BD (Base de Datos)	8	200	16

Por lo tanto, los costos aproximados que hemos investigado para el cada una de las opciones a evaluar son los siguientes:

	Google Cloud
Keystone	744.768
LDAP	744.768
Controller	552.384
Base de datos	1,104.768
TOTAL	3,519.072

Cabe decir que este TOTAL de \$ 3,519.072 es un costo mensual. Por lo que el costo anual asciende a \$ 42,228.864

Para los costos de una infraestructura física hemos añadido un Ethernet Switch y un Router. Por lo que los costos quedarían de la siguiente manera.

	Infraestructura física		
	Precio	Cantidad	Total
Easy Cube I9 Micro Data	10000	1	10000
Ethernet Switch	2900	1	2900
Network Security Firewall	2000	1	2000
Servidor (Nodos)	5000	5	25000
TOTAL			\$39,900

Cabe mencionar que, para los costos de la infraestructura física, no se han tomado en cuenta aspectos relacionado por ejemplo a la inflación, personal especializado, espacio físico para el centro de datos, aire acondicionado, etc.

12.3. Comparativa de Costos

Proveedor	Primer Año	Segundo Año	Tercer Año	Cuarto Año
Google Cloud	\$ 42,228.864	\$ 42,228.864	\$ 42,228.864	\$ 42,228.864
Infraestructura Física	\$ 39,900 + \$ 12,800 (Mantenimiento + Energía Eléctrica)	\$ 12,800 (Mantenimiento + Energía Eléctrica)	\$ 12,800 (Mantenimiento + Energía Eléctrica)	\$ 12,800 (Mantenimiento + Energía Eléctrica)

Recordando que esta es una estimación general y que los costos pueden variar en función de factores específicos, por lo que es importante ajustar estos cálculos según la situación y los datos exactos disponibles. También tener en cuenta que la vida útil del hardware puede afectar significativamente el costo anual.

Basándonos en lo anterior, se puede observar que, en un principio, la inversión en Google Cloud supone un costo inicial más bajo en comparación con la inversión necesaria para una infraestructura física. Sin embargo, también es evidente que, a partir del segundo año, los costos anuales asociados a Google Cloud son sustancialmente más altos en comparación con la infraestructura física. Esto implica que, a largo plazo, la implementación de una infraestructura física se vuelve una opción económicamente rentable.

El diseño propuesto en el prototipo es escalable horizontalmente, lo que significa que se pueden agregar más recursos de hardware según las necesidades.

En consecuencia, existe una clara viabilidad para la implementación de la infraestructura física en lugar de Google Cloud, especialmente desde una perspectiva

económica y a un plazo mayor a dos años. Esto indica que la inversión inicial se amortizaría a lo largo del tiempo, proporcionando beneficios económicos significativos.

12.4. Factibilidad humana

Para estimar un número mínimo de personas necesarias para cada aspecto de la factibilidad humana en la implementación de una nube comunitaria basada en OpenStack, es importante tener en cuenta que estos números pueden variar según el tamaño de la empresa, la complejidad del proyecto y la infraestructura existente. A continuación, proporciono una estimación generalizada basada en estos factores:

Aspecto	Descripción	Número Mínimo de Personas
Disponibilidad de Expertos	Evaluar la existencia de personal con experiencia en OpenStack o la necesidad de contratar o capacitar.	2
Capacitación y Desarrollo	Programas de formación para el personal en la administración de nubes, seguridad de datos, y operaciones de OpenStack.	1
Gestión del Cambio Organizacional	Administrar la transición a la nueva tecnología, asegurando una adaptación organizacional efectiva.	1
Soporte y Mantenimiento	Asegurar personal calificado para el mantenimiento y soporte continuos de la infraestructura.	2
Disposición para la Innovación	Medir y fomentar la disposición de la empresa para adoptar nuevas tecnologías y cambios.	1 (parte del equipo de gestión)
Recursos para Operación Continua	Considerar los recursos humanos necesarios para la operación y mantenimiento constantes de la nube.	2
Impacto en Personal de TI y Usuarios	Evaluación del impacto en las tareas y responsabilidades actuales, y ajuste de roles o descripciones de trabajo.	1

Estos números son estimaciones y pueden variar. Por ejemplo, en una empresa más grande o un proyecto más complejo, se necesitarán más recursos humanos. Además, algunas

personas pueden tener roles multifuncionales, especialmente en organizaciones más pequeñas. La planificación detallada del proyecto proporcionará una comprensión más clara de los recursos humanos necesarios.

13. Conclusiones

1. La investigación destacó la importancia crítica de desarrollar una infraestructura de nube comunitaria basada en OpenStack que va más allá de la simple implementación de una arquitectura de nube. Se concluyó que es esencial garantizar la disponibilidad y la seguridad de los datos, lo cual es un reflejo directo de la necesidad creciente en el mercado de soluciones en la nube que son no solo funcionales sino también seguras y resilientes.
2. El estudio abordó efectivamente uno de los retos más significativos en la creación de infraestructuras de nube: el desarrollo de un sistema capaz de soportar múltiples regiones. La capacidad de manejar cargas de trabajo distribuidas geográficamente es crucial, y este proyecto demostró la viabilidad de soluciones multirregionales en el contexto de la nube comunitaria.
3. Se identificó que una robusta autenticación federada y la persistencia de sesión son vitales para entornos de nube seguros y eficientes. La investigación concluyó que estas características son fundamentales para asegurar que solo los usuarios autorizados accedan a recursos específicos y que su experiencia sea coherente y protegida.
4. La capacidad de administrar la infraestructura de manera centralizada a través de una interfaz web y una base de datos unificada se reveló como un componente esencial. Esta gestión centralizada es crucial para una operación

efectiva y para mantener una visión integrada y cohesiva de la infraestructura, especialmente cuando se trabaja con múltiples regiones.

14. Recomendaciones

1. Evaluación Continua: A medida que se desarrolle el prototipo, es crucial realizar evaluaciones periódicas para garantizar que se cumplan todos los objetivos específicos y que el prototipo siga siendo relevante y adecuado para las necesidades de la comunidad.
2. Seguridad: Aunque se menciona la seguridad en los objetivos, sería recomendable implementar un enfoque de seguridad por diseño y considerar la realización de auditorías de seguridad para identificar y mitigar posibles vulnerabilidades.
3. Formación y Capacitación: Acompañar la documentación con sesiones de formación y capacitación para garantizar que los usuarios y administradores estén familiarizados con la nueva infraestructura y puedan aprovecharla al máximo.
4. Integración con Otras Plataformas: Para aumentar la adopción y versatilidad del prototipo, podría ser beneficioso considerar cómo podría integrarse con otras plataformas o servicios en la nube existentes.
5. Feedback de la Comunidad: Dado que es una nube comunitaria, sería esencial obtener retroalimentación regular de la comunidad de usuarios para identificar áreas de mejora y garantizar que el servicio continúe satisfaciendo sus necesidades.

6. Escalabilidad Futura: Aunque el prototipo se someterá a pruebas de escalabilidad, es fundamental considerar cómo la infraestructura podría adaptarse o ampliarse en el futuro para atender a una comunidad en crecimiento.

15. Referencias

- Adkins, S., Belamaric, J., Giersch, V., Makogon, D., & Robinson, J. (2015). *OpenStack Cloud Application Development*. Indianapolis: A Wiley Brand.
- Age2. (10 de 09 de 2023). *Regiones Cloud ¿Cuál es su papel?* Recuperado el 9 de 09 de 2023, de Age2: <https://www.age2.es/noticias/regiones-cloud-cual-es-su-papel/>
- Aguilar, L. J. (2012). *COMPUTACIÓN EN LA NUBE - Estrategias de Cloud Computing en las Empresas*. México: Alfaomega Grupo Editor, S.A. de C. V.
- Arias, Á. (2015). *Computación en la nube: 2a Edición*. IT Campus Academy.
- AWS. (s.f.). *¿Qué es la virtualización? - Explicación de la virtualización de la computación en la nube*. Recuperado el 8 de 09 de 2023, de Amazon Web Services: <https://aws.amazon.com/es/what-is/virtualization/>
- CloudAMQP. (23 de 09 de 2019). *Part 1: RabbitMQ for beginners - What is RabbitMQ?* Obtenido de CloudAMQP: <https://www.cloudamqp.com/blog/part1-rabbitmq-for-beginners-what-is-rabbitmq.html#:~:text=Message%20queueing%20allows%20web%20servers,to%20balance%20loads%20between%20workers>
- databricks. (s.f.). *Orchestration*. Obtenido de databricks: <https://www.databricks.com/glossary/orchestration>
- Dawood, O. (07 de 11 de 2022). *What are message queues?* Obtenido de Contrast Security: <https://www.contrastsecurity.com/security-influencers/what-is-a-message-queue->

Luna, A. C. (2014). *Cloud: herramientas para trabajar en la nube*. España: Editorial ICB.

Niedringhaus, C. (09 de 12 de 2019). *What Is OpenLDAP?* Obtenido de JumpCloud:

<https://jumpcloud.com/blog/openldap-active-directory-migration#8203;%60%60%E3%80%90oacite:0%E3%80%91%60%60​>

Nutanix. (s.f.). *¿Qué es un hipervisor?* Obtenido de Nutanix:

<https://www.nutanix.com/mx/info/hipervisor#:~:text=Los%20ejemplos%20de%20hipervisores%20m%C3%A1s,Hyper%2DV%20y%20Citrix%20Hypervisor.>

openstack. (27 de 01 de 2022). *Image service overview*. Recuperado el 9 de 9 de 2023, de

openstack: <https://docs.openstack.org/glance/yoga/install/get-started.html>

Pathak, A. (1 de 08 de 2023). *Orquestación de datos en términos simples [+5 herramientas]*.

Recuperado el 10 de 09 de 2023, de Geekflare: <https://geekflare.com/es/data-orchestration/#geekflare-toc-what-is-data-orchestration>

Rackspace. (2023). Obtenido de *Qué es OpenStack y por qué es la plataforma de nube de*

elección.: <https://www.rackspace.com/es/info/cloud/openstack>

Red Hat. (28 de 05 de 2019). *What is cloud infrastructure?* Obtenido de Red Hat:

<https://www.redhat.com/en/topics/cloud-computing/what-is-cloud-infrastructure#:~:text=Cloud%20infrastructure%20is%20a%20term,cloud%2C%20you%20need%20cloud%20infrastructure>

Red Hat. (11 de 05 de 2022). *¿Qué es KVM?* Recuperado el 9 de 09 de 2023, de Red Hat:

<https://www.redhat.com/es/topics/virtualization/what-is-KVM>

Red Hat. (23 de 03 de 2023). *¿Qué es un hipervisor?* Obtenido de Red Hat:

<https://www.redhat.com/es/topics/virtualization/what-is-a-hypervisor>

Red Hat. (s.f.). *What is lightweight directory access protocol (LDAP) authentication?*

Obtenido de Red Hat: <https://www.redhat.com/en/topics/security/what-is-ldap-authentication>

Stephen J. Bigelow. (10 de 05 de 2019). *Improve cloud performance with these 6 techniques.*

Obtenido de TechTarget:

<https://www.techtarget.com/searchcloudcomputing/tip/Improve-cloud-performance-with-these-6-techniques>

SUSE . (s.f.). *Computer Cluster*. Obtenido de Technology Definitions | SUSE Defines:

<https://www.suse.com/suse-defines/definition/computer-cluster/>

La siguiente encuesta fue realizada a través de Google forms

Exploring Cloud Solutions Adoption and Perception: A Focus on OpenStack and Community Clouds

This survey is designed to gather valuable insights into current trends in the adoption and perception of cloud solutions within the Information Technology sector. With a specific focus on OpenStack and community clouds, we aim to better understand how IT professionals, especially software developers, view and use these technologies in their daily work. Your responses will help us identify key areas of interest and development in cloud infrastructure, as well as improve the practices and tools that support these solutions. Your participation is crucial in gaining a comprehensive view of the current state and potential future of cloud technologies in the professional realm.

Are you familiar with OpenStack technology as a solution for offering IaaS (Infrastructure as a Service)? *

- Yes
- No

Have you ever developed a project on a community cloud? *

- Yes
- No

In your work as a software developer, what kind of technology do you primarily use to host your applications and data? *

- Cloud
- In a local environment (on-premise)
- Hybrid combination

As a software developer, if you were offered the opportunity to use a community cloud for your projects, would you consider its use? *

- Yes
- No



**UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERIA Y ARQUITECTURA**

**DOCUMENTO HLD
(DISEÑO DE ALTO NIVEL)**

**PROTOTIPO CENTRADO EN UNA INFRAESTRUCTURA DE
NUBE COMUNITARIA GEO-DISTRIBUIDA**

Noviembre 2023
Ver. 2.0 30/Nov/2023

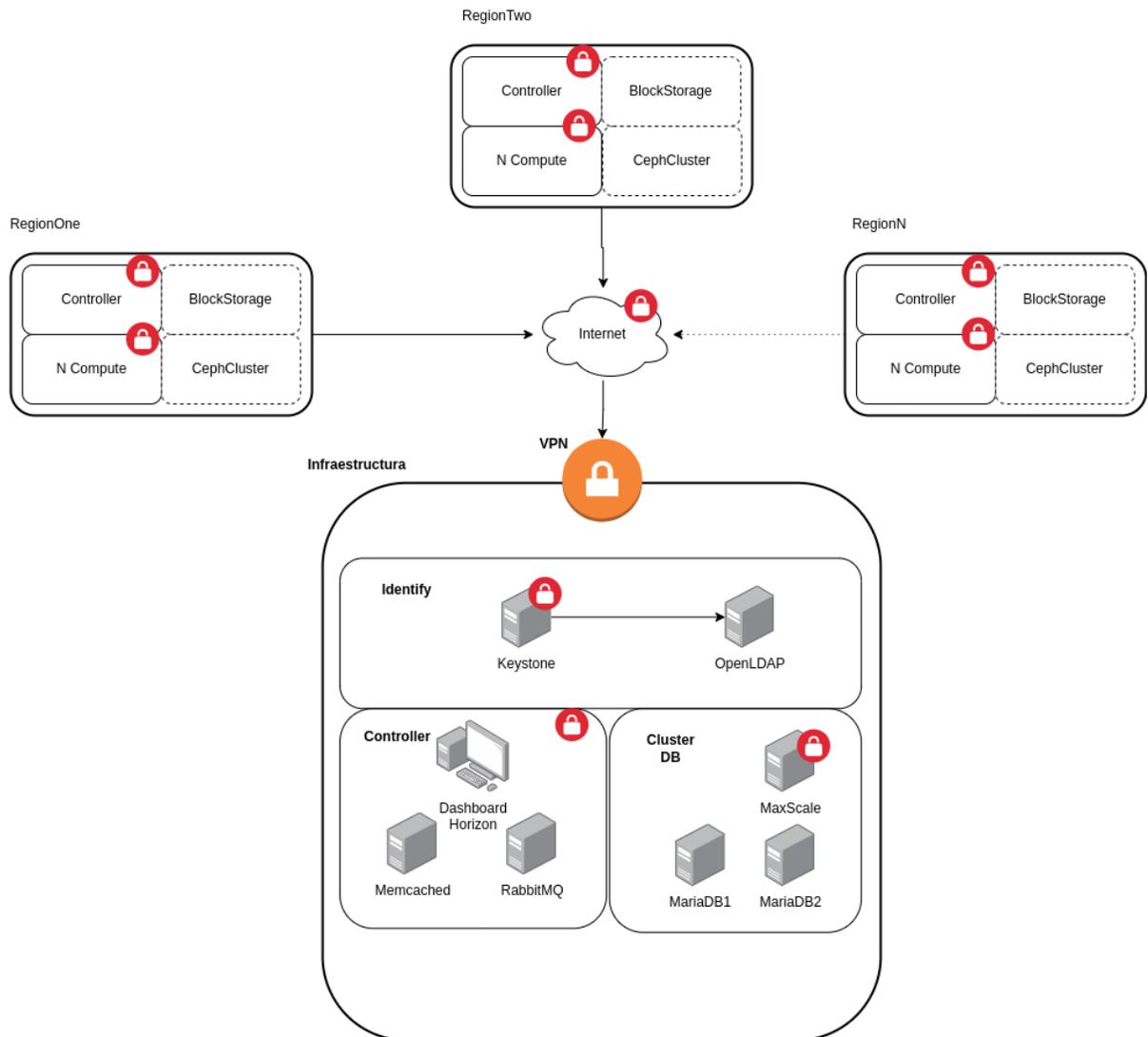
Contenido

Objetivo del High-Level Design (HLD):	1
Diagrama de arquitectura de alto nivel.	2
Descripciones de los componentes principales.	2
Infraestructura	2
Regiones.....	4
Diagrama de Red de Alto nivel.....	6
Listado de Redes:.....	6

Objetivo del High-Level Design (HLD):

El propósito central de este documento es presentar una visión panorámica de la arquitectura del prototipo de nube comunitaria, utilizando OpenStack como base. Se busca proporcionar una comprensión clara de la estructura general del sistema, identificando sus componentes fundamentales y estableciendo las interconexiones esenciales entre ellos. Este HLD sirve como referencia principal para los involucrados en el proyecto, ofreciendo una visión global que facilita la comprensión del enfoque adoptado en el diseño del prototipo. Este documento servirá como guía fundamental para desarrolladores, arquitectos y otros interesados, brindando una comprensión global de la solución propuesta que facilita la comprensión del enfoque adoptado en el diseño del prototipo.

