



UNIVERSIDAD  
CATÓLICA DE  
TEMUCO

INGENIERÍA CIVIL  
EN INFORMÁTICA  
FACULTAD DE INGENIERÍA

# **IMPLEMENTACIÓN DE SISTEMA DE COMPUTACIÓN EN LA NUBE PARA LA EXPANSIÓN DE LA INFRAESTRUCTURA TECNOLÓGICA EN CONTEXTO ACADÉMICO**

Por

Michel Alexis Muñoz Castro

Trabajo de Título presentado a la  
Facultad de Ingeniería de la Universidad Católica de Temuco  
para optar al título de Ingeniero Civil en Informática

Temuco, Noviembre del 2021



UNIVERSIDAD  
CATÓLICA DE  
TEMUCO

INGENIERÍA CIVIL  
EN INFORMÁTICA  
FACULTAD DE INGENIERÍA

## COMISIÓN EXAMEN DE TÍTULO

**Este Examen de Título ha sido realizado en el Departamento de Ingeniería Informática:**

Presidente Comisión: .....

**Sr. Marcos Lévano Huamaccto**

Magíster en Ingeniería en Informática

Ingeniero Informático

Profesor Guía : .....

**Sr. Alejandro Mellado Gatica**

Ingeniero de Ejecución en Informática

Magister en telecomunicaciones

Master en Soc. de la Información y conocimiento

Profesor Informante: .....

**Sr. Luis Alberto Caro Saldivia**

Ingeniero Civil Informático

Jefe Carrera de Ingeniería Civil Informática: .....

**Sr. Marcos Lévano Huamaccto**

Magíster en Ingeniería en Informática

Ingeniero Informático

Temuco, Noviembre de 2021.



## INFORME TRABAJO DE TÍTULO

**TITULO:** “IMPLEMENTACIÓN DE SISTEMA DE COMPUTACIÓN EN LA NUBE PARA LA EXPANSIÓN DE LA INFRAESTRUCTURA TECNOLÓGICA EN CONTEXTO ACADÉMICO”.

**ALUMNO:** MICHEL ALEXIS MUÑOZ CASTRO

En mi calidad de Profesor Guía, mis apreciaciones del presente informe de Trabajo de Título, son las siguientes:

- El presente trabajo posee un alto nivel de configuración en área de infraestructura que es el resultado la integración de múltiples elementos de hardware y software.
- Se aprecia que el estudiante posee altas habilidades técnicas validadas a través de una implementación práctica, que presenta una solución de optimización de recursos frente a un problema dado.
- La propuesta cumple con los objetivos planteados y constituye una base documentada para la transformación de una plataforma que debe ser actualizada considerando además, las nuevas estructuras que permiten mayor flexibilidad y escalabilidad en la puesta en marcha y aprovisionamiento de servicios en nube.

De acuerdo a estas consideraciones califico el desarrollo de este Trabajo de Título con nota **7,0 (siete coma cero)**.

---

Alejandro Mauricio Mellado Gatica

Profesor Guía

Temuco, 03 Noviembre de 2021.



## INFORME TRABAJO DE TÍTULO

**TITULO:** **“IMPLEMENTACIÓN DE SISTEMA DE COMPUTACIÓN EN LA NUBE PARA LA EXPANSIÓN DE LA INFRAESTRUCTURA TECNOLÓGICA EN CONTEXTO ACADÉMICO”.**

**ALUMNO:** **MICHEL ALEXIS MUÑOZ CASTRO**

En mi calidad de Profesor Informante, mis apreciaciones del presente informe de Trabajo de Título, son las siguientes:

- El trabajo aborda la implementación computacional en la nube expandiendo las capacidades virtuales de los servidores del centro de datos de la carrera de ingeniería civil informática.
- El trabajo está bien articulado y su desarrollo es atractivo e interesante.
- La solución planteada es adecuada para este tipo de problemas.
- El dominio de los temas tratados en este trabajo habilitan al estudiante Michel Muñoz Castro a ejercer la profesión como Ingeniero Civil Informático

De acuerdo a estas consideraciones califico el desarrollo del Trabajo de Título con nota máxima **7,0 (siete coma cero)**.

---

**Luis Alberto Caro Saldivia**

**Profesor Informante**

*Dedicado a mi madre Sandra Castro Ailef y a mi abuela Sabina Ailef,  
también a cada una de las personas que hizo posible este trabajo a lo largo de los años.*

*Muchas gracias a todos.*

## Agradecimientos

Muchas son las personas a las que debo agradecer, empezando por mi familia, mi madre, padre, hermana, abuelos, primos, tíos, entre muchos más. Por apoyarme y motivarme a estudiar mi pasión. Además por alentarme a la distancia desde Osorno y por preocuparse de que tenga todo lo necesario para poder estudiar fuera de mi ciudad de origen.

Agradezco también a mis amigos de toda la vida, que también son parte de mi familia, por su amistad, comprensión, motivación, por estar presentes y poder crecer juntos. Ahora ya somos adultos, profesionales y seguimos unidos, divirtiéndonos como si fuéramos unos niños.

Agradezco también al departamento de la Carrera de Ingeniería Civil Informática, a los profesores, secretarias y asistentes de mi educación durante el transcurso de la carrera. Al profesor Alejandro Mellado, por ser el profesor guía del desarrollo del trabajo de título y por darme la oportunidad de ser su ayudante durante varios años. También he de agradecer en especial a Romina Neira y Carla Arias Quiroz, por ser el corazón del departamento de carrera, por su preocupación y apoyo en los momentos más difíciles.

También agradezco a mis compañeros de carrera que estuvimos colaborando y trabajando juntos para lograr este objetivo final, espero que les vaya bien a todos en sus carreras profesionales y caminos en la vida.

Finalmente agradezco a cada una de las personas que pudo hacer esto posible.

Aprendí mucho en todos los ámbitos en lo personal y profesional.

Sin ustedes no hubiese sido posible, muchas gracias.