Diagrama de arquitectura de alto nivel.



Descripciones de los componentes principales.

Infraestructura

En el marco de la infraestructura de la nube comunitaria basada en OpenStack, varios componentes esenciales trabajan en conjunto para proporcionar funcionalidades clave. A continuación, se detallan estos componentes, cada uno desempeñando un papel crucial en la creación y gestión efectiva de la infraestructura en la nube.

Keystone: es el servicio de identidad de OpenStack. Gestiona la autenticación de usuarios y la asignación de permisos dentro del sistema. Proporciona tokens de acceso para autenticar servicios y usuarios en toda la nube.

OpenLDAP: es un sistema de directorio que se utiliza para almacenar información de usuarios y configuraciones. En el contexto de OpenStack, puede integrarse con Keystone para almacenar información de identidad, como usuarios y grupos, de manera centralizada.

Horizon: es el panel de control web de OpenStack. Proporciona una interfaz gráfica de usuario para gestionar y supervisar los recursos de la nube, permitiendo a los usuarios y administradores realizar acciones como lanzar instancias, gestionar redes y almacenamiento, entre otras cosas. Además, es la que nos permitirá cambiar entre Regiones en la nube comunitaria.

Memcached: es un sistema de almacenamiento en caché de objetos en memoria. En el contexto de OpenStack, se utiliza para mejorar el rendimiento al almacenar en caché datos frecuentemente accedidos, como consultas de base de datos y tokens de autenticación.

RabbitMQ: es un sistema de mensajería que facilita la comunicación entre los diferentes componentes de OpenStack. Actúa como un intermediario para la transmisión eficiente de mensajes entre servicios, permitiendo la escalabilidad y la gestión de la carga de trabajo.

MaxScale: MaxScale es un servidor de proxy y enrutamiento que se utiliza para mejorar la escalabilidad y la disponibilidad de la base de datos. En un entorno OpenStack, podría implementarse para gestionar el tráfico de la base de datos, distribuir cargas y mejorar la redundancia.

VPN: La VPN (Red Privada Virtual) puede utilizarse para establecer conexiones seguras entre diferentes segmentos de la nube comunitaria o permitir el acceso seguro desde ubicaciones externas. Puede ser esencial para garantizar la privacidad y la seguridad de la comunicación en una infraestructura distribuida.

Regiones

Controller:

El componente "Controller" es esencial para la gestión y coordinación del entorno OpenStack. En una implementación multi-región, los servicios de controlador gestionan y coordinan las operaciones en cada región. Aquí hay una descripción de los servicios específicos:

Nova (Compute Service): responsable de la gestión de instancias de máquinas virtuales. Coordinación y programación de máquinas virtuales en los nodos de cómputo. Ofrece servicios relacionados con la creación, ejecución y terminación de instancias de máquinas virtuales.

Neutron (Networking Service): Proporciona servicios de red como la creación y gestión de redes y subredes. Asociación de direcciones IP a instancias de máquinas virtuales. Gestión de reglas de seguridad y cortafuegos.

Glance (Image Service): Almacena y recupera imágenes de máquinas virtuales. Proporciona una interfaz para la gestión y el descubrimiento de imágenes.

Placement: Gestiona la colocación de cargas de trabajo en los recursos disponibles. Ayuda a optimizar la distribución de recursos en el entorno de OpenStack.

Compute:

El componente "Compute" se encarga de la ejecución de instancias de máquinas virtuales y del procesamiento de las operaciones relacionadas con ellas.

Nova (Compute Service): es responsable de la implementación y gestión de las instancias de máquinas virtuales.

BlockStorage (Opcional): proporciona almacenamiento de bloques que se puede conectar a las instancias de máquinas virtuales.

Cinder (Block Storage Service): Gestiona volúmenes de bloques y su asociación con instancias de máquinas virtuales.

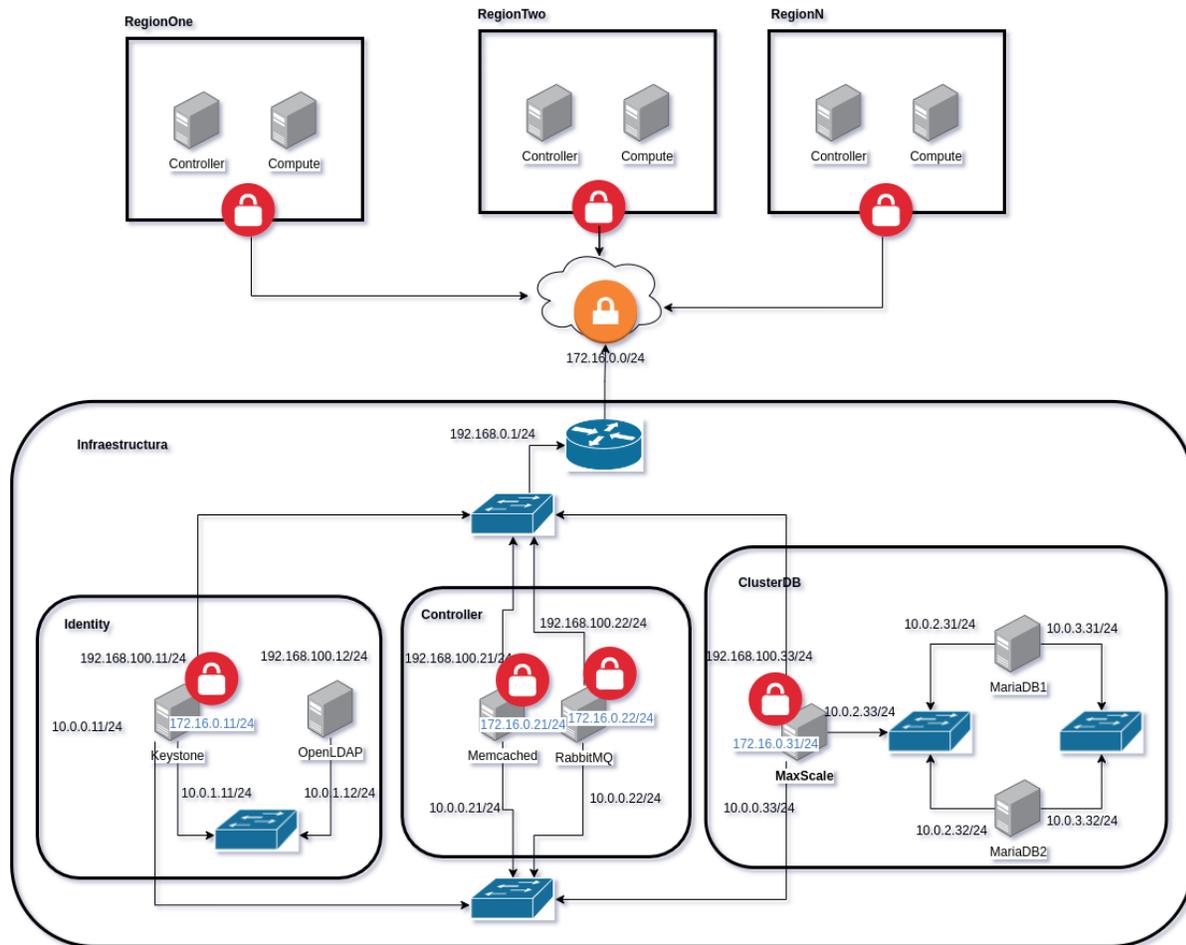
CephFS (Opcional):

CephFS es un sistema de archivos distribuido que puede utilizarse para proporcionar almacenamiento de archivos distribuido y escalable.

CephFS: Proporciona almacenamiento de archivos distribuido. Escala horizontalmente para manejar grandes volúmenes de datos.

Estas descripciones proporcionan un resumen de las funciones y responsabilidades clave de cada componente en las regiones del prototipo de nube comunitaria. Cabe destacar que la inclusión de componentes opcionales como BlockStorage y CephFS dependerá de los requisitos específicos del proyecto y de la necesidad de proporcionar servicios adicionales a los usuarios de la nube comunitaria.

Diagrama de Red de Alto nivel



Listado de Redes:

InternetAccessNet (192.168.0.0/24): Esta red se utiliza para la conexión a Internet desde las máquinas virtuales. Proporciona acceso a recursos externos y servicios en la nube.

IdentityControllerNet (10.0.0.0/24): Esta red interconecta los módulos de Identity, Controller y ClusterDB en la infraestructura.

IdentityInternalNet (10.0.1.0/24): Red interna utilizada por el módulo de Identity, que incluye Keystone y OpenLDAP. Facilita la comunicación interna entre estos servicios.

ClusterDBBalanceNet (10.0.2.0/24): Esta red conecta MaxScale con MariaDB1 y MariaDB2 para proporcionar balanceo de carga en el módulo de ClusterDB.

ClusterDBGaleraNet (10.0.3.0/24): Red utilizada para el Galera Clúster en el módulo de ClusterDB. MariaDB1 y MariaDB2 se comunican a través de esta red para la sincronización del clúster.

VPNet (172.16.0.0/24): Esta red VPN conecta máquinas como Keystone, Memcached, RabbitMQ y MaxScale, permitiendo la vinculación de estos servicios en diferentes regiones de la nube comunitaria.



**UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERIA Y ARQUITECTURA**

**DOCUMENTO LLD
(DISEÑO DE BAJO NIVEL)**

**PROTOTIPO CENTRADO EN UNA INFRAESTRUCTURA DE NUBE
COMUNITARIA GEO-DISTRIBUIDA**

Noviembre 2023
Ver. 2.0 30/Nov/2023

Contenido

Objetivo del Low-Level Design(LLD):	1
Instalación de Máquinas Virtuales KVM (Kernel-based Virtual Machine)	2
Indicaciones de comandos	2
Hardware de Virtualización:	2
Pasos para Instalar KVM:	2
Creación de Máquina Virtual en KVM.....	4
Creacion de Maquina Virtual.....	4
Pasos para añadir una nueva red virtual.....	12
Configuración de Nodos	15
Modulo Clúster DB (Data Base).....	15
Instalación Clúster MariaDB	19
Modulo Controller	25
OpenStack packages for Ubuntu.....	27
Instalación Memcached.....	28
Instalación Etc.....	28
Modulo Entity	32
Instalación OpenLDAP.....	35
Instalación VPN.....	47

Objetivo del Low-Level Design(LLD):

El objetivo principal del Diseño de Nivel Bajo (LLD) es proporcionar una especificación técnica detallada para la implementación del prototipo de nube comunitaria, abordando de manera exhaustiva la configuración y conexión de los componentes de OpenStack en las diferentes regiones, así como detallando la interconexión entre las redes internas, y asegurando la escalabilidad, seguridad y eficiencia de la infraestructura. Este LLD está dirigido a desarrolladores, ingenieros de sistemas y otros profesionales de TI involucrados en la implementación y mantenimiento del prototipo de nube comunitaria.

Instalación de Máquinas Virtuales KVM (Kernel-based Virtual Machine)

Indicaciones de comandos

```
$ comando
```

Cualquier usuario, incluido el usuario root, puede ejecutar comandos que tengan el prefijo \$ en el prompt.

```
# comando
```

El usuario root debe ejecutar comandos que tengan el prefijo #. También podemos agregar el prefijo sudo a estos comandos, si está disponible, para ejecutarlos.

Para nuestra instalación de Máquinas Virtuales ocuparemos el Sistema Operativo (SO) Ubuntu 22.04.3 LTS. Antes de comenzar con la instalación de KVM, nos aseguraremos de que el sistema cumple con los siguientes requisitos:

Hardware de Virtualización:

Verificamos que el procesador admite virtualización y que la virtualización está habilitada en la configuración del BIOS/UEFI.

Sistema Operativo:

Asegurémonos de tener una instalación limpia de Ubuntu 22.04.3 LTS.

Pasos para Instalar KVM:

Verificar la Compatibilidad del Hardware:

Ejecuta el siguiente comando para verificar que la virtualización está habilitada:

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo  
16
```

Si obtenemos:

0 – La CPU No es apta para la virtualización.

1 o más – La CPU es apta para la virtualización. En este ejemplo hemos obtenido 16.

De forma adicional o alternativa podemos instalar la siguiente herramienta:

```
$ apt-get install cpu-checker
```

Para luego ejecutar el siguiente comando:

```
$ kvm-ok
```

Si el resultado es:

```
$ kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
```

Significa que puede ser utilizado. De lo contrario tendrá la siguiente salida en pantalla:

```
$ kvm-ok
INFO: Your CPU does not support KVM extensions
KVM acceleration can NOT be used
```

Es recomendable realizar la instalación en un sistema operativo con arquitectura de procesador de 64 bits.

Instalar KVM y Herramientas Relacionadas:

Actualizamos el sistema e instalamos los paquetes necesarios:

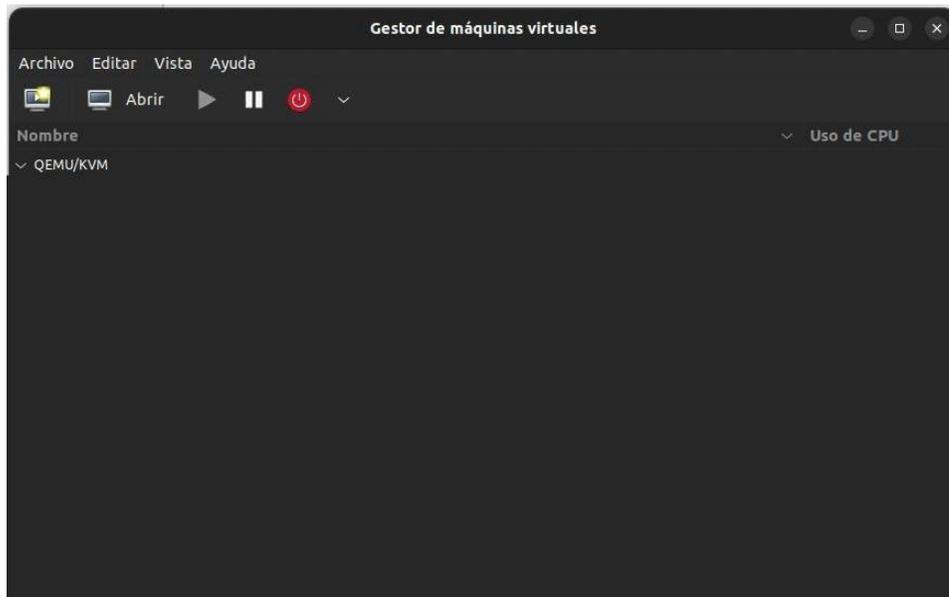
```
$ sudo apt update
$ sudo apt install qemu-kvm libvirt-daemon-system libvirt-
clients bridge-utils virtinst virt-manager
```

Creación de Máquina Virtual en KVM

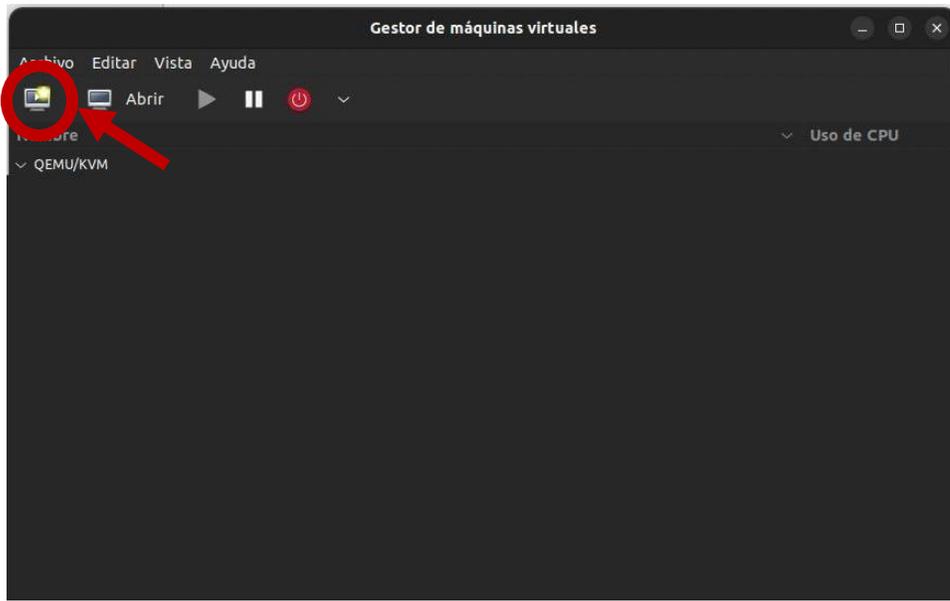
Cabe decir que las máquinas virtuales que se creen en el KVM serán SO Ubuntu Server 22.04.3 LTS para cada uno de los servicios requerido en la Infraestructura y/o en las Regiones.

Creación de Máquina Virtual

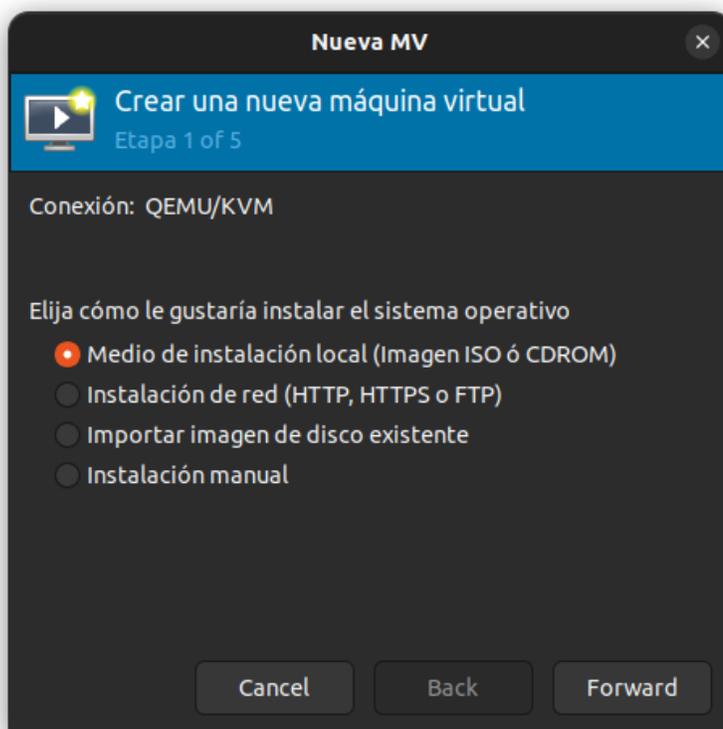
Para la creación de una nueva máquina virtual abriremos el programa “Gestor de Máquinas Virtuales”, se puede encontrar en la búsqueda de programas. Se mostrará esto:



Luego daremos nos iremos a Archivo->Nueva máquina virtual, o podemos dar click directamente al icono para Crear una máquina virtual nueva.

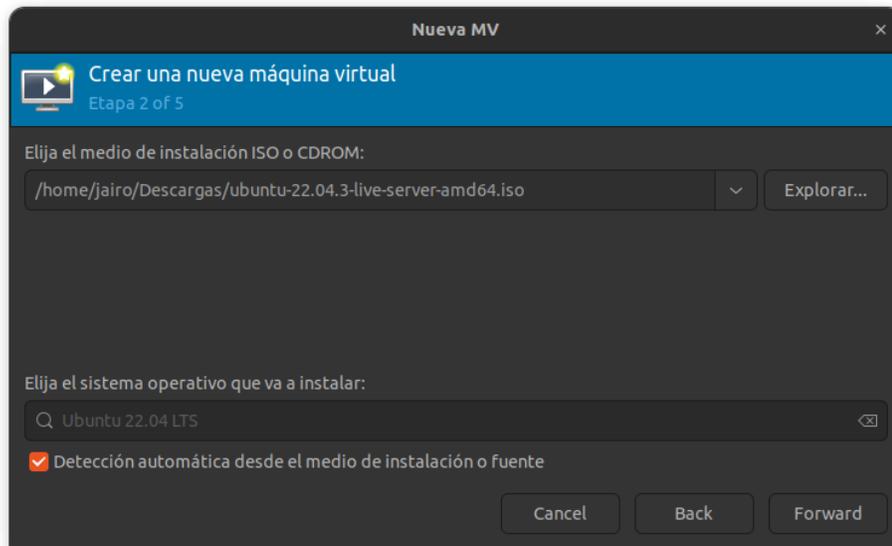


Aparecerá la siguiente ventana:

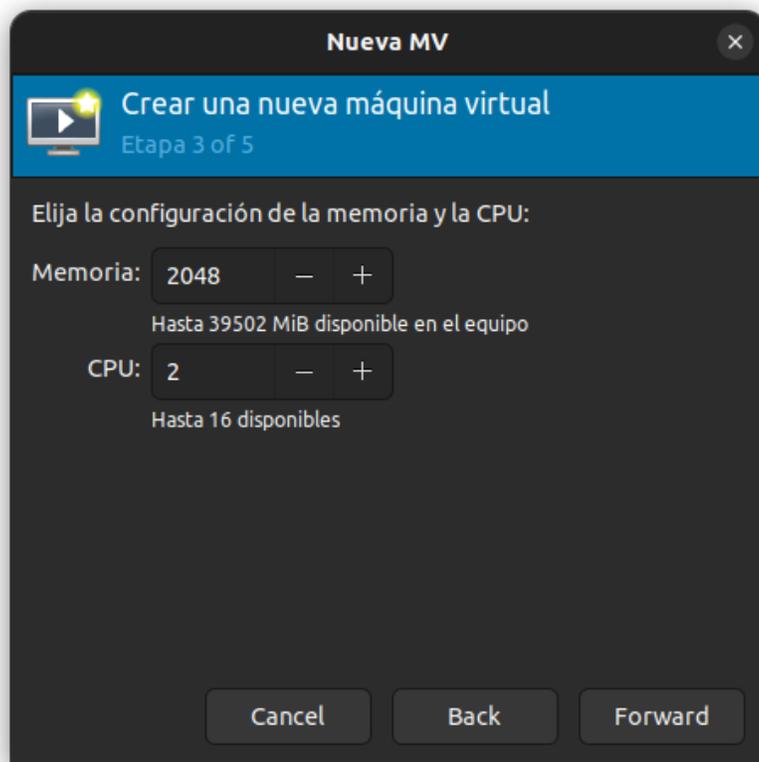


Seleccionamos la opción de Medio de instalación local (Imagen ISO o CDROM)

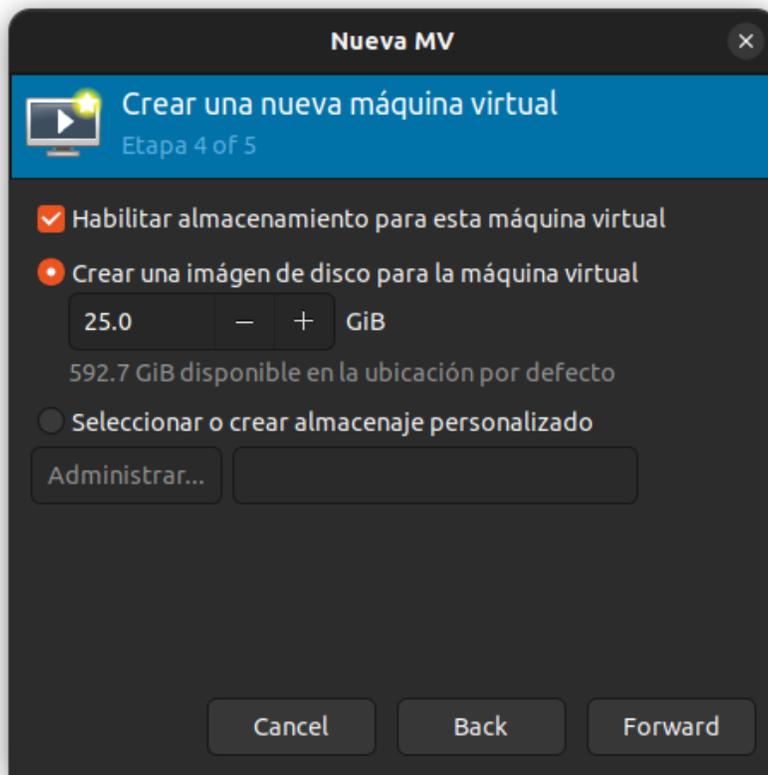
Damos click en Explorar y seleccionamos la imagen ISO del SO (Sistema Operativo) de Ubuntu.



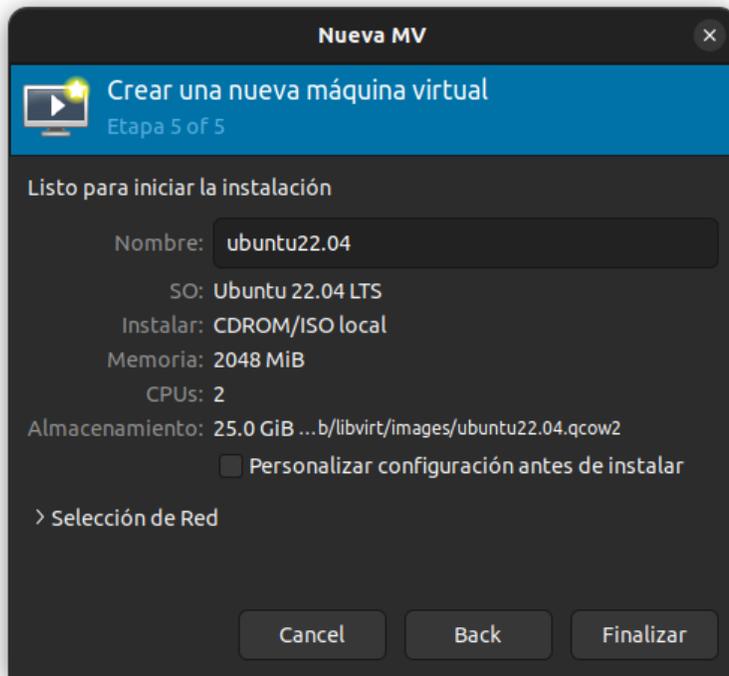
Damos click en Forward y asignamos el recurso necesario que queramos ponerle a la máquina, para este ejemplo se pondrá 2048 RAM y 2 CPU. Nota: Para las especificaciones de recursos de la maquinas del prototipo tendrán asignado otra configuración de recursos. Y se debe considerar la cantidad de recurso que tiene la maquina anfitrión donde se creen las máquinas virtuales.



Además, podremos asignarle la cantidad de disco en GiB, para este ejemplo pondremos 25 GiB.

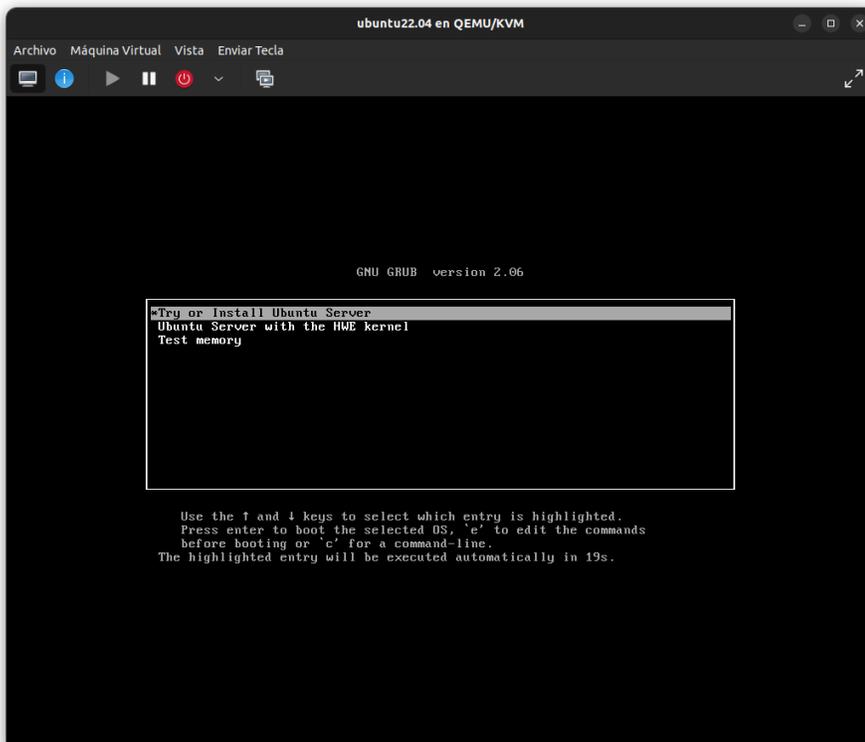


Luego solo se pedirá un nombre para la máquina virtual el cual debe tener un nombre no repetible para todas las maquinas que se lleguen a desplegar.

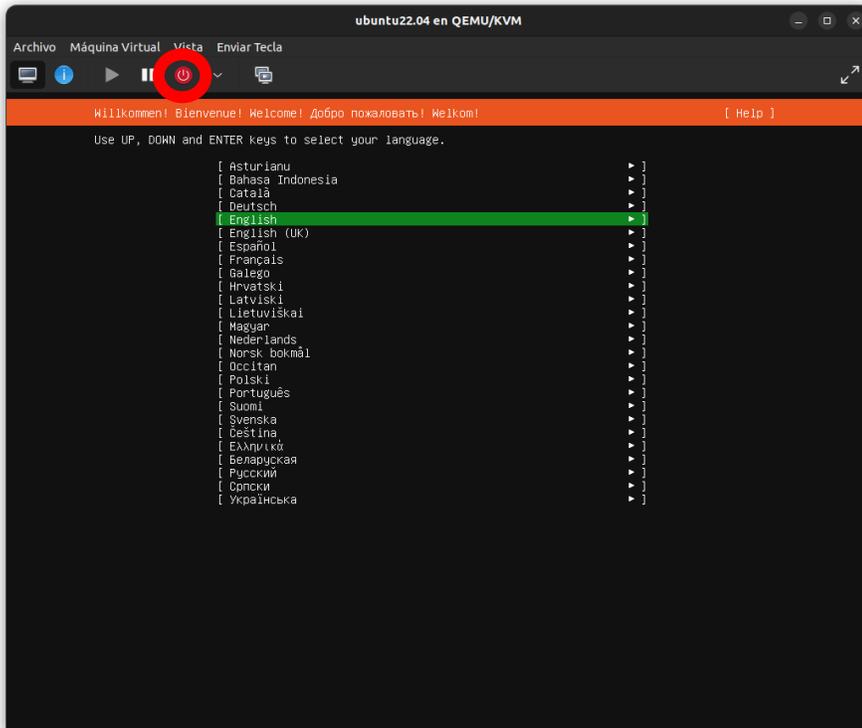


Damos click al botón de Finalizar, ya podremos iniciar nuestra máquina.

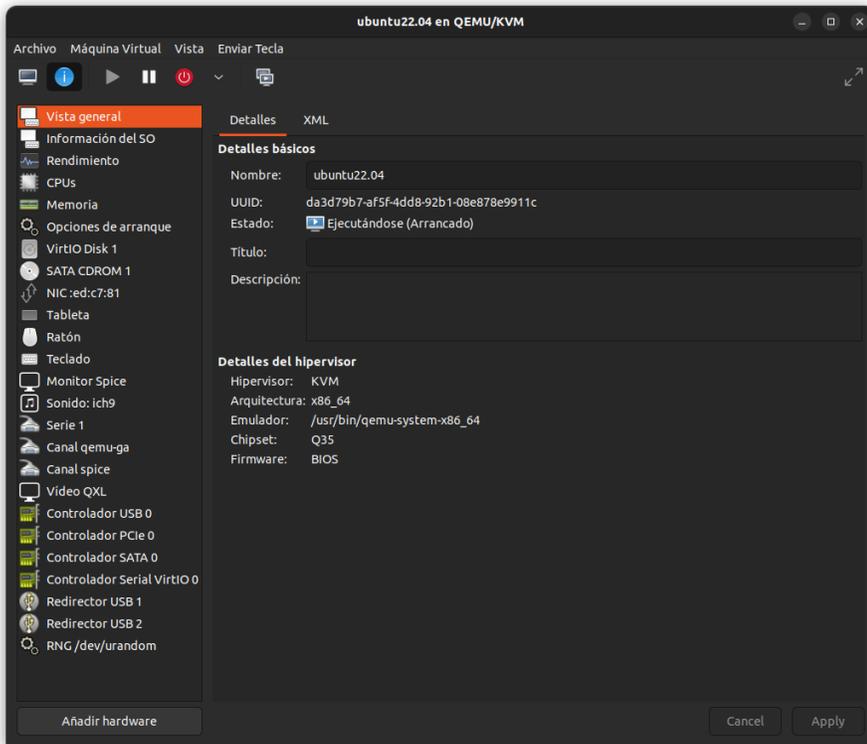
Automáticamente se muestra una nueva ventana con la interfaz de nuestra maquina:



Para apagar la máquina virtual damos click en el icono rojo de apagado.



Para revisar el recurso asignado podemos dar click al icono azul de detalles del hardware virtual.



Aquí podremos agregar hardware virtual necesario para el despliegue de las diferentes maquinas que se requieran. Como añadir una nueva Interfaz de Red Virtual para que la maquina tenga acceso a alguna subred en específico.

Para volver a correr la máquina virtual primero seleccionamos la maquina a iniciar, y damos click al icono de Play “Iniciar la máquina virtual”.



Para mostrar la ejecución de la maquina es necesario dar doble click sobre la máquina.

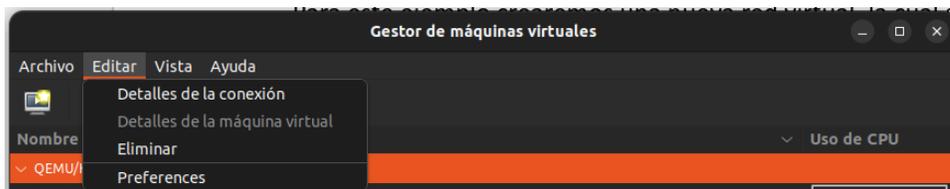
Pasos para añadir una nueva red virtual

Una nueva red virtual nos ayudara a acceder desde la maquina anfitrión con acceso ssh a la red especifica la cual queremos colocar. Esto ayudara a tener el monitoreo de las máquinas virtuales, y así proveerlas de internet para actualización, instalación, etc. de servicios necesarios.

Para este ejemplo crearemos una nueva red virtual, la cual será 192.168.100.0.

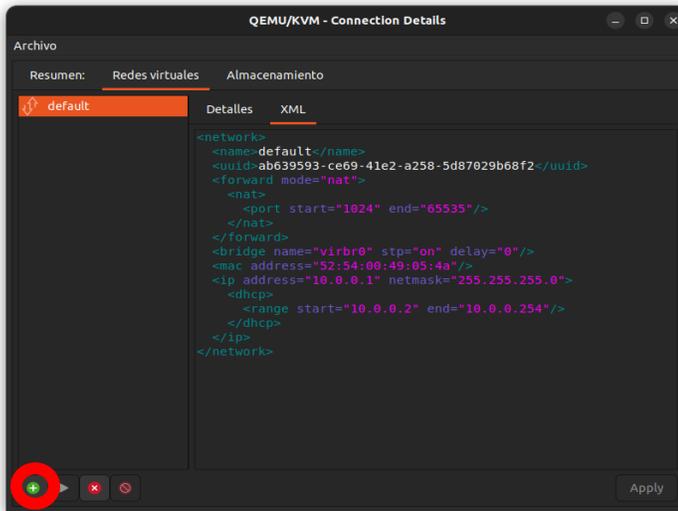
Cabe destacar que es necesario revisar que el router al que físicamente esté conectada la maquina anfitrión no tenga la misma red que crearemos para que no haya confusiones.

Para ello seleccionamos la opción de Editar en la barra superior, en la opción Detalles de conexión.



Nos vamos a la pestaña de Redes Virtuales, se mostrará el listado de Redes Virtuales que tengamos ya agregadas.

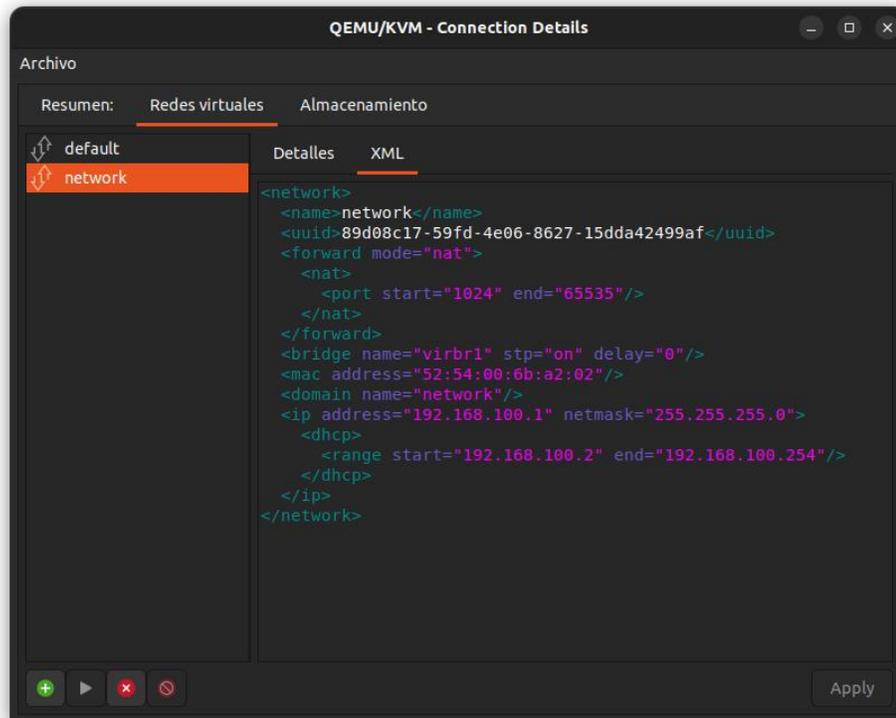
Para agregar una nueva red daremos click al botón verde + (Añadir Red) que se encuentra en la esquina inferior izquierda:



Hacemos las configuraciones correspondientes, y damos click en Finalizar.



Se mostrará la nueva red agregada, ahora ya podremos acceder mediante una conexión ssh con terminal de nuestra maquina anfitrión si la máquina virtual a la que queremos conectarnos tiene una interfaz conectada a esta red.



Configuración de Nodos

Listado de Nodos a configurar - Infraestructura

Nombre	SO	Disco (GiB)	RAM (MiB)	CPU
MaxScale	Ubuntu 22.04.3 Server LTS	20	2048	2
MariaDB1	Ubuntu 22.04.3 Server LTS	50	2048	2
MariaDB2	Ubuntu 22.04.3 Server LTS	50	2048	2
Keystone	Ubuntu 22.04.3 Server LTS	50	6144	4
OpenLDAP	Ubuntu 22.04.3 Server LTS	50	1024	1
Memcached	Ubuntu 22.04.3 Server LTS	50	4096	2
RabbitMQ	Ubuntu 22.04.3 Server LTS	50	4096	2

Modulo Clúster DB (Data Base)

Este módulo lo componen los nodos de MaxScale, MariaDB1 y MariaDB2.

Se han utilizado 3 redes internas diferentes, además se conectó el nodo MaxScale a una red VPN para brindar servicio de base de datos a las N Regiones:

Redes (Ipv4)	Descripción
192.168.100.0/24 InternetAccessNet	Esta red se utiliza para la conexión a Internet desde las máquinas virtuales. Proporciona acceso a recursos externos y servicios en la nube.
10.0.2.0/24 ClusterDBBalanceNet	Esta red conecta MaxScale con MariaDB1 y MariaDB2 para proporcionar balanceo de carga en el módulo de ClusterDB
10.0.3.0/24 ClusterDBGaleraNet	Red utilizada para el Galera Clúster en el módulo de ClusterDB. MariaDB1 y MariaDB2 se comunican a través de esta red para la sincronización del clúster
172.16.0.0/24 VPNNet	Esta red VPN conecta máquinas como Keystone, Memcached, RabbitMQ y MaxScale, permitiendo la vinculación de

	estos servicios en diferentes regiones de la nube comunitaria.
--	--

Lista de ip asignados a los Nodos para el Módulo Clúster DB

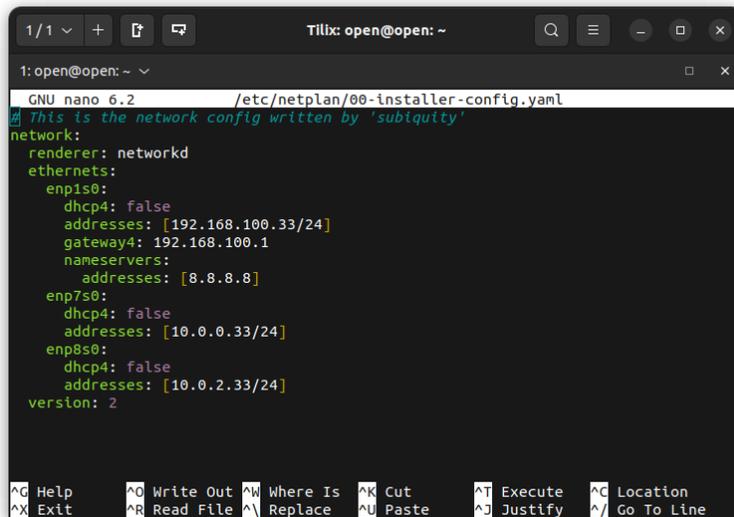
Nodo	IP's
MaxScale	192.168.100.33/24 10.0.0.33/24 10.0.2.33/24 172.16.0.31/24
MariaDB1	10.0.2.31/24 10.0.3.31/24
MariaDB2	10.0.2.32/24 10.0.3.32/24

Configuración de Interfaces

Tomaremos como ejemplo uno de los nodos, MaxScale. Editamos el siguiente archivo.