## Índice

<b>Índice general</b>	<b>VII</b>
<b>Índice figuras</b>	<b>VIII</b>
<b>Índice de cuadros</b>	<b>IX</b>
<b>Resumen (Abstract)</b>	<b>XI</b>
<b>I. Introducción</b>	<b>12</b>
1.1. Contexto y antecedentes	12
1.2. Problema o necesidad	13
1.3. Solución	14
1.4. Objetivos	15
<b>II. Metodología</b>	<b>15</b>
<b>III. Marco teórico</b>	<b>17</b>
3.1. La Computación en la nube	17
3.2. Virtualización	21
3.3. OpenStack	23
<b>IV. Desarrollo</b>	<b>29</b>
<b>V. Resultados y discusión</b>	<b>32</b>

<b>VI. Conclusiones</b>	<b>39</b>
<b>VII. Bibliografía</b>	<b>40</b>
<b>VIII. Anexos</b>	<b>41</b>
<b>IX. Anexo del Desarrollo</b>	<b>44</b>

## Índice de figuras

1. Modelos de servicio de computación en la nube	17
2. Tipos de virtualización	20
3. Arquitectura conceptual de OpenStack	24
4. Funcionamiento de redes de proveedor	31
5. Funcionamiento de redes de autoservicio	32
6. Verificación funcionamiento de la red de autoservicio	33
7. Verificación funcionamiento de la red de autoservicio	33
8. Verificación funcionamiento de los routers virtuales	33
9. Verificación funcionamiento de los pares de claves	35
10. Verificación funcionamiento de los servidores virtuales	37
11. Verificación funcionamiento de VNC	37

12. Verificación funcionamiento de VNC	37
13. Verificación funcionamiento de red máquina virtual	38
14. Verificación funcionamiento y acceso a servidor virtual ubuntu	39
15. Requerimientos básicos de OpenStack	45
16. Opción de redes de autoservicio	48
17. Esquema de red de ambiente	49
18. Esquema de red propuesta para la implementación	50
19. Captura de configuración de red mediante Yast	51
20. Captura de verificación de conectividad. Fuente propia.	51
21. Captura de verificación de funcionamiento de NTP en nodo controlador	52
22. Captura de verificación del funcionamiento de NTP en nodo de cómputo	52
23. Captura de pantalla de verificación del funcionamiento de Keystone	62
24. Captura de pantalla verificación del funcionamiento de Keystone	63
25. Captura de pantalla comprobación del funcionamiento de Glance	67
26. Captura de pantalla de la verificación del funcionamiento del servicio Placement	70
27. Captura de pantalla de comprobación del módulo del kernel br_netfilter	75
28. Captura de pantalla verificación del funcionamiento de Neutro	78

29. Captura de pantalla funcionamiento Nova	86
30. Captura de pantalla funcionamiento Nova	87
31. Captura de pantalla funcionamiento Nova	87
32. Captura de pantalla funcionamiento Nova	88
33. Captura de pantalla funcionamiento Nova	88
34. Captura de pantalla verificación funcionamiento de Horizon	91

## Índice de tablas

1. Especificaciones técnicas Nodo Controlador	46
2. Especificaciones técnicas Nodo Controlador	47
3. Configuración de red de nodos	50

## Resumen

Este trabajo de título trata acerca de la posibilidad de implementar la Computación en la Nube en un contexto académico. Esto para expandir las capacidades de servidores virtuales dentro del centro de datos de la carrera Ingeniería Civil Informática de la Universidad Católica de Temuco. El objetivo es aprovechar y estudiar los beneficios que aporta la Computación en la nube y la Infraestructura como Servicio. De forma exploratoria se revisan los conceptos y la documentación necesaria para lograr una implementación funcional del sistema OpenStack. Una vez ya implementado se realiza una prueba de funcionamiento para validar el despliegue. Finalmente se analizan los beneficios y se concluye acerca del trabajo realizado.

## Abstract

This degree work is about the possibility of implementing Cloud Computing in an academic context. In order to expand the capabilities of virtual servers within the data center from Ingeniería Civil Informática at Universidad Católica de Temuco. The objective is to take advantage of implementation and study the benefits of Cloud Computing and Infrastructure as a Service. In an exploratory way, the concepts and documentation necessary to achieve a functional implementation of the OpenStack system are reviewed. Once implemented, a functional test is realized to validate the deployment. Finally, the benefits are analyzed and a conclusion is made about the work.

## I. Introducción

### 1.1. Contexto y antecedentes

Actualmente el departamento de Ingeniería Civil Informática dispone de una infraestructura de servidores también llamado centro de datos (ó datacenter en inglés) para diversas funciones relacionadas con la carrera de estudios. Son alrededor de veinte los servidores que están alojados físicamente en las dependencias del San Juan Pablo II de la universidad (Ver anexos 1). Alumnos, tesistas y profesores se benefician de este sistema para desplegar diferentes servicios, páginas web, aplicaciones, almacenar datos, entre otras funciones.

Dentro de esta infraestructura se destacan las máquinas anfitrionas que alojan servidores virtuales, estas definición son:

“Una capa de abstracción o un ambiente entre los componentes de hardware y el usuario final. Las máquinas virtuales ejecutan sistemas operativos y a veces se les denomina servidores virtuales. Un sistema operativo anfitrión puede ejecutar muchas máquinas virtuales y compartir componentes de hardware como los CPUs, controladores, discos, memoria y entradas/salidas entre servidores virtuales” [1](Daniels, 2012).

Estos servidores virtuales desde su implementación han crecido en gran número, habiendo alrededor de treinta de ellos, ampliando la capacidad del centro de datos y la cantidad de sistemas que puede albergar.

La gran cantidad de servidores virtuales ha vuelto su mantenimiento y administración una tarea difícil. Desde la creación de un servidor virtual hasta su uso los administradores del centro de datos deben hacerse cargo de este proceso de forma manual y rudimentaria. Mientras que el usuario final de estos servidores virtuales son dependientes del proceso y por ello deben esperar a los administradores, lo cual causa retrasos en sus labores y les dificulta la accesibilidad a estos sistemas.

Para solucionar esta situación de una forma automatizada se pretende implementar el paradigma de **Computación en la Nube** que es “un nuevo concepto donde los recursos (tecnológicos) son virtualizados, dinámicamente extendidos y proveídos como un servicio a través de internet.” [2](Sefraoui et al, 2012).

Uno de sus métodos de servicio es la **Infraestructura como Servicio** el cual “es la entrega de hardware (servidor, almacenamiento, redes), y el software asociado (tecnología de sistemas operativos virtualizados, sistema de archivos), como un servicio.” [3] (Bhardwaj et al, 2010).

Para implementar una Infraestructura como Servicio se requiere implementar diversos sistemas en diferentes áreas tecnológicas, por lo cual es necesario un software que cubra todas las necesidades de virtualización, almacenamientos, redes, bases de datos, clientes gráficos, entre otros. Para implementar estos sistemas ha escogido la plataforma de software libre **OpenStack** que existe para el propósito de implementar servicios privados o locales en nube, que es usado con mucho éxito en el mercado y en

organizaciones gubernamentales como la NASA. Como lo describe [2](Sefraoui et al 2012) “Openstack es una solución reciente bajo desarrollo activo. Tiene un gran potencial dado a su arquitectura y comunidad y los partners que lo apoyan. (...). Este proyecto se dedica a proveer a la industria computacional la oportunidad de construir sus propias arquitecturas anfitrionas con escalabilidad masiva y es completamente software libre”.

Lo que se busca con la implementación de este robusto sistema es la automatización del uso de servidores virtuales, dotar de independencia al usuario, simplificar el proceso de creación de máquinas virtuales y optimizar la utilización de los recursos de hardware. De paso también aprovechar todos los beneficios que provee la computación en la nube en un contexto local.

En este documento se revisa en detalle la implementación del sistema y las posibilidades que aporta un sistema de computación en la nube a la infraestructura computacional ya existente en el centro de datos de la carrera Ingeniería Civil Informática en la Universidad Católica de Temuco.

## 1.2. Problema o necesidad

La infraestructura actual de los servidores virtuales dentro del centro de datos posee principalmente dos problemas o necesidades que deben ser trabajados.

En primer lugar, la gran cantidad de servidores virtuales y su difícil administración: Estos son alrededor de treinta, los cuales cada uno debe ser desplegado por el administrador del centro de datos de forma manual. En segundo lugar, vuelve desde un principio a los usuarios dependientes de la administración. Lo que finalmente puede causar retrasos en sus labores.

Para entender cómo se dan estos problemas hay que explicar el modo de trabajo actual que existe con los servidores virtuales.

Dentro del datacenter existen los llamados servidores anfitriones, estos utilizan “Las capas de abstracción llamadas hipervisores (...), estos realizan llamadas desde la máquina virtual a la máquina real. Los hipervisores actuales usan los componentes de hardware de la máquina real, pero permiten (al mismo tiempo) diferentes sistemas operativos de máquinas virtuales y configuraciones. (...). Estos tienen funciones especializadas de administración que permiten a múltiples máquinas virtuales coexistir pacíficamente mientras comparten recursos de la máquina real.” [1](Daniels, 2012).

El hipervisor específico que se ocupa en el departamento es **Xen**: “La comunidad del proyecto Xen desarrolla un software libre hipervisor tipo-1, el cual hace posible correr muchas instancias de un sistema operativo o de hecho ejecutar diferentes sistemas operativos en paralelo en una sola máquina anfitrión. (...). Este es usado como base para un número de diferentes aplicaciones comerciales y de software libre, como lo son: virtualización de servidores, Infraestructura como Servicio (IaaS), virtualización de escritorio, aplicaciones de seguridad, aparatos empotrados y hardware. El proyecto hipervisor Xen está impulsando las nubes más grandes que están en producción actualmente” [5](Xen Project, 2021).

Con este sistema de virtualización configurado en un servidor anfitrión se puede acceder a administrar servidores virtuales mediante el software **Virtual Machine Manager** el cual conecta con la tecnología Xen:

“La aplicación virt-manager es una interfaz de escritorio para manejar máquinas virtuales a través de libvirt (librería interfaz entre los diferentes tipos de hipervisores). Este principalmente apunta a máquinas virtuales KVM (Kernel Virtual Machines, otro hipervisor basado en el kernel Linux), pero también maneja Xen y LXC (Linux containers, virtualización por software). Este presenta una vista resumen de los dominios en ejecución, su rendimiento en tiempo real y estadísticas de uso de recursos. Las utilidades permiten crear nuevas máquinas virtuales (dominios), configurar y ajustar las ubicaciones de recursos de los dominios y el hardware virtual.” [6] (Daniel P. Berrangé, 2021).

A partir de este punto es que el administrador ejecuta los pasos necesarios dentro del virt-manager para crear un tipo de máquina virtual, escogiendo un archivo imagen del sistema operativo que se desea obtener. Luego se configura el sistema y la red donde se encuentra. Para que finalmente se le pueda dar acceso al usuario final.

Estos pasos son esenciales y son realizados cada vez que un usuario necesita una máquina virtual. Son pasos técnicos difíciles de enseñar a la comunidad de estudiantes, profesores y tesis. Incluyen tecnologías que no son usuales para todos los informáticos/as, lo que vuelve a todo el sistema de difícil acceso, en comparación a las soluciones del mercado de pago como lo son Amazon Web Services, Microsoft Azure o Google Cloud.

### 1.3. Solución

Por todos estos inconvenientes se vuelve necesario tener un sistema más **automatizado** para el administrador e **independiente** para el usuario. Donde este último pueda desde un principio decidir qué sistema utilizar y que luego obtenga el acceso **sin la necesidad de intervención del administrador o la dificultad técnica y el aprendizaje que requiere el método actual (Xen y Virtual Machine Manager)**.

Esto en otras palabras sería solicitar el servicio al sistema y éste entregue la máquina virtual lista para su utilización. Por lo cual el usuario podría empezar a trabajar rápidamente en sus labores propias sin la necesidad de configurar la máquina virtual o solicitar a la administración una.