```
$ sudo nano /etc/netplan/00-installer-config.yaml
```

Realizamos la configuración de las interfaces.



```
GNU nano 6.2 /etc/netplan/00-installer-config.yaml
# This is the network config written by 'subiquity'
network:
  renderer: networkd
  ethernets:
    enp1s0:
      dhcp4: false
      addresses: [192.168.100.33/24]
      gateway4: 192.168.100.1
      nameservers:
        addresses: [8.8.8.8]
    enp7s0:
      dhcp4: false
      addresses: [10.0.0.33/24]
    enp8s0:
      dhcp4: false
      addresses: [10.0.2.33/24]
  version: 2
```

Guardamos la configuración, y luego ejecutamos el siguiente comando:

```
$ sudo netplan apply
```

Si todo sale correcto, podemos verificar las ips de las interfaces con el siguiente

comando:

```
$ ip a
```

```
Tilix: open@open: ~
1: open@open: ~
open@open:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen
1000
    link/ether 52:54:00:d9:a3:e4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.33/24 brd 192.168.100.255 scope global enp1s0
        valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fed9:a3e4/64 scope link
        valid_lft forever preferred_lft forever
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen
1000
    link/ether 52:54:00:1a:e0:9b brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.33/24 brd 10.0.0.255 scope global enp7s0
        valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe1a:e09b/64 scope link
        valid_lft forever preferred_lft forever
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen
1000
    link/ether 52:54:00:cd:62:9a brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.33/24 brd 10.0.2.255 scope global enp8s0
        valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fcd:629a/64 scope link
        valid_lft forever preferred_lft forever
```

Ya con esto podemos hacer la configuración respectiva de los demás nodos.

Cabe agregar que para mantener un mejor orden podemos cambiar el nombre host de la maquina con el siguiente comando.

```
$ sudo hostnamectl set-hostname maxscale
$ sudo exec bash
```

```
open@open:~$ sudo hostnamectl set-hostname maxscale
open@open:~$ exec bash
open@maxscale:~$
```

Agregar a los nodos MariaDB OpenStack 2023.1 Antelope for Ubuntu 22.04 LTS:

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

Instalación Clúster MariaDB

En los nodos de MariaDB:

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

Editar el siguiente archivo

```
# nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

Encontrar la línea siguiente en dicho archivo:

```
bind-address = 127.0.0.1
```

Cambiarla a:

```
bind-address = 0.0.0.0
```

Crear el siguiente archivo de configuración:

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

Escribir lo siguiente (este paso cambia según la ip correspondiente):

```
[mysqld]
query_cache_size=0
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
query_cache_type=0
bind-address=0.0.0.0
# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so
#wsrep_provider_options="gcache.size=32G"
# Galera Cluster Configuration
wsrep_cluster_name="ClusterMaria"
wsrep_cluster_address="gcomm://10.0.3.31,10.0.3.32"

# Galera Synchronization Congifuration
wsrep_sst_method=rsync
#wsrep_sst_auth=user:pass
# Galera Node Configuration
wsrep_node_address="10.0.2.31"
```

```
wsrep_node_name="maria1"
```

Nota: En caso de maria2, cambiar las líneas correspondientes:

```
wsrep_node_address="10.0.2.31"
```

```
wsrep_node_name="maria1"
```

Para crear el cluster con los dos nodos MariaDB ejecutamos el siguiente comando en uno de los nodos:

```
# galera_new_cluster
```

Luego ejecutamos el siguiente comando en el siguiente nodo de MariaDB:

```
# service mariadb restart
```

Ahora para verificar que el clúster se haya creado de manera correcta, ejecutamos el siguiente comando en cualquiera de los nodos MariaDB:

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

Se nos mostrara una salida como la siguiente en la consola:

```
+-----+-----+
| Variable_name          | Value |
+-----+-----+
| wsrep_cluster_size    | 2     |
+-----+-----+
```

Ahora cualquier comando ejecutado en uno de los nodos también se verá reflejado en el otro nodo.

Para la configuración de MaxScale:

```
apt install curl
curl -Ls https://r.mariadb.com/downloads/mariadb_repo_setup
| sudo bash
```

El repositorio de MariaDB debería estar configurado en tu sistema, y podrás instalar MariaDB utilizando el gestor de paquetes de tu sistema operativo. Ahora ejecutamos el siguiente comando para instalar MaxScale:

```
sudo apt install maxscale
```

Configuramos en cualquier de los nodos MariaDB el usuario de MaxScale, con los siguientes comandos:

```

MariaDB [(none)]> CREATE USER 'maxscale'@'%' IDENTIFIED BY
'iccl15';
MariaDB [(none)]> GRANT SELECT ON mysql.user TO
'maxscale'@'%' ;
MariaDB [(none)]> GRANT SELECT ON mysql.db TO 'maxscale'@'%' ;
MariaDB [(none)]> GRANT SELECT ON mysql.tables_priv TO
'maxscale'@'%' ;
MariaDB [(none)]> GRANT SELECT ON mysql.columns_priv TO
'maxscale'@'%' ;
MariaDB [(none)]> GRANT SELECT ON mysql.proxies_priv TO
'maxscale'@'%' ;
MariaDB [(none)]> GRANT SELECT ON mysql.procs_priv TO
'maxscale'@'%' ;
MariaDB [(none)]> GRANT SELECT ON mysql.roles_mapping TO
'maxscale'@'%' ;
MariaDB [(none)]> GRANT SHOW DATABASES ON *.* TO
'maxscale'@'%' ;
MariaDB [(none)]> GRANT REPLICATION CLIENT on *.* to
'maxscale'@'%' ;
MariaDB [(none)]> GRANT ALL ON infinidb_vtable.* TO
'maxscale'@'%' ;
MariaDB [(none)]> CREATE USER 'maxscale_monitor'@'%'
IDENTIFIED BY 'iccl15-monitor' ;
MariaDB [(none)]> GRANT REPLICATION CLIENT, FILE, SUPER,
RELOAD, PROCESS, SHOW DATABASES, EVENT ON *.* TO
'maxscale_monitor'@'%' ;

```

Configuramos el siguiente archivo: nano /etc/maxscale.cnf

```

# MaxScale documentation:
# https://mariadb.com/kb/en/mariadb-maxscale-6/
# Global parameters
#
# Complete list of configuration options:
# https://mariadb.com/kb/en/mariadb-maxscale-6-mariadb-
maxscale-configuration-guide/
[maxscale]
threads=auto
log_augmentation = 1
ms_timestamp = 1
syslog = 1
admin_host = 0.0.0.0
admin_secure_gui = false
# Server definitions
#
# Set the address of the server to the network
# address of a MariaDB server.
#