Para poder llevar a cabo tales objetivos se propone la Infraestructura como Servicio y OpenStack como el sistema que solucione esas necesidades.

Openstack en su documentación oficial se define como “un sistema operativo de la nube que permite controlar grandes grupos de recursos informáticos, de almacenamiento y de redes en todo un centro de datos, todo administrado a través de un panel que brinda control a los administradores al tiempo que permite a sus usuarios aprovisionar recursos a través de una interfaz web.” [7] (OpenStack, 2021) .

Dentro de esta definición la última parte es lo que se busca solucionar en este trabajo. Que una vez ya se encuentre el sistema en funcionamiento, los usuarios puedan aprovisionar sus recursos a través de una simple interfaz web, sin la necesidad de instalar software adicional y realizar la configuración de Xen y virt-manager, además de proveer de servidores virtuales pre-configurados que no necesiten intervención de los administradores y estos se limiten al control del sistema general.

La simplificación y automatización del proceso de obtención de máquinas virtuales (MV) permitiría ampliar los posibles usuarios, disminuyendo el tiempo de despliegue de máquinas virtuales y volviéndolos independientes de la administración. Mientras que para el administrador se reduciría el tiempo de configuración que requiere desplegar las máquinas virtuales y la responsabilidad del funcionamiento de estas.

Por todas estas razones, en este documento se explora la posibilidad de utilizar OpenStack como solución y expansión de los sistemas ya existentes. Esto mediante la implementación del sistema y el despliegue de un servidor virtual que podría ser usado dentro de la infraestructura de servidores del departamento.

## 1.4. Objetivos

### Objetivo general:

Implementar un sistema de computación en la nube para expandir la infraestructura tecnológica de la carrera Ingeniería Civil Informática de la Universidad Católica de Temuco.

### Objetivos específicos:

1. Diseñar una arquitectura de sistemas para computación en la nube
2. Desplegar IaaS mediante OpenStack
3. Prueba de funcionamiento de la implementación

## II. Metodología

La metodología de este proyecto consistirá en tres pasos principales. El primero es acerca de los requisitos y la configuración del entorno. La segunda es acerca de la implementación del sistema de computación en la nube con modelo de Infraestructura como Servicio mediante OpenStack. El tercero es una prueba del funcionamiento de implementación como solución funcional y verificación del trabajo hecho.

### 2.1. Metodología de diseño de arquitectura de sistemas para computación en la nube

Para implementar este robusto sistema de computación en la nube es necesario tener en cuenta los requisitos de hardware y del entorno de red. Para ello se revisará el equipo de servidores y red existentes en el datacenter para revisar si cumplen con los requisitos mediante una tabla comparativa de ficha técnica de los equipos y los requerimientos oficiales de OpenStack. También se configurarán los equipos de red el entorno necesario de interconexiones que el sistema necesita.

La finalidad de este método es poder dejar la arquitectura de sistemas del centro de datos (servidores y redes) preparada para desplegar OpenStack sobre ella.

## **2.2. Metodología de despliegue de IaaS mediante OpenStack**

Mediante la documentación oficial de OpenStack y diversas fuentes en internet se realizará la configuración y despliegue del sistema en la nube.

Siguiendo la guía oficial y los recursos que existen en internet se configurará cada componente según estos estén ordenados en la guía oficial. Cada uno de estos componentes o sub-componentes poseen sus pasos a seguir y una verificación final, la cual debe ser pasada. Si no pasase la revisión se deberá revisar de nuevo la guía oficial. Si esto tampoco arregla el problema se pasará a la revisión de diversos recursos en internet, principalmente foros oficiales y no oficiales. Si finalmente funciona se puede continuar con el paso siguiente hasta terminar la configuración del sistema completa.

Esto incluye un gran número de componentes interconectados operando correctamente, por lo cual es una tarea ardua y extensa. Además la guía posee algunas falencias donde se vuelve super necesaria la comunidad que existe alrededor del software OpenStack en los foros de la comunidad como de comunidades cercanas al proyecto.

En esta parte se incluirá las configuraciones más elementales, se detalla cuales son los pasos a seguir y los resultados a obtener para que el servicio se vaya configurando y completando adecuadamente.

## **2.3. Metodología de prueba de funcionamiento de la implementación**

Se realizará una prueba del uso mediante la creación de una máquina virtual que use el sistema operativo Ubuntu 18.04 a modo de probar el correcto funcionamiento de la implementación. Esto servirá para demostrar que la configuración expuesta en este documento está operacional y lista para ser revisada. Esto da la posibilidad de aprender de la implementación ya realizada.

# **III. Marco teórico**

Para llevar a cabo el trabajo de implementación es necesario entender de antemano los conceptos bases de la Computación en la Nube, métodos de virtualización, Infraestructura como Servicio cómo el modelo de servicio y OpenStack como el sistema que lleva a cabo este modelo de trabajo. En este orden se podrá introducir al proyecto de forma inductiva, desde lo más esencial hasta el software que será utilizado en la implementación.

## **3.1. La Computación en la Nube**

La ‘computación en la nube’ es un nuevo concepto “**donde los recursos tecnológicos son virtualizados, dinámicamente extendidos y proveídos como un servicio a través de internet**” [2](Sefraoui et al, 2012). Es un concepto relativamente nuevo que junta todas las disciplinas, tecnologías y modelos de negocio para dotar de capacidad de TI como un servicio solicitado, escalable y elástico. Es una nueva tendencia donde la computación de los recursos de TI son dinámicamente escalables, virtualizados y expuestos como un servicio a través de internet.

“La computación en la nube se suele asociar con el suministro de nuevos mecanismos que permiten a los proveedores dar accesos a los usuarios a un número ilimitado de recursos virtuales (Resource Outsourcing). También usa sistema de pagos para usar esos recursos en base a su consumo, permitiendo modelos en demanda: pagar-por-usar. Existen garantías ofrecidas por el proveedor de la infraestructura a través de contratos a la medida. Actualmente las empresas más grandes participan ofreciendo soluciones en la nube, en especial Amazon EC2, Microsoft Azure, Google Apps e IBM blue cloud.” [2](Sefraoui et al, 2012).