```

```
[maria1]
type=server
address=10.0.2.31
port=3306
protocol=MariaDBBackend
[maria2]
type=server
address=10.0.2.32
port=3306
protocol=MariaDBBackend
[Galera-Monitor]
type=monitor
module=galeramon
servers=maria1,maria2
user=maxscale_monitor
password=iccl15-monitor
monitor_interval=2s

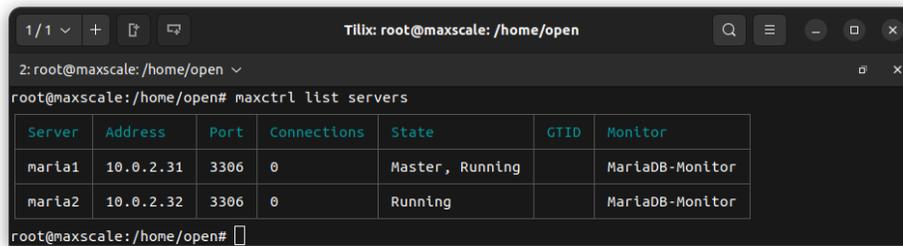
#ReadConnRoute documentation:
# https://mariadb.com/kb/en/mariadb-maxscale-6-readconnroute/
[Read-Con-Route-Galera-Service]
type=service
router=readconnroute
servers=maria1,maria2
user=maxscale
password=iccl15
[Read-Con-Route-Galera-Listener]
type=listener
service=Read-Con-Route-Galera-Service
protocol=MariaDBClient
port=3306
address=0.0.0.0
# ReadWriteSplit documentation:
# https://mariadb.com/kb/en/mariadb-maxscale-6-readwritesplit/
[Read-Con-Route-Galera-Listener]
type=listener
service=Read-Con-Route-Galera-Service
protocol=MariaDBClient
port=4008
address=0.0.0.0
```

Reiniciamos el servicio:

```
# service maxscale restart
```

Verificamos los servicios:

```
# maxctrl list servers
```



Server	Address	Port	Connections	State	GTID	Monitor
maria1	10.0.2.31	3306	0	Master, Running		MariaDB-Monitor
maria2	10.0.2.32	3306	0	Running		MariaDB-Monitor

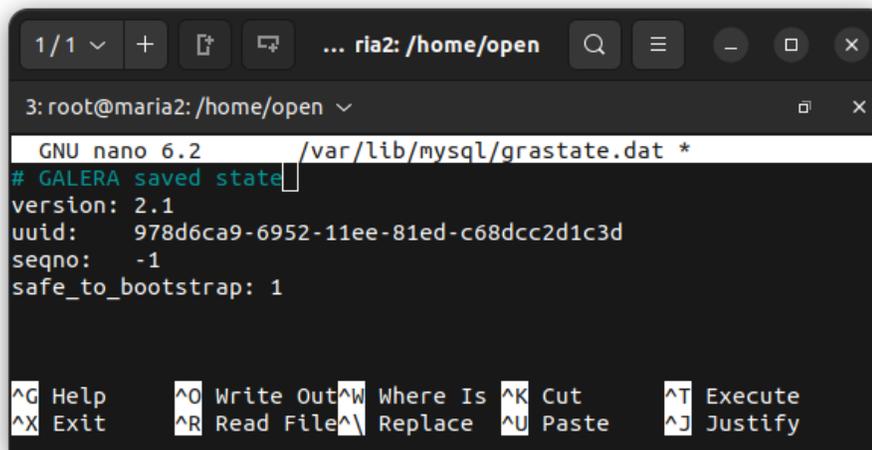
Con esta configuración lograremos conectarnos al balanceador de MaxScale en el puerto 4008

Recuperación de fallas

Si por algún motivo el clúster MariaDB no logra iniciar, se hace lo siguiente:

```
$ sudo nano /var/lib/mysql/grastate.dat
```

Editaremos este archivo en los nodos y pondremos 1 en `safe_to_bootstrap`, de esta manera y lo guardaremos:



```
GNU nano 6.2 /var/lib/mysql/grastate.dat *
# GALERA saved state
version: 2.1
uuid: 978d6ca9-6952-11ee-81ed-c68dcc2d1c3d
seqno: -1
safe_to_bootstrap: 1

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute
^X Exit      ^R Read File ^\ Replace  ^U Paste    ^J Justify
```

Ejecutamos el siguiente comando en todos los nodos que tengan MariaDB y estén en el cluster a recuperar:

```
# galera_recovery
```

Mostrará el `uuid`, y luego solo ejecutaremos lo siguiente en el nodo que tenga un `uuid` más alto que los demás (si hay varios iguales se toma cualquiera)

```
# galera_new_cluster
```

Luego reiniciaremos el servicio de MariaDB en los nodos restantes:

```
# service mariadb restart
```

Mostramos el status del servicio:

```
# service mariadb status
```

```
Tilix: root@maria2: /home/open
3:root@maria2: /home/open
● mariadb.service - MariaDB 10.6.12 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2023-11-19 06:12:28 UTC; 28min ago
     Docs: man:mariadb(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 1381 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run/mysqld (code=exited, status=0/SUCCESS)
   Process: 1382 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)
   Process: 1384 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR= || VAR=cd /usr/bin/./; /usr/bin/galera_re
   Process: 1652 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)
   Process: 1654 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCCESS)
  Main PID: 1467 (mariadb)
    Status: "Taking your SQL requests now..."
     Tasks: 13 (limit: 2191)
    Memory: 104.9M
       CPU: 3.642s
    CGroup: /system.slice/mariadb.service
           └─1467 /usr/sbin/mariadb --wsrep_start_position=978d6ca9-6952-11ee-81ed-c68dcc2d1c3d:189906

nov 19 06:12:28 maria2 /etc/mysql/debian-start[1656]: Upgrading MySQL tables if necessary.
nov 19 06:12:28 maria2 /etc/mysql/debian-start[1659]: Looking for 'mariadb' as: /usr/bin/mariadb
nov 19 06:12:28 maria2 /etc/mysql/debian-start[1659]: Looking for 'mariadb-check' as: /usr/bin/mariadb-check
nov 19 06:12:28 maria2 /etc/mysql/debian-start[1659]: This installation of MariaDB is already upgraded to 10.6.12-MariaDB.
nov 19 06:12:28 maria2 /etc/mysql/debian-start[1659]: There is no need to run mysql upgrade again for 10.6.12-MariaDB.
nov 19 06:12:28 maria2 /etc/mysql/debian-start[1659]: You can use --force if you still want to run mysql_upgrade
nov 19 06:12:28 maria2 /etc/mysql/debian-start[1667]: Checking for insecure root accounts.
nov 19 06:12:28 maria2 /etc/mysql/debian-start[1671]: Triggering myisam-recover for all MyISAM tables and aria-recover for all A
nov 19 06:12:28 maria2 mariadb[1467]: 2023-11-19  6:12:28 0 [Note] WSREP: (9bc7cb31-b1de, 'tcp://0.0.0.0:4567') turning message
nov 19 06:30:22 maria2 mariadb[1467]: 2023-11-19  6:30:22 37 [Warning] IP address '10.0.2.33' could not be resolved: Name or se
~
[lines 1-28/28 (END)]
```

Modulo Controller

Este módulo lo componen los nodos Memcached y RabbitMQ.

Se han utilizado 2 redes internas, además cada uno de los dos nodos tienen una conexión VPN para brindar servicios a las Regiones de la nube.

Redes (Ipv4)	Descripción
192.168.100.0/24 InternetAccessNet	Esta red se utiliza para la conexión a Internet desde las máquinas virtuales. Proporciona acceso a recursos externos y servicios en la nube.
10.0.0.0/24 ClusterDBBalanceNet	Esta red interconecta los módulos de Identity, Controller y ClusterDB en la infraestructura.
172.16.0.0/24 VPNNet	Esta red VPN conecta máquinas como Keystone, Memcached, RabbitMQ y MaxScale, permitiendo la vinculación de estos servicios en diferentes regiones de la nube comunitaria.

Lista de ip asignados a los Nodos para el Módulo Clúster DB

Nodo	IP's
Memcached	192.168.100.21/24 10.0.0.21/24 172.16.0.21/24
RabbitMQ	192.168.100.22/24 10.0.0.22/24 172.16.0.22/24

Configuramos las interfaces correspondientes.

Enviroment

Instalación Chrony

Para esta instalación la haremos primero en el nodo de Memcached.

```
# apt install chrony
```

Editamos el archivo de configuración de Chrony.

```
$ sudo nano /etc/chrony/chrony.conf
```

Agregamos el siguiente server ntp:

```
server ntp.ues.edu.sv iburst
```

Agregamos lo siguiente en el mismo archivo al final:

```
# Permite el servicio de Chrony en las redes siguientes  
# Interno  
allow 10.0.0.0/24  
# VPN  
allow 172.16.0.0/24
```


Instalación Memcached

Ejecutamos el siguiente comando:

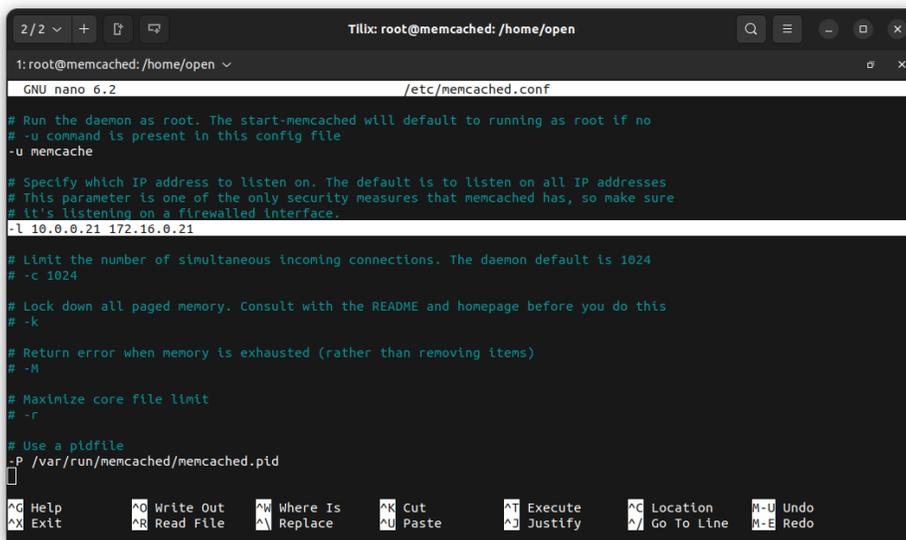
```
# apt install memcached python3-memcache
```

Editamos el archivo de configuración de memcached:

```
$ sudo nano /etc/memcached.conf
```

Agregamos la siguiente línea:

```
-l 10.0.0.21 172.16.0.21
```



The screenshot shows a terminal window titled 'Tilix: root@memcached: /home/open'. Inside, the nano editor is open to the file '/etc/memcached.conf'. The configuration file content is as follows:

```
GNU nano 6.2 /etc/memcached.conf
# Run the daemon as root. The start-memcached will default to running as root if no
# -u command is present in this config file
-u memcache

# Specify which IP address to listen on. The default is to listen on all IP addresses.
# This parameter is one of the only security measures that memcached has, so make sure
# it's listening on a firewalled interface.
-l 10.0.0.21 172.16.0.21

# Limit the number of simultaneous incoming connections. The daemon default is 1024
# -c 1024

# Lock down all paged memory. Consult with the README and homepage before you do this
# -k

# Return error when memory is exhausted (rather than removing items)
# -M

# Maximize core file limit
# -f

# Use a pidfile
-P /var/run/memcached/memcached.pid
```

Reiniciamos el servicio Memcached:

```
# service memcached restart
```

Instalación Etcd

Instalamos este servicio en el nodo Memcached.

```
# apt install etcd
```

Configuramos el siguiente archivo /etc/default/etcd

```
ETCD_NAME="controller"
ETCD_DATA_DIR="/var/lib/etcd"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
ETCD_INITIAL_CLUSTER="controller=http://10.0.0.21:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://10.0.0.21:2380"
```

```
ETCD_ADVERTISE_CLIENT_URLS="http://10.0.0.21:2379"  
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"  
ETCD_LISTEN_CLIENT_URLS="http://10.0.0.21:2379"
```

Guardamos el archivo y reiniciamos el servicio:

```
# systemctl enable etcd  
# systemctl restart etcd
```

Instalación RabbitMQ

Ejecutamos el siguiente comando en el nodo RabbitMQ:

```
# apt install rabbitmq-server
```

Agregamos el usuario Openstack:

```
# rabbitmqctl add_user openstack RABBIT_PASS
```

Permitimos accesos:

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

Agregando nuevas colas para cada Región:

```
$ sudo rabbitmqctl add_user region PASS  
$ sudo rabbitmqctl add_vhost /region  
$ sudo rabbitmqctl set_permissions -p /region region ".*"  
".*" ".*"
```

Esto permitirá ocupar una cola para una Región en específico. [\[OBJ\]](#)

Habilitar UI WEB

Esto nos permitirá acceder una interfaz web donde podremos administrar de mejor manera el servicio de RabbitMQ. Ejecutamos el siguiente comando.

```
$ sudo rabbitmqctl add_user USER PASS
```

Con esto agregaremos un usuario. Ejecutamos los siguientes comandos para darle permisos:

```
$ rabbitmqctl add_vhost /VHOST  
$ sudo rabbitmqctl set_permissions -p /VHOST USER ".*" ".*"  
".*" ".*"  
$ sudo rabbitmqctl set_user_tags main administrator
```

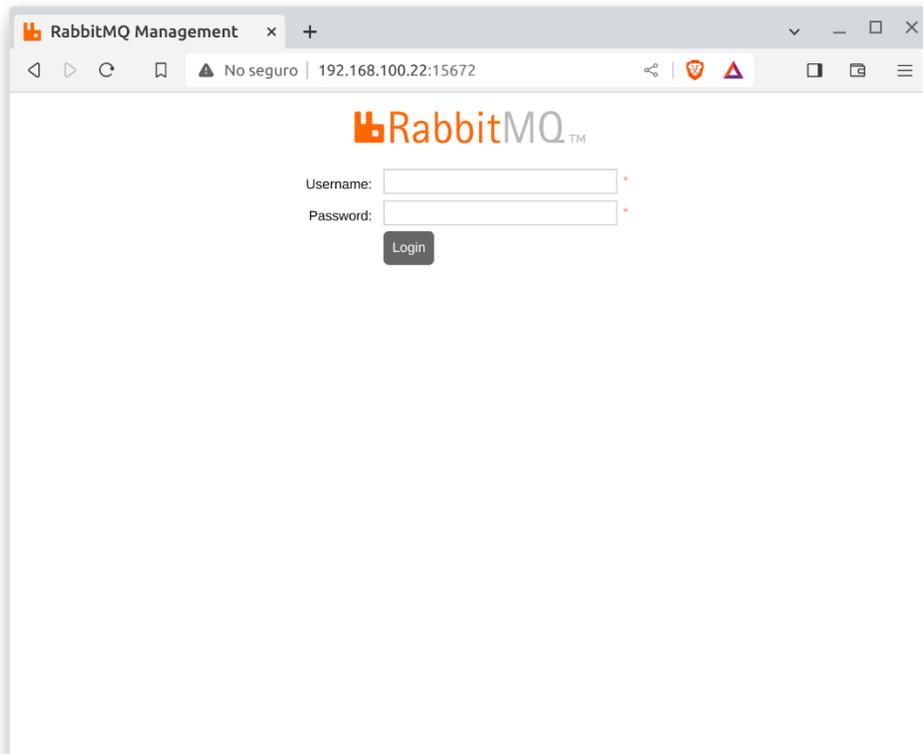
Habilitamos el ambiente web

```
$ rabbitmq-plugins enable rabbitmq_management
```

Para ingresar en un navegador escribimos la siguiente URL:

```
http://192.168.100.22:15672/
```

Se nos mostrara lo siguiente:



Ingresamos con las credenciales que creamos:

RabbitMQ Management x +

No seguro | 192.168.100.22:15672/#/

Refreshed 2023-11-27 16:22:42 Refresh every 5 seconds

RabbitMQ™ RabbitMQ 3.9.13 Erlang 24.2.1

Virtual host All

Cluster rabbit@rabbitmq

User main Log out

Overview Connections Channels Exchanges Queues Admin

Overview

▼ Totals

Queued messages last minute ?

Currently idle

Message rates last minute ?

Currently idle

Global counts ?

Connections: 0 Channels: 0 Exchanges: 14 Queues: 0 Consumers: 0

▼ Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats	+/-
rabbit@rabbitmq	39 65536 available	0 58893 available	370 1048576 available	129 MiB 1.5 GiB high watermark	37 GiB 48 MiB low watermark	13h 40m	basic disc 1 rss	This node All nodes	

▼ Churn statistics

Connection operations last minute ?

1.0 /s

Created 0.00/s

Modulo Entity

Este módulo lo componen los nodos Keystone y OpenLDAP.

Se han utilizado 3 redes internas, además el nodo Keystone tiene una conexión VPN para brindar servicios a las Regiones de la nube de autenticación.