Otra definición es que el concepto de ‘nube’ es “una forma metafórica de nombrar a Internet. Básicamente la computación en la nube consiste en los servicios ofrecidos a través de la red tales como correo electrónico, almacenamiento, uso de aplicaciones, etc. Los cuales son normalmente accedidos mediante un navegador web. Al utilizar estos servicios la información utilizada y almacenada, así como la mayoría de las aplicaciones requeridas, son procesadas y ejecutadas por un servidor en Internet. Dicho en otras palabras, se trata de una implementación que pretende transformar el arquetipo habitual de la computación y la informática, para trasladarla a Internet.” [8] (Mejía, 2011).

Según [3] (Bhardwaj et al, 2010) “La computación en la nube es una forma de referirnos al uso de recursos computacionales compartidos, y es la alternativa a tener servidores locales manejando aplicaciones. La computación en la nube agrupa en conjunto un gran número de servidores y otros recursos y usualmente ofrecen su capacidad combinada en la base de pago en-demanda o pagar-por-ciclo. Los usuarios finales de una red de computación en la nube no tienen idea donde están físicamente alojados sus servidores, ellos solo levantan sus aplicaciones y empiezan a trabajar. De acuerdo a Wang y von Laszewski, la computación en la nube puede ser definida como un set de servicios en red, proveídos de forma escalable, con garantías QoS (calidad de servicio, del inglés quality of service), normalmente personalizables, plataformas de computación baratas en demanda, los cuales pueden ser fácilmente adoptables”.

### **3.1.2. Características**

Escalabilidad: “Todo el sistema y su arquitectura es predecible y eficiente. Si un servidor maneja 1000 transacciones, 2000 transacciones serán manejadas por 2 servidores. Se establece un nivel de servicios que crea nuevas instancias de acuerdo a la demanda de operaciones existente de tal forma que reduzca el tiempo de espera y los cuellos de botella.” [8] (Mejía, 2011).

Virtualización: “Las aplicaciones son independientes del hardware en el que corran, incluso varias aplicaciones pueden correr en una misma máquina o una aplicación puede usar varias máquinas a la vez.

El usuario es libre de usar la plataforma que desee en su terminal (Windows, Unix, Mac, etc), al utilizar las aplicaciones existentes en la nube puede estar seguro de que su trabajo conservará sus características bajo otra plataforma.” [8] (Mejía, 2011).

Potencia computacional: “Conectando cientos de miles de computadores en conjunto en una nube crea una riqueza de poder computacional imposible de tener con una sola computadora de escritorio. La computación en la nube es accesible. Debido a que la información está almacenada en la nube, los usuarios pueden instantáneamente acceder a más información desde múltiples repositorios. No estás limitado a una sola fuente de información, como lo estás en la computadora de escritorio.” [9] (Mirashe y Kalyankar, 2010).

Seguridad: “El sistema está creado de tal forma que permite a diferentes clientes compartir la infraestructura sin preocuparse de ello y sin comprometer su seguridad y privacidad; de eso se ocupa el sistema de proveedor que se encarga de cifrar los datos.” [8] (Mejía, 2011). A lo cual personalmente añadiría la característica de que los proveedores usualmente suelen incluir implementaciones de seguridad, privacidad y de mantenerlas en funcionamiento.

Disponibilidad de la información: “No se hace necesario guardar los documentos editados por el usuario en su computadora o en medios físicos propios ya que la información radicará en Internet permitiendo su acceso desde cualquier dispositivo conectado a la red (con autorización requerida)” [8] (Mejía, 2011).

### **3.1.3. Modelos de servicio o arquitectura de capas**

Usualmente se divide a las implementación de nube en tres tipos según su modelo de servicio (de qué forma proveen el servicio al usuario) o a la arquitectura de capas según está conformada (siendo la capa del sistema la que define como el usuario usa el servicio). En la bibliografía están presentes estas dos definiciones para explicar el mismo concepto “La computación en la nube usualmente dividida en tres ofertas de niveles de servicio: Software como Servicio, Plataforma como Servicio e Infraestructura como servicio. Estos niveles soportan la virtualización y administración de diferente pila de soluciones.” [8] (Mejía, 2011).

En la siguiente figura se establece las tres capas según qué interacción tienen con el cliente que usa el servicio (izquierda de la pirámide), la capa correspondiente, a la derecha se añaden flechas indicando la posibilidad de sostener capas superiores y una flecha indicando la visibilidad de usuarios finales.

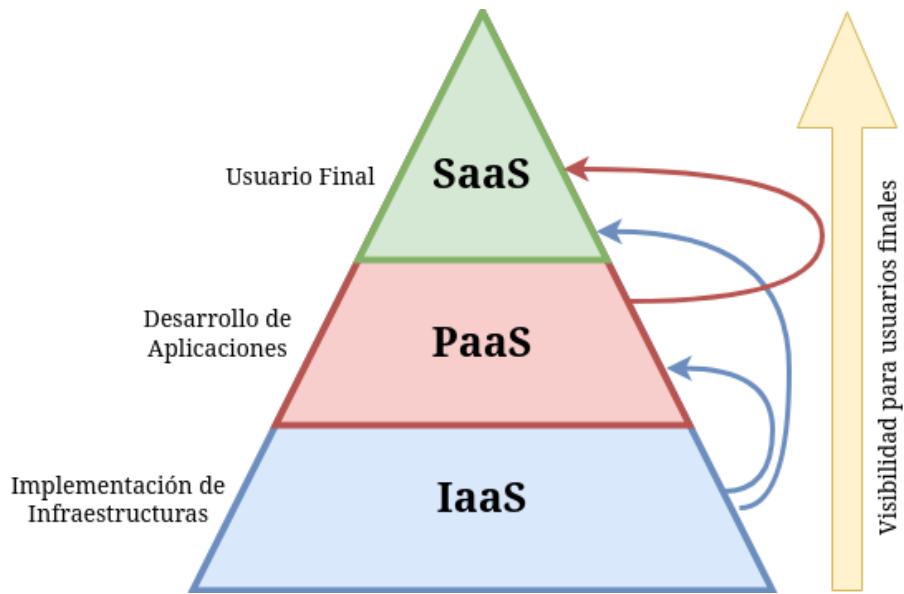


Figura 1 - Modelos de servicio de computación en la nube. Por Mejía (2011), edición propia.

### 3.1.4. Software como Servicio (SaaS)

El Software como servicio es la arquitectura de nube más usual y reconocible por sus usuarios, es la capa más alta y consiste en la entrega de aplicaciones completas como servicio.

“El proveedor de tecnologías de la información y comunicación (TIC) ofrece el SaaS (Software as a Service). Para ello dispone de una aplicación que se encarga de operar y mantener y que frecuentemente es desarrollada por él mismo. Con ella se encarga de dar servicio a multitud de clientes a través de la red, sin que éstos tengan que instalar ningún software adicional. La distribución de las aplicación tiene el modelo de uno a muchos, es decir, se elabora un producto y el mismo lo usan varios clientes.”

Los proveedores de SaaS son responsables de la disponibilidad y funcionalidad de sus servicios no dejando de lado las necesidades de los clientes que finalmente son los que usarán el software.

Las actividades son gestionadas desde alguna ubicación central, en lugar de hacerlo desde la sede de cada cliente, permitiendo a los clientes el acceso remoto a las aplicaciones mediante la web. Igualmente las actualizaciones son centralizadas, eliminando la necesidad de descargar parches por parte de los usuarios finales.

Un ejemplo claro es la aplicación para manejo de correo electrónico por medio de un navegador de internet.” [8] (Mejía, 2011).

A estos ejemplos se le puede sumar el caso de Google Docs, la plataforma de ofimática en línea de Google, se caracteriza por ser sencilla y poder usarse sin instalar aplicaciones en el computador del

usuario, además añade funciones de conexión y edición en tiempo real, lo cual amplía las posibilidades ofrecidas por herramientas de ofimática tradicional de escritorio.

### **3.1.5. Plataforma como Servicio (PaaS)**

“Esta es la idea de que alguien puede proveer el hardware más una cantidad de aplicaciones de software; como lo son las integraciones dentro de un set común de funciones de programación o bases de datos como base donde se podrá construir una aplicación. Las PaaS son una aplicación de desarrollo y plataforma de despliegue entregado como un servicio a los desarrolladores de aplicaciones sin el costo y la complejidad de comprar y administrador una infraestructura inferior, proveyendo de todas las facilidades requeridas para soportar el ciclo de vida completo de construcción y la entrega de aplicaciones web y servicios disponible desde Internet. Esta plataforma consiste en un software de infraestructura y típicamente incluye una base de datos, un middleware y herramientas de desarrollo. La arquitectura de virtualización y la clusterización en malla suelen ser las bases para esta infraestructura de software. Algunos PaaS ofrecen lenguajes de programación específicos o APIs. Por ejemplo Google App Engine es un PaaS que ofrece a los desarrolladores desarrollar en Python o Java. Engine Yard con Ruby on Rails. A veces los Proveedores PaaS tienen lenguajes propietarios como force.com de Salesforce.com y Coghead, ahora propiedad de SAP.” [3] (Bhardwaj et al, 2010).

En resumen PaaS permite “desarrollar tus propias aplicaciones usando los servicios proveídos. El cliente mantiene sólo aquellas aplicaciones mientras el proveedor mantiene los tiempos de ejecución de la nube, las interacciones SOA, bases de datos, software de servidor, virtualización, hardware del servidor y las redes de almacenamiento. Entre los principales proveedores: Google App Engine, Windows Azure.” [2] (Sefraoui et al, 2012).

### **3.1.6. Infraestructura como Servicio (IaaS)**

Infraestructura como Servicio (Infrastructure as a Service) corresponde al nivel más bajo de los modelos de entrega. La IaaS es “la entrega de hardware (servidores, almacenamiento y redes), y el software asociado (sistemas operativos, tecnologías de virtualización, sistemas de archivo) como un servicio. Es una evolución del tradicional hosting que no requiere ningún compromiso a largo plazo y permite a los usuarios aprovisionar recursos en demanda. A diferencia de las PaaS, los proveedores de IaaS hacen bastante poca administración más que proveer el centro de datos operacional y los usuarios deben desplegar y administrar los servicios de software por sí mismos al igual de la manera en que lo harían en sus propios centro de datos. Amazon Web Series Elastic Compute Cloud (EC2) y Secure Storage Service (S3) son ejemplos de oferta de IaaS.” [2] (Sefraoui et al, 2012). Cabe destacar que la IaaS al ser administrada desde máquinas virtuales, estas poseen las libertades de tener máquinas virtuales privadas, mientras que el soporte de la infraestructura de la IaaS debe ser mantenida por el proveedor. Si se requiere de aún más independencia por debajo de este nivel existen alternativas que escapan a la computación en la nube como lo son el arriendo de servidores, la colocación de servidores o de por si montar una infraestructura propia. Para cada uno de los casos de trabajo se debe decidir de antemano cual

acomoda más a las necesidades de la implementación de TI, siendo necesario en la industria de la tecnología contar con una o muchas de las posibilidades de desarrollo de arquitectura de sistemas donde se combina la nube con los centro de datos locales de una organización o empresa. Existen múltiples posibilidades para satisfacer las necesidades de infraestructura tecnológica que posean la organización o empresa.

“Por hacer una distinción respecto a las plataformas como servicio, las IaaS se presentan como una propuesta con mucho más flexibilidad para el uso que el usuario le tenga en mente, pero también requiere mucho más del cliente en lo que instalación, configuración y mantenimiento del software se refiere. Para proyectos que no se adapten en ninguna PaaS o en los que se quiera contar con libertad al momento de hacerlos evolucionar, existe la opción (y es preferible) de una Infraestructura como Servicio.