Redes (Ipv4)	Descripción
192.168.100.0/24 InternetAccessNet	Esta red se utiliza para la conexión a Internet desde las máquinas virtuales. Proporciona acceso a recursos externos y servicios en la nube.
10.0.0.0/24 ClusterDBBalanceNet	Esta red interconecta los módulos de Identity, Controller y ClusterDB en la infraestructura.
10.0.1.0/24 IdentityInternalNet	Red interna utilizada por el módulo de Identity, que incluye Keystone y OpenLDAP. Facilita la comunicación interna entre estos servicios.
172.16.0.0/24 VPNNet	Esta red VPN conecta máquinas como Keystone, Memcached, RabbitMQ y MaxScale, permitiendo la vinculación de estos servicios en diferentes regiones de la nube comunitaria.

Lista de ip asignados a los Nodos para el Módulo Clúster DB

Nodo	IP's
Keystone	192.168.100.11/24 10.0.0.11/24 10.0.1.11/24 172.16.0.11/24
OpenLDAP	10.0.1.11/24

Configuramos las interfaces correspondientes.

Instalación Keystone

OpenStack packages for Ubuntu

Agregar al nodo Keystone. OpenStack 2023.1 Antelope for Ubuntu 22.04 LTS:

```
# add-apt-repository cloud-archive:antelope
# apt update
```

Instalamos el cliente de OpenStack:

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

En cualquier nodo del clúster de MariaDB agregamos la siguiente Base de Datos par

Keystone:

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

Cambiamos KEYSTONE_DBPASS por el password que quiera agregarse a la DB.

Instalamos Keystone en el nodo correspondiente:

```
# apt install keystone
```

Configuramos el siguiente archivo de Keystone:

```
$ sudo nano /etc/keystone/keystone.conf
```

Agregamos la siguiente, según la sección correspondiente.

```
[database]
# Para este ejemplo la conexión sería
# connection =
mysql+pymysql://keystone:iccl15@10.0.0.33:4008/keystone
connection =
mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone

[token]
# ...
provider = fernet
```

Poblamos la base de datos con:

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

Iniciamos los servicios Fernet:

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

Agregamos las credenciales de la primera region RegionOne:

```
# keystone-manage bootstrap --bootstrap-password icc115 \  
--bootstrap-admin-url http://172.16.0.11:5000/v3/ \  
--bootstrap-internal-url http://172.16.0.11:5000/v3/ \  
--bootstrap-public-url http://172.16.0.11:5000/v3/ \  
--bootstrap-region-id RegionOne
```

Cambiamos ADMIN_PASS por alguna contraseña, en este ejemplo utilizamos icc115.

Configuramos el servicio de Apache HTTP Server:

```
$ sudo nano /etc/apache2/apache2.conf
```

Agregamos lo siguiente:

```
ServerName controller
```

Reiniciamos el servicio:

```
# service apache2 restart
```

Creamos el siguiente archivo de autenticacion:

```
$ sudo nano admin-openrc
```

Agregamos lo siguiente:

```
export OS_PROJECT_DOMAIN_NAME=Default  
export OS_USER_DOMAIN_NAME=Default  
export OS_PROJECT_NAME=admin  
export OS_USERNAME=admin  
export OS_PASSWORD=icc115  
export OS_AUTH_URL=http://172.16.0.11:5000/v3  
export OS_IDENTITY_API_VERSION=3  
export OS_IMAGE_API_VERSION=2  
export OS_REGION_NAME = RegionOne
```

Ahora podemos ejecutar el archivo de la siguiente manera:

```
$ . admin-openrc
```

Estamos autenticados.

Ahora crearemos el dominio y el servicio nuevo:

```
# openstack domain create --description "An Example Domain"
example
# openstack project create --domain default \
--description "Service Project" service
```

Instalación OpenLDAP

Actualizamos el sistema (apt update). Ejecutamos el siguiente comando:

```
$ sudo hostnamectl set-hostname ldap.identity.net
```

Instalamos slapd y ldap-utils

```
$ sudo apt install slapd ldap-utils
```

Nos pedirá una contraseña.

Configuramos nuestro archivo hosts:

```
$ sudo nano /etc/hosts
```

Agregamos la siguiente línea:

```
10.0.0.12 ldap.identity.net ldap
```

Hacemos las configuraciones pertinentes:

```
$ sudo dpkg-reconfigure slapd
```

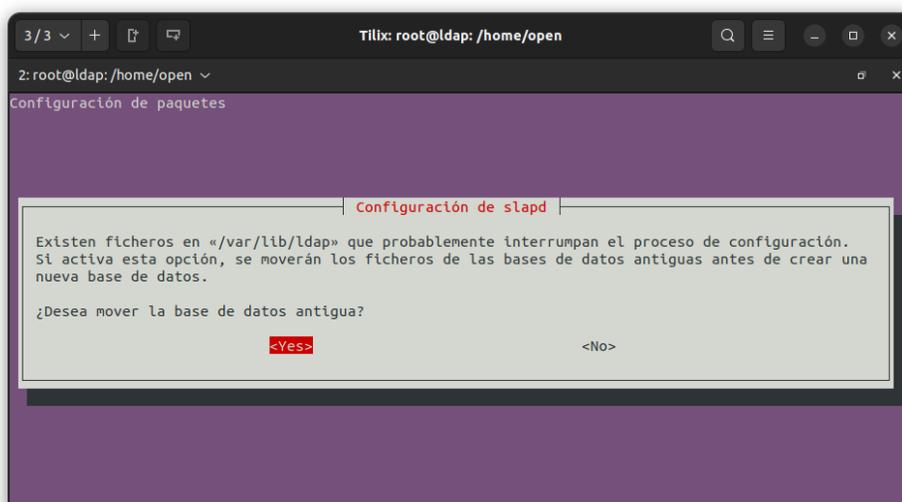
Seleccionamos que no:



Dejamos como esta:



Nos preguntara si deseamos mover la base de datos antigua, elegimos que sí.



Reiniciamos el servicio:

```
$ sudo systemctl restart slapd
```

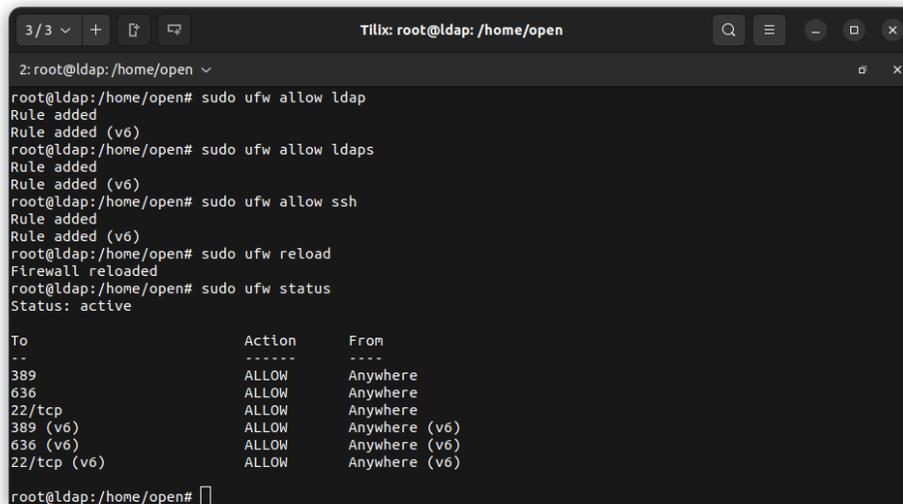
Activamos el servicio ufw:

```
$ sudo ufw enable
```

Ejecutamos los siguientes comandos para habilitar los distintos puertos necesarios:

```
$ sudo ufw allow ldap
$ sudo ufw allow ldaps
$ sudo ufw allow ssh
$ sudo ufw reload
```

```
$ sudo ufw status
```



```
Tilix: root@ldap: /home/open
2: root@ldap: /home/open
root@ldap: /home/open# sudo ufw allow ldap
Rule added
Rule added (v6)
root@ldap: /home/open# sudo ufw allow ldaps
Rule added
Rule added (v6)
root@ldap: /home/open# sudo ufw allow ssh
Rule added
Rule added (v6)
root@ldap: /home/open# sudo ufw reload
Firewall reloaded
root@ldap: /home/open# sudo ufw status
Status: active

To Action From
--
389 ALLOW Anywhere
636 ALLOW Anywhere
22/tcp ALLOW Anywhere
389 (v6) ALLOW Anywhere (v6)
636 (v6) ALLOW Anywhere (v6)
22/tcp (v6) ALLOW Anywhere (v6)
root@ldap: /home/open#
```

Creacion de Arbol de Direcciones

Creamos el archivo tree.ldif, y agregamos lo siguiente:

```
dn: cn=accounts,dc=identity,dc=net
objectClass: organizationalRole
cn: accounts

dn: cn=users,cn=accounts,dc=identity,dc=net
objectClass: organizationalRole
cn: users

dn: cn=groups,cn=accounts,dc=identity,dc=net
objectClass: organizationalRole
cn: groups
```

Ejecutamos el siguiente comando para agregarlo al servicio:

```
# ldapadd -x -D "cn=admin,dc=identity,dc=net" -W -f tree.ldif
```

Nota: Se pedirá la contraseña que agregamos al momento de hacer la reconfiguración.

Creacion de Usuario Lookup

Creamos el archivo usuario.ldif, y agregamos lo siguiente:

Nota: Para generar una contraseña encriptada para los usuarios debemos ejecutar lo

siguiente `slappasswd -s PASSWORD`

Para este ejemplo la contraseña es `icc115`.

```
dn: uid=svc-ldap,cn=users,cn=accounts,dc=identity,dc=net
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: svc-ldap
sn: svc-ldap
cn: Service LDAP
userPassword: {SSHA}vYeyCBOSo2W5XwLU2LGAin/xF5WppAU8

dn: cn=grp-openstack,cn=groups,cn=accounts,dc=identity,dc=net
objectClass: top
objectClass: groupOfNames
cn: grp-openstack
member: uid=svc-ldap,cn=users,cn=accounts,dc=identity,dc=net
```

Luego ejecutamos el siguiente comando:

```
# ldapadd -x -D "cn=admin,dc=identity,dc=net" -W -f
usuario.ldif
```

Creación de Usuarios Normales

Creamos un archivo llamado usuarios.ldif, y agregaremos estos dos usuarios:

```
dn: uid=johndoe,cn=users,cn=accounts,dc=identity,dc=net
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: johndoe
cn: John Doe
sn: Doe
givenName: John
mail: johndoe@example.com
userPassword: {SSHA}vYeyCBOSo2W5XwLU2LGAin/xF5WppAU8

dn: uid=janedoe,cn=users,cn=accounts,dc=identity,dc=net
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: janedoe
cn: Jane Doe
sn: Doe
givenName: Jane
mail: janedoe@example.com
userPassword: {SSHA}vYeyCBOSo2W5XwLU2LGAin/xF5WppAU8
```

Luego ejecutamos el siguiente comando:

```
# ldapadd -x -D "cn=admin,dc=identity,dc=net" -W -f
usuario.ldif
```

Configuración del Certificado

Al final del archivo de configuración de ldap está la ruta del certificado. En el caso de Ubuntu 22.04 la ruta es /etc/ssl/certs/ca-certificates.crt, puede depender del sistema operativo. Lo copiamos al servidor donde esta Keystone. Se puede hacer por el comando scp.

```
# scp /etc/ssl/certs/ca-certificates.crt
USUARIO@IP_KEYSTONE_SERVER:/home/USUARIO
```

Para ello debemos habilitar la conexión ssh de esta manera.

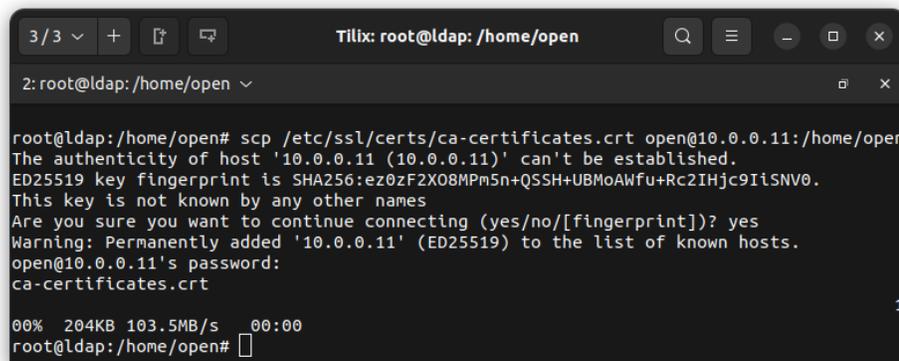
Nos vamos al archivo /etc/ssh/sshd_config, y cambiamos la siguiente línea:

```
PermitRootLogin without-password
#Cambiarla por
PermitRootLogin yes
```

Reiniciamos el servicio con:

```
# service ssh restart
```

Ya con eso podemos ejecutar el comando para copiar el certificado a Keystone:



```
Tilix: root@ldap: /home/open
2: root@ldap: /home/open
root@ldap: /home/open# scp /etc/ssl/certs/ca-certificates.crt open@10.0.0.11:/home/open
The authenticity of host '10.0.0.11 (10.0.0.11)' can't be established.
ED25519 key fingerprint is SHA256:ez0zF2X08MPm5n+QSSH+UBMoAWfu+Rc2IHjc9IiSNV0.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.0.11' (ED25519) to the list of known hosts.
open@10.0.0.11's password:
ca-certificates.crt
1
00% 204KB 103.5MB/s 00:00
root@ldap: /home/open#
```

Una vez en la carpeta del usuario en el servidor de Keystone generar el archivo .pem

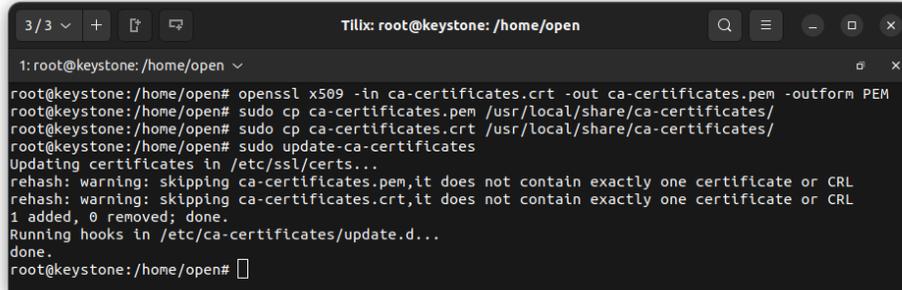
```
# openssl x509 -in ca-certificates.crt -out ca-
certificates.pem -outform PEM
```

Mover ambos archivos al directorio de llaves:

```
# sudo cp ca-certificates.pem /usr/local/share/ca-  
certificates/  
# sudo cp ca-certificates.crt /usr/local/share/ca-  
certificates/
```

Actualizar certificados

```
# sudo update-ca-certificates
```



```
Tilix: root@keystone: /home/open  
1:root@keystone: /home/open  
root@keystone: /home/open# openssl x509 -in ca-certificates.crt -out ca-certificates.pem -outform PEM  
root@keystone: /home/open# sudo cp ca-certificates.pem /usr/local/share/ca-certificates/  
root@keystone: /home/open# sudo cp ca-certificates.crt /usr/local/share/ca-certificates/  
root@keystone: /home/open# sudo update-ca-certificates  
Updating certificates in /etc/ssl/certs...  
rehash: warning: skipping ca-certificates.pem, it does not contain exactly one certificate or CRL  
rehash: warning: skipping ca-certificates.crt, it does not contain exactly one certificate or CRL  
1 added, 0 removed; done.  
Running hooks in /etc/ca-certificates/update.d...  
done.  
root@keystone: /home/open#
```

Configuración Nuevo Dominio

En el nodo de Keystone, se crea un nuevo directorio para las configuraciones del nuevo dominio.

```
$ sudo mkdir /etc/keystone/domains/  
$ sudo chown keystone /etc/keystone/domains/
```

Al archivo de configuración de keystone (/etc/keystone/keystone.conf) agregamos nuevas configuraciones

```
[assignment]  
driver = sql  
[identity]  
domain_specific_drivers_enabled = true  
domain_config_dir = /etc/keystone/domains
```

Se crea el nuevo dominio con la CLI de openstack (Recordemos habernos logueado)

```
# openstack domain create LAB
```

```
3/3 v + [?] [?] ... stone: /home/open Q ☰ - □ ×
1: root@keystone: /home/open v
root@keystone:/home/open# openstack domain create LAB
+-----+-----+
| Field | Value |
+-----+-----+
| description | |
| enabled | True |
| id | b0b50fe378644cf4bca44cdd89869d05 |
| name | LAB |
| options | {} |
| tags | [] |
+-----+-----+
root@keystone:/home/open#
```

Se crea un archivo para la configuración de la conexión con ldap del nuevo dominio:

/etc/keystone/domains/keystone.LAB.conf Debe tener el nombre del dominio creado

anteriormente. Se agrega lo siguiente:

```
[ldap]
url = ldap://10.0.0.12
user = uid=svc-ldap,cn=users,cn=accounts,dc=identity,dc=net
password = icc115
user_tree_dn = cn=users,cn=accounts,dc=identity,dc=net
user_objectclass = inetOrgPerson
user_id_attribute = uid
user_name_attribute = uid
user_mail_attribute = mail
user_pass_attribute =
user_allow_create = False
user_allow_update = False
user_allow_delete = False
tls_cacertfile = /usr/local/share/ca-certificates/ca-
certificates.crt

[identity]
driver = ldap
```

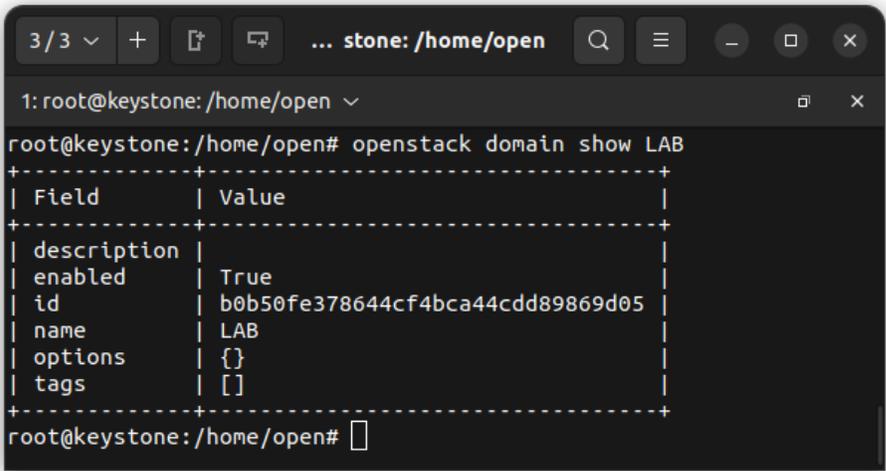
Cambiamos el propietario del archivo

```
$ sudo chown keystone /etc/keystone/domains/keystone.LAB.conf
```

Acceso al nuevo dominio para el usuario admin

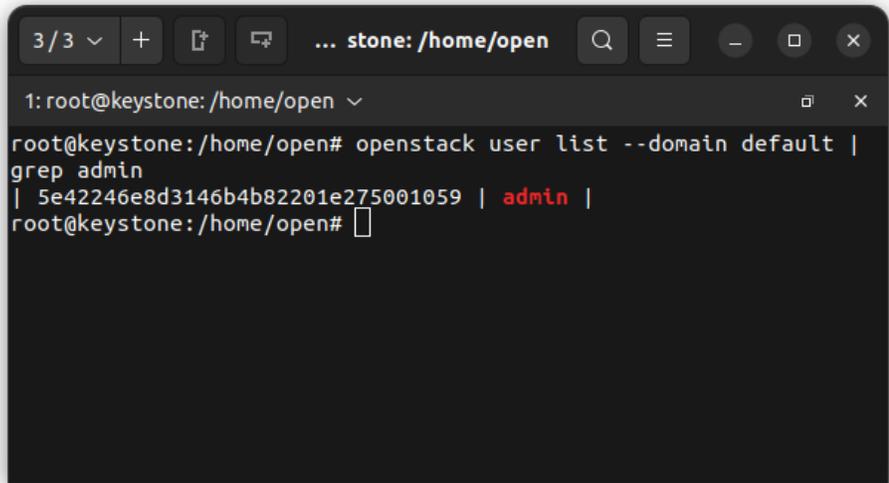
Se otorga acceso al usuario admin al dominio LAB. Para lo que necesitamos algunos datos ID del dominio LAB

```
# openstack domain show LAB
```



ID del usuario admin

```
# openstack user list --domain default | grep admin
```



ID del rol admin

```
# openstack role list
```

```
3/3 v + [ ] [ ] ... stone: /home/open Q [ ] - [ ] x
1: root@keystone: /home/open v [ ] x
root@keystone:/home/open# openstack role list
+-----+-----+
| ID | Name |
+-----+-----+
| 35f201a42b5f4c15934531c99976b51d | reader |
| 85a1db599eba44788ba9f7076039b13d | admin |
| d41d78d3dc4c45c08d1424cb53ffd37a | member |
+-----+-----+
root@keystone:/home/open# [ ]
```

Ahora si asignamos el rol de admin sobre el dominio, al usuario admin

```
# openstack role add --domain ID_DOMAIN --user ID_USER_ADMIN
ID_ROLE_ADMIN
```

Quedaría así:

```
# openstack role add --domain
b0b50fe378644cf4bca44cdd89869d05 --user
5e42246e8d3146b4b82201e275001059
85a1db599eba44788ba9f7076039b13d
```

Reiniciar el servicio de apache

```
# service apache2 restart
```

Listamos los usuarios que están en el LDAP

```
# openstack user list --domain LAB
```

Si un dado caso nos da un error, quizás sea necesario instalar python, luego se podrá ejecutar el comando

```
# apt install python3-pip
# apt update
```

```
Tilix: root@keystone: /home/open
1: root@keystone: /home/open
root@keystone: /home/open# openstack user list --domain LAB
+-----+-----+
| ID | Name |
+-----+-----+
| c8d38f7e81ff6d86367bf9c477aefdab08f092fae6b38bb6d244628985d26b78 | svc-ldap |
| 186beebce4725aeab5d1b0e2c6b5e2dcc1d0c2348a0f23d84b4addc98f078115 | johndoe |
| 43bbf55baa47d2f99d07f332680bcb758e3c18378f65d85ef7f99cef8ac0482c | janedoe |
+-----+-----+
root@keystone: /home/open#
```

Dando permisos a usuarios del ldap a ver el panel de openstack

Necesitamos el ID del usuario, así que seleccionamos uno de la lista

```
# openstack user list --domain LAB
```

```
Tilix: root@keystone: /home/open
1: root@keystone: /home/open
root@keystone: /home/open# openstack user list --domain LAB
+-----+-----+
| ID | Name |
+-----+-----+
| c8d38f7e81ff6d86367bf9c477aefdab08f092fae6b38bb6d244628985d26b78 | svc-ldap |
| 186beebce4725aeab5d1b0e2c6b5e2dcc1d0c2348a0f23d84b4addc98f078115 | johndoe |
| 43bbf55baa47d2f99d07f332680bcb758e3c18378f65d85ef7f99cef8ac0482c | janedoe |
+-----+-----+
root@keystone: /home/open#
```

Ya que tenemos el ID le agregamos un rol, ya sea admin, member, reader dependiendo los roles que tengamos. En este caso otorgamos el rol member en el dominio default

```
# openstack role add --user ID_JANE --domain default member
```

Quedaría de esta manera:

```
# openstack role add --user
43bbf55baa47d2f99d07f332680bcb758e3c18378f65d85ef7f99cef8ac04
82c --domain default member
```

Ahora tiene permisos sobre el dominio default, pero no sobre un proyecto del dominio, así que hay que darle permisos sobre un proyecto, en este ejemplo proyecto admin

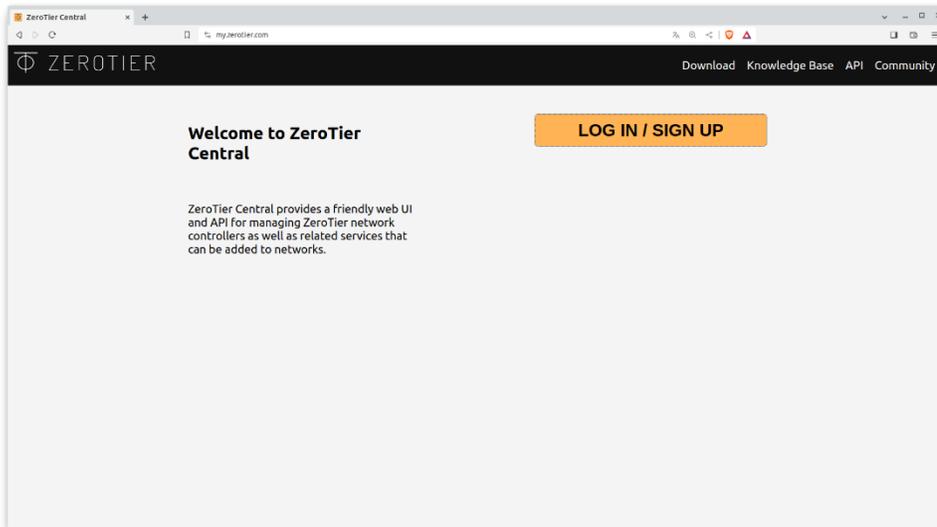
```
# openstack role add --user  
43bbf55baa47d2f99d07f332680bcb758e3c18378f65d85ef7f99cef8ac04  
82c --project admin member
```

Ahora ya podremos iniciar sesión en el panel de Horizon con su usuario, contraseña y el dominio LAB.

Instalación VPN

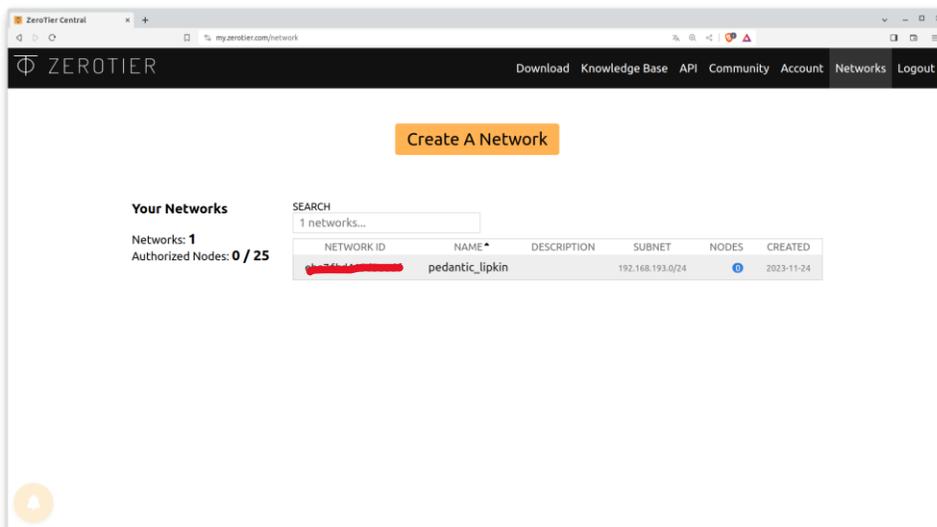
Para esta instalación hemos tomado un servidor externo el cual nos ayudará a conectar nuestros servicios principales a las regiones que agreguemos a nuestra infraestructura. Para ello hemos utilizado ZeroTier un servidor VPN externo.

Ingresamos a la página de ZeroTier URL: <https://my.zerotier.com/>



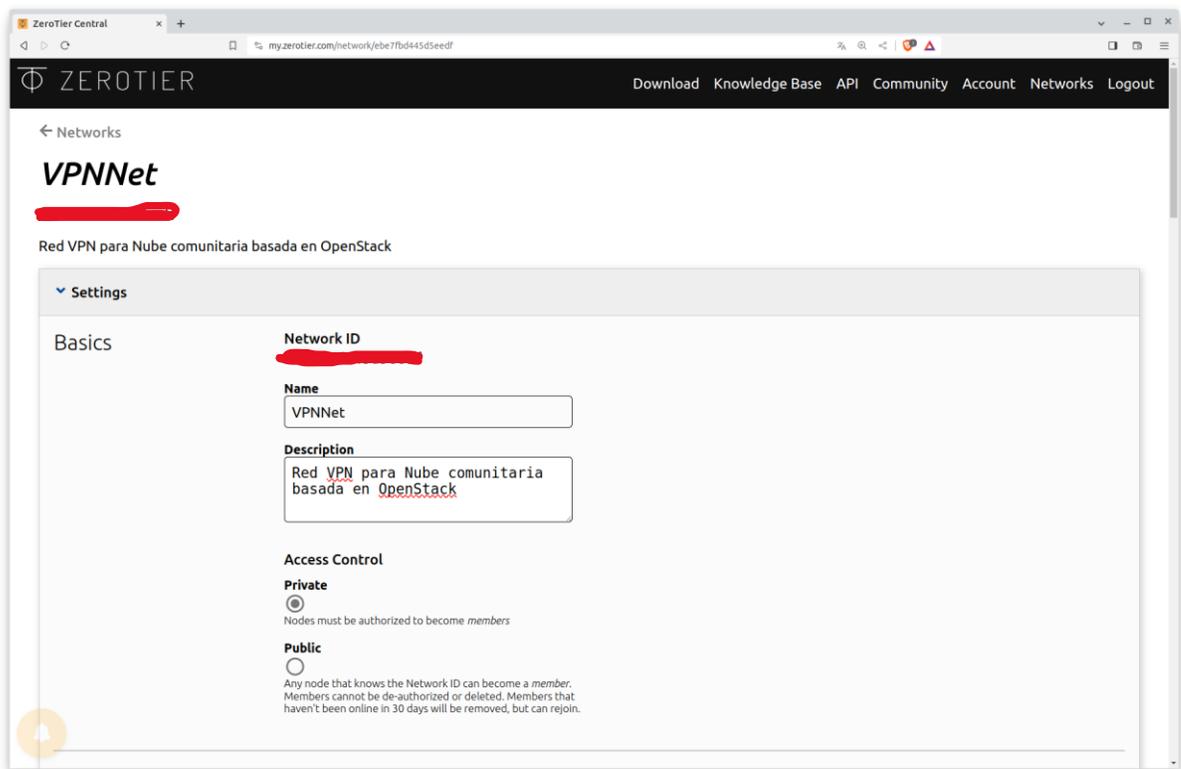
Podemos crear una cuenta gratuita con ciertas restricciones.

Al ingresar creamos una red nueva:

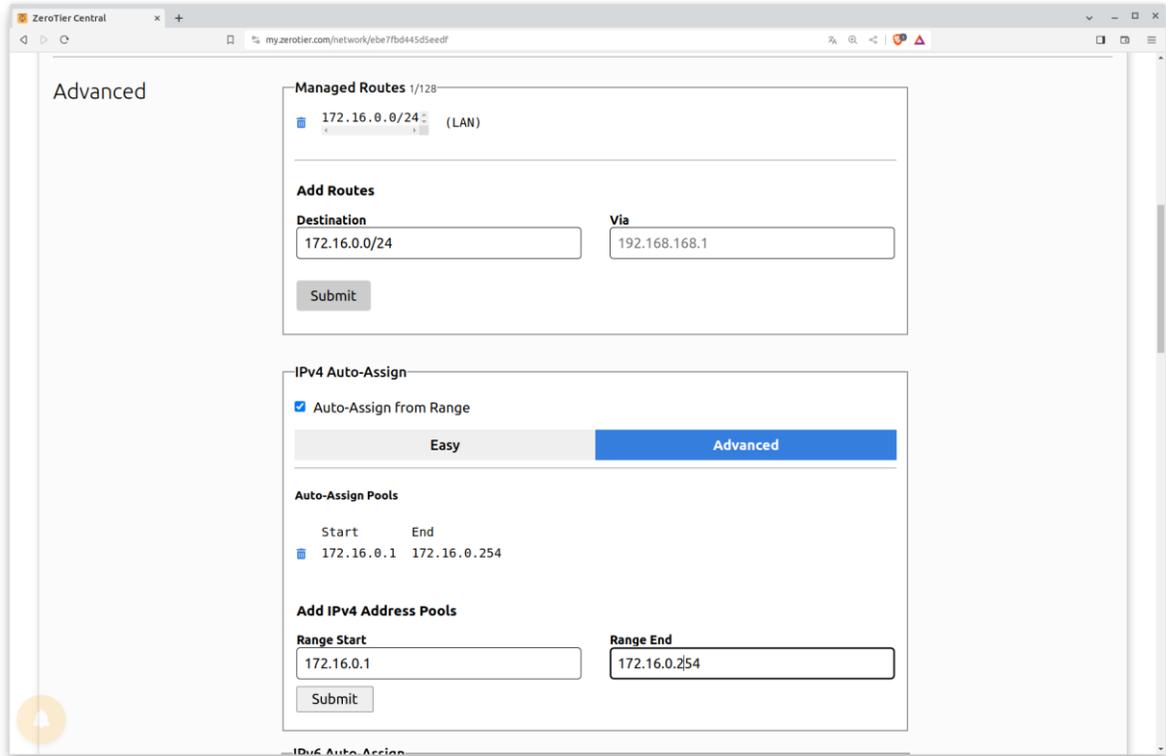


Ahora ya podemos modificar nuestra VPN. Es necesario tener en cuenta el ID de nuestra red (Network ID) este nos permitirá conectarnos a esa red VPN en específico.