La IaaS permite desplazar al proveedor la mayor parte de los factores relacionados con la gestión de las máquinas con el ahorro de costos al pagar sólo por lo consumido y olvidarse de tratar con máquinas y su mantenimiento. Por otro lado, IaaS puede permitir una escalabilidad automática o semiautomática de forma que se puedan contratar más recursos según se requieran.” (Mejía, 2011).

La infraestructura como servicio es una forma de hosting. “Este incluye acceso a redes, enrutamientos de servicios y almacenamiento. El proveedor de la IaaS generalmente provee el hardware y los niveles administrativos necesarios para almacenar aplicaciones y una plataforma para correr aplicaciones. Escalar el ancho de banda, memoria y el almacenamiento son generalmente incluidos y los vendedores compiten en el rendimiento y precio ofrecidos en sus servicios dinámicos. El proveedor de servicios posee el equipamiento y es responsable por el hosting, ejecución y su mantención. Las IaaS pueden ser compradas por un contrato en pagar-por-uso. De todas formas, la mayor parte de los compradores consideran que el beneficio clave de las IaaS es la flexibilidad del precio, desde que solo deberías pagar por los recursos que tu aplicación requiere. Algunas de las características y componentes de la IaaS incluyen: Utilidad del poder computacional y el modelo de pago, automatización de tareas administrativas, escalamiento dinámico, virtualización de escritorio, servicios basados en políticas, conectividad a internet.

### **3.2. Virtualización**

La virtualización (también llamada tecnología de máquinas virtuales) es la base del funcionamiento de sistemas de computación en la nube.

“Las máquinas virtuales son una capa de abstracción o un ambiente entre los componentes de hardware y el usuario final. Las máquinas virtuales ejecutan sistemas operativos y a veces se les denomina servidores virtuales. Un sistema operativo anfitrión puede ejecutar muchas máquinas virtuales y compartir componentes de hardware como los CPUs, controladores, discos, memoria y entradas/salidas entre servidores virtuales.

Una ‘máquina real’ es el sistema operativo anfitrión y sus componentes de hardware, a veces descritos como ‘bare metal’ (metal básico, traducido literalmente), como la memoria, CPU, placa madre e interfaz de red.

La máquina real es en esencia un sistema operativo anfitrión sin máquinas virtuales. El sistema operativo de la máquina real accede a los componentes de hardware realizando llamadas a través de un programa de bajo nivel.

Las capas de abstracción llamadas hipervisores (ó monitores de máquinas virtuales), realizan llamadas desde la máquina virtual a la máquina real. Los hipervisores actuales usan los componentes de hardware de la máquina real, pero permiten (al mismo tiempo) diferentes sistemas operativos de máquinas virtuales y configuraciones. Por ejemplo, un sistema operativo anfitrión podría correr en SuSE Linux y podría ejecutar máquinas virtuales en Windows 2003 y Solaris 10. Estos tienen funciones especializadas de administración que permiten a múltiples máquinas virtuales coexistir pacíficamente mientras comparten recursos de la máquina real.” [1] (Daniels, 2012)

### 3.2.1. Tipos de virtualización

Dentro de la tecnología de la virtualización se encuentran implementadas en varias formas, pero se destacan tres tipos:

1. Máquinas virtuales por software (ver figura 3), el cual maneja la interacción entre el sistema operativo anfitrión y el sistema operativo huésped (Por ejemplo, Microsoft Virtual Server 2005).
2. Máquinas virtuales por hardware (ver figura 3), en cual la tecnología de virtualización se sitúa directamente en el sistema anfitrión (bare metal) usando un hipervisor, código modificado o APIs para facilitar la rápida transacción a los dispositivos de hardware (Por ejemplo. VMWare ESX).
3. Máquinas virtuales de sistema operativo/contenedores en el cual el sistema operativo anfitrión es particionado en zonas de contenedores (Por ejemplo Solaris Zones, BSD Jail)”.

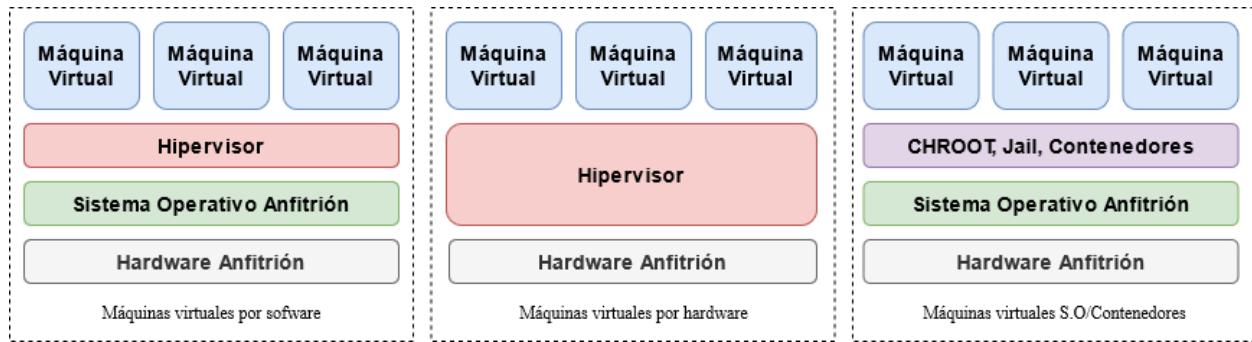


Figura 2 - Tipos de virtualización por Daniels (2012), edición propia.

En este trabajo se destaca el uso del segundo . El cuál dará la posibilidad de tener un gran número de máquinas virtuales en crecimiento con un rendimiento óptimo y una eficaz utilización de los recursos de hardware disponible, esto mediante el uso de la tecnología QEMU/KVM. En el caso particular de la implementación anterior (Xen y Virtual Machine Manager) revisaremos Xen que es una tecnología de hipervisor.

### **3.2.2. Xen**

Xen es el hipervisor que se encuentra en uso en la implementación anterior de virtualización en el departamento de carrera.