Empezamos configurando lo básico, nombre de la red, descripción y colocar que será privado:



Luego empezamos con configuraciones avanzadas. Agregamos nuestra red 172.16.0.0/24 y el rango de IP's asignables.



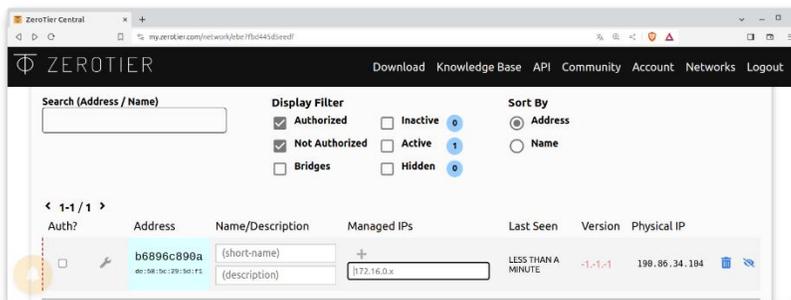
Con esta configuración podemos empezar a asignar ip a nuestras maquinas.

Para la asignación de ip a nuestra red VPN debes ejecutar los siguientes comandos en un nodo.

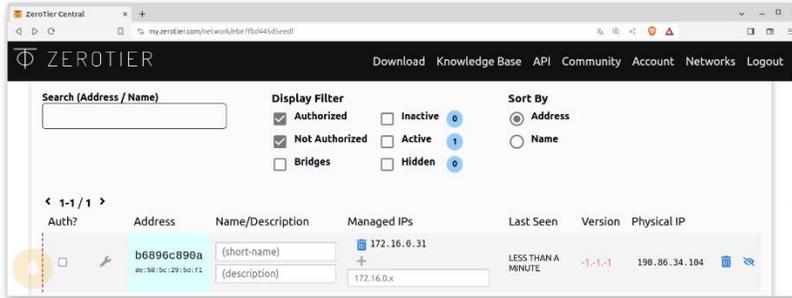
```
sudo apt install curl
curl -s https://install.zerotier.com | sudo bash
sudo zerotier-cli join Network_ID
sudo zerotier-cli status
```

El valor de Network_ID es el valor que se encuentra en la página del servidor VPN.

Ya con eso tendremos una interfaz nueva:



En el sitio de ZeroTier asignamos la ip de manera estática y tendremos esto:



Se nos mostrara la nueva interfaz:

```

1/1 | Tlxi: root@maxscale: /home/open
2: root@maxscale: /home/open
root@maxscale: /home/open# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 52:54:00:14:aa:f5 brd ff:ff:ff:ff:ff:ff
   inet 192.168.100.33/24 brd 192.168.100.255 scope global enp1s0
       valid_lft forever preferred_lft forever
   inet6 fe80::5054:ff:fe14:aaf5/64 scope link
       valid_lft forever preferred_lft forever
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 52:54:00:0d:1d:0e brd ff:ff:ff:ff:ff:ff
   inet 10.0.0.33/24 brd 10.0.0.255 scope global enp7s0
       valid_lft forever preferred_lft forever
   inet6 fe80::5054:ff:fe0d:1d0e/64 scope link
       valid_lft forever preferred_lft forever
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 52:54:00:0f:7d:34 brd ff:ff:ff:ff:ff:ff
   inet 10.0.2.33/24 brd 10.0.2.255 scope global enp8s0
       valid_lft forever preferred_lft forever
   inet6 fe80::5054:ff:fe0f:7d34/64 scope link
       valid_lft forever preferred_lft forever
5: ...
6: zth0rc40u2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 2800 qdisc fq_codel state UNKNOWN group default qlen 1000
   link/ether de:58:5c:29:5d:f1 brd ff:ff:ff:ff:ff:ff
   inet6 fe80::dc58:5c29:5df1/64 scope link
       valid_lft forever preferred_lft forever

```

Agregamos de igual manera a los nodos correspondientes (Keystone, Memcached, RabbitMQ y MaxScale)

Listado de Nodos a Configurar – Regiones

Para esta instalación nos basaremos primero a instalar Openstack en una sola region, y ocuparemos primero la guía oficial de Openstack, luego de ello se hacen las configuraciones correspondientes para conectarnos y/o consumir los servicios de nuestra infraestructura.

<https://docs.openstack.org/install-guide/>

Así como la guía tendremos 2 nodos un Controller y un Compute.

Creación de cola en RabbitMQ

Sera necesario crear una cola para cada una de las regiones, se tomará el siguiente ejemplo como base para una region:

```
$ sudo rabbitmqctl add_user NAME_REGION PASS
$ sudo rabbitmqctl add_vhost /VHOST_REGION
$ sudo rabbitmqctl set_permissions -p /VHOST_REGION
NAME_REGION ".*" ".*" ".*"
```

En nuestro caso NAME_REGION = regionone y VHOST_REGION = regionone.

NOTA: El password que normalmente ocupamos es icc115.

Integración Servicios OpenStack RegionOne

Glance

Ingresamos a mysql y agregamos una base de datos para Glance de RegionOne llamado glance_one.

```
# mysql
MariaDB [(none)]> CREATE DATABASE glance_one;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance_one.* TO
'glance_one'@'localhost' IDENTIFIED BY 'GLANCE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance_one.* TO
'glance_one'@'%' IDENTIFIED BY 'GLANCE_DBPASS';
```

Cambiar GLANCE_DBPASS por icc115.

En Keystone agregamos los servicios para Glance RegionOne.

```

$ . admin-openrc
$ openstack user create --domain default --password-prompt
glance
$ openstack role add --project service --user glance admin
$ openstack service create --name glance --description
"OpenStack Image" image
$ openstack endpoint create --region RegionOne image public
http://172.16.0.101:9292
$ openstack endpoint create --region RegionOne image internal
http://172.16.0.101:9292
$ openstack endpoint create --region RegionOne image admin
http://172.16.0.101:9292

```

Abrimos el archivo de configuración de Glance `/etc/glance/glance-api.conf` en la

RegionOne y reemplazamos con lo siguiente:

```

[database]
# ...
connection =
mysql+pymysql://glance_one:iccl15@172.16.0.31:4008/glance_one

[keystone_authtoken]
# ...
www_authenticate_uri = http://172.16.0.11:5000
auth_url = http://172.16.0.11:5000
memcached_servers = 172.16.0.21:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = iccl15

[oslo_limit]
auth_url = http://172.16.0.11:5000
auth_type = password
user_domain_id = default
username = glance
system_scope = all
password = iccl15
endpoint_id = ENDPOINT_PUBLIC
region_name = RegionOne

```

El `ENDPOINT_PUBLIC` es el que se registró en el nodo de Keystone. Si un dado caso

perdemos ese dato, podemos ejecutar el siguiente comando para recuperarlo:

```

$ openstack endpoint list --service image

```

Nos mostrara una lista de endpoint para el servicio de Glance, luego solo lo copiamos, en este caso nuestro id publico es: cb267e7fc5274596b8d71cb8ace40bf4

Luego ejecutamos los siguientes comandos en el nodo de Keystone:

```
$ openstack role add --user glance --user-domain Default --
system all reader
# su -s /bin/sh -c "glance-manage db_sync" glance
```

Placement

Agregamos lo siguiente a la base de datos.

```
MariaDB [(none)]> CREATE DATABASE placement_one;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement_one.* TO
'placement_one'@'localhost' IDENTIFIED BY 'iccl15';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement_one.* TO
'placement_one'@'%' IDENTIFIED BY 'iccl15';
```

Agregamos los endpoint al nodo Keystone

```
$ . admin-openrc
$ openstack user create --domain default --password-prompt
placement
$ openstack role add --project service --user placement admin
$ openstack service create --name placement --description
"Placement API" placement
$ openstack endpoint create --region RegionOne placement
public http://172.16.0.101:8778
$ openstack endpoint create --region RegionOne placement
internal http://172.16.0.101:8778
$ openstack endpoint create --region RegionOne placement
admin http://172.16.0.101:8778
```

Configuramos el archivo en la RegionOne /etc/placement/placement.conf

```
[placement_database]
connection =
mysql+pymysql://placement_one:iccl15@172.16.0.31:4008/placeme
nt_one

[keystone_authtoken]
# ...
auth_url = http://172.16.0.11:5000/v3
memcached_servers = 172.16.0.21:11211
auth_type = password
project_domain name = Default
```

```
user_domain_name = Default
project_name = service
username = placement
password = icc115
```

Ejecutamos lo siguiente para poblar la base de datos de placement:

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

Reiniciamos el servicio de placement

```
# service apache2 restart
```

Nova

Agregamos lo siguiente en la base de datos.

```
MariaDB [(none)]> CREATE DATABASE novaone_api;
MariaDB [(none)]> CREATE DATABASE novaone;
MariaDB [(none)]> CREATE DATABASE novaone_cell0;

MariaDB [(none)]> GRANT ALL PRIVILEGES ON novaone_api.* TO
'novaone'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON novaone_api.* TO
'novaone'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON novaone.* TO
'novaone'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON novaone.* TO
'novaone'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON novaone_cell0.* TO
'novaone'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON novaone_cell0.* TO
'novaone'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
```

Sustituimos NOVA_DBPASS por icc115.

Agregamos el usuario, el servicio y los endpoints de nova en el nodo de Keystone.

```
$ . admin-openrc
```

```
$ openstack user create --domain default --password-prompt
nova
$ openstack role add --project service --user nova admin
$ openstack service create --name nova --description
"OpenStack Compute" compute
$ openstack endpoint create --region RegionOne compute public
http://172.16.0.101:8774/v2.1
$ openstack endpoint create --region RegionOne compute
internal http://172.16.0.101:8774/v2.1
$ openstack endpoint create --region RegionOne compute admin
http://172.16.0.101:8774/v2.1
```

Modificamos el siguiente archivo del Controller de RegionOne /etc/nova/nova.conf

```
[DEFAULT]
# ...
transport_url =
rabbit://regionone:iccl15@172.16.0.22:5672//regionone
my_ip = 172.16.0.101

[api_database]
# ...
connection =
mysql+pymysql://novaone:iccl15@172.16.0.31:4008/novaone_api

[database]
# ...
connection =
mysql+pymysql://novaone:iccl15@172.16.0.31:4008/novaone

[keystone_authtoken]
# ...
www_authenticate_uri = http://172.16.0.11:5000/
auth_url = http://172.16.0.11:5000/
memcached_servers = 172.16.0.21:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = iccl15

[service_user]
send_service_user_token = true
auth_url = https://172.16.0.11:5000/v3
auth_strategy = keystone
auth_type = password
project_domain_name = Default
```

```

project_name = service
user_domain_name = Default
username = nova
password = icc115

[glance]
# ...
api_servers = http://172.16.0.101:9292

[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://172.16:5000/v3
username = placement
password = icc115

```

Poblamos las bases de datos:

```

# 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
# su -s /bin/sh -c "nova-manage cell_v2 list_cells" nova

```

Reiniciamos los servicios de Nova

```

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

```

Modificamos el siguiente archivo en el nodo Compute de RegionOne /etc/nova/nova.conf

```

[DEFAULT]
# ...
transport_url =
rabbit://regionone:icc115@172.16.0.22:5672//regionone
my_ip = 172.16.0.102

[keystone_authtoken]
# ...
www_authenticate_uri = http://172.16.0.11:5000/
auth_url = http://172.16.0.11:5000/
memcached_servers = 172.16.0.21:11211

```

```

auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = iccl15

[service_user]
send_service_user_token = true
auth_url = https://172.16.0.11:5000/v3
auth_strategy = keystone
auth_type = password
project_domain_name = Default
project_name = service
user_domain_name = Default
username = nova
password = iccl15

[vnc]
novncproxy_base_url = http://172.16.0.101:6080/vnc_auto.html

[glance]
# ...
api_servers = http://172.16.0.101:9292

[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://172.16.0.11:5000/v3
username = placement
password = iccl15

```

Reiniciamos el servicio de Nova:

```
# service nova-compute restart
```

En el nodo Controller de RegionOne confirmamos que se encuentren los hosts compute:

```
$ . admin-openrc
$ openstack compute service list --service nova-compute
```

Poplamos la base de datos:

```
# su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --
verbose" nova
```

Neutron

Agregamos la siguiente base de datos y permisos:

```
MariaDB [(none)]> CREATE DATABASE neutron_one;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron_one.* TO
'neutron_one'@'localhost' IDENTIFIED BY 'iccl15';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron_one.* TO
'neutron_one'@'%' IDENTIFIED BY 'iccl15';
```

En el nodo Keystone de nuestra infraestructura agregamos el usuario, el servicio y los endpoint para neutron.

```
$ . admin-openrc
$ openstack user create --domain default --password-prompt
neutron
$ openstack role add --project service --user neutron admin
$ openstack service create --name neutron --description
"OpenStack Networking" network
$ openstack endpoint create --region RegionOne network public
http://172.16.0.101:9696
$ openstack endpoint create --region RegionOne network
internal http://172.16.0.101:9696
$ openstack endpoint create --region RegionOne network admin
http://172.16.0.101:9696
```

Editamos el archivo de configuración de neutron en el nodo Controller de nuestra

RegionOne /etc/neutron/neutron.conf:

```
[DEFAULT]
# ...
transport_url =
rabbit://regionone:iccl15@172.16.0.22:5672//regionone

[database]
# ...
connection =
mysql+pymysql://neutron_one:iccl15@172.16.0.31:4008/neutron_o
ne

[keystone_authtoken]
www_authenticate_uri = http://172.16.0.11:5000
auth_url = http://172.16.0.11:5000
```

```
memcached_servers = 172.16.0.21:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = icc115

[nova]
# ...
auth_url = http://172.16.0.11:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = nova
password = icc115
```

Modificamos el archivo openvswitch del Controller

`/etc/neutron/plugins/ml2/openvswitch_agent.ini`

```
[vxlan]
local_ip = 172.16.0.101
l2_population = true
```

Modificamos el archivo de configuración de neutron `/etc/nova/nova.conf`:

```
[neutron]
# ...
auth_url = http://172.16.0.11:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = icc115
service_metadata_proxy = true
```

Poplamos la base de datos de neutron

```
su -s /bin/sh -c "neutron-db-manage --config-file
/etc/neutron/neutron.conf --config-file
/etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
```

Reiniciamos servicios:

```
# service nova-api restart
```

```
# service neutron-server restart
# service neutron-openvswitch-agent restart
# service neutron-dhcp-agent restart
# service neutron-metadata-agent restart
# service neutron-l3-agent restart
```

En el nodo Compute de RegionOne editamos el archivo `/etc/neutron/neutron.conf`:

```
[DEFAULT]
# ...
transport_url = rabbit://regionone:iccl15@172.16.0.22:
5672//regionone

[keystone_authtoken]
# ...
www_authenticate_uri = http://172.16.0.11:5000
auth_url = http://172.16.0.11:5000
memcached_servers = 172.16.0.21:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = iccl15
```

Editamos el archivo de configuración de Nova `/etc/nova/nova.conf`:

```
[neutron]
# ...
auth_url = http://172.16.0.11:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = iccl15
```

Reiniciamos los servicios de nodo Compute:

```
# service nova-compute restart
# service neutron-openvswitch-agent restart
```

Instalación Horizon

Horizon se puede instalar en cualquier nodo de nuestra infraestructura y además también en los nodos de nuestras Regiones, por convicción lo instalaremos en el nodo Keystone de nuestra infraestructura a modo de ejem

Para instalarlo ejecutamos:

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

Editamos el siguiente archiv `tc/openstack-dashboard/local_settings.py`, contiene la configuracion de Horizon para conectarse a los servicios que hemos creado:

```
OPENSTACK_HOST = "172.16.0.11"
...
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
...
CACHES = {
    'default': {
        'BACKEND':
'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '10.0.0.21:11211',
    }
}
...
OPENSTACK_KEYSTONE_URL = "http://%s:5000/identity/v3" %
OPENSTACK_HOST

OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True

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

OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"

OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
```

Las credenciales en nuestro caso son usuario: admin, password: iccl15.