“La comunidad del proyecto Xen desarrolla un software libre hipervisor tipo-1, el cual hace posible correr muchas instancias de un sistema operativo o de hecho ejecutar diferentes sistemas operativos en paralelo en una sola máquina anfitrión. El proyecto desarrolla el único hipervisor de tipo-1 y que está disponible como código libre. Este es usado como base para un número de diferentes aplicaciones comerciales y de software libre, como lo son: virtualización de servidores, infraestructura como servicio (IaaS), virtualización de escritorio, aplicaciones de seguridad, aparatos empotrados y hardware. Esto permite al usuario incrementar la utilización de los servidores, consolidado granjas de servidores, reduciendo la complejidad y decrementando el costo total de la propiedad.” [5] (Xen Project, 2021).

### **3.2.2. Virtual Machine Manager**

“La aplicación virt-manager es una interfaz de escritorio para manejar máquinas virtuales a través de libvirt (librería interfaz entre los diferentes tipos de hipervisores). Este principalmente apunta a máquinas virtuales KVM (Kernel Virtual Machines, otro hipervisor basado en el kernel Linux), pero también maneja Xen y LXC (Linux containers, virtualización por software). Este presenta una vista resumen de los dominios en ejecución, su rendimiento en tiempo real y estadísticas de uso de recursos. Las utilidades permiten crear nuevas máquinas virtuales (dominios) y configuración y ajuste de las ubicaciones de recursos de los dominios y el hardware virtual. Incluye clientes visuales VNC y SICE para presentar una consola gráfica completa al dominio anfitrión.” [6] (Daniel P. Berrangé, 2021).

En el anexo 2 se puede ver la actual implementación y el funcionamiento de de virt-manager y un servidor ejecutando el hipervisor Xen. En él se pueden ver diversos servidores virtuales que realizan funciones importantes para la carrera, por ejemplo ‘meucci’ que es el servidor de telefonía IP del departamento.

## **3.3. OpenStack**

OpenStack en su documentación oficial se define como “un sistema operativo de la nube que permite controlar grandes grupos de recursos informáticos, de almacenamiento y de redes en todo un centro de datos, todo administrado a través de un panel que brinda control a los administradores al tiempo que permite a sus usuarios aprovisionar recursos a través de una interfaz web.

Más allá de una estructura estándar y la funcionalidad de infraestructura como servicio, los componentes adicionales proveen de orquestación, administración de fallas y servicio de administración entre medio y otros servicios para asegurar la alta disponibilidad de las aplicaciones de usuario.” [7](Openstack, 2021).

Para comprender mejor la visión que tiene la organización de OpenStack acerca de la computación en la nube revisaremos su documentación, para luego detallar de forma técnica como esa visión es llevada a cabo dentro del sistema.

## **Los Pilares de la Nube**

“La computación en la nube promueve una utilización de recursos más eficiente mediante la reducción de los costos de transacción envuelta en un aprovisionamiento y desaprovisionamiento de la infraestructura a casi cero, y es capaz de hacerlo debido a que difiere en forma cualitativas de modelos anteriores de computación en la nube (incluyendo virtualización).” [10] (OpenStack, 2021).

### **Autoservicio**

“Las nubes son de autoservicio. Ellas proveen a los usuarios la habilidad de desplegar aplicaciones en la nube en demanda sin tener que esperar a la interacción humana o al ciclo de revisión. La nube no tiene rastreadores de tickets. Estos requerimientos han tenido un número de consecuencias inmediatas.

Primero, la nube debe proveer multi-alquiler. En orden de asegurar el servicio a múltiples usuarios (o grupo de usuario) sin ninguna revisión humana, los recursos deben estar aislados entre arrendatarios del sistema, así entonces los recursos controlados por un arrendatario no tendrían ningún acceso para impactar en los recursos controlados por otros arrendatarios.” [10] (OpenStack, 2021).

### **Control de aplicaciones**

“Las nubes permiten el control de una infraestructura de aplicación para ser conferido a la propia aplicación. Solo la nube puede eliminar la necesidad de un humano del ciclo de aprobación, entonces elimina la necesidad de un humano de estar en el ciclo. Mientras una nube podría tener una interfaz gráfica de usuario, esta debe tener una interfaz de programación de aplicaciones. Este debería proveer información operacional relevante de una forma que es legible para otras aplicaciones, incluyendo notificaciones de eventos cuando sea apropiado. Este también debería designar la facilidad de asegurar el acceso a las APIs para aplicaciones que están ejecutándose en la misma nube, porque ninguna parte de la aplicación debería residir fuera de la nube.” [10] (OpenStack, 2021).

### **Consideraciones específicas de OpenStack**

“La mayor parte de las nubes propietarias son operadas por un software designado por una sola organización. OpenStack es diferente, existen muchas nubes OpenStack, ambas públicas y privadas, cada una operada por diferentes organizaciones con diferentes objetivos y tomando diferentes decisiones. Estas nubes pueden tener conjuntos de usuarios superpuestos. Esto lleva a algunos requerimientos que son específicos de OpenStack, y puede que no compartido con otras nubes.” [10] (OpenStack, 2021).

## **Interoperabilidad**

“La misma descripción de aplicaciones debería ser desplegable, solo con un mínimo, modificaciones bien aisladas, a una variedad de nubes públicas y privadas. Donde los requerimientos comunes de los patrones de despliegue de OpenStack difiere, en este sistema puedes esforzarte en diseñar mecanismos que son usables en cada uno de los ambientes, así las aplicaciones pueden ser operadas desde una nube OpenStack a otra.” [10] (OpenStack, 2021).

## **Escalación continua e infinita**

“OpenStack se esfuerza por proporcionar a los desarrolladores de aplicaciones interfaces que les permitan, en principio, escalar de manera eficiente desde cargas de trabajo muy pequeñas a muy grandes sin volver a diseñar sus aplicaciones.” [10] (OpenStack, 2021).

## **Interfaz gráfica de usuario**

“Es la mejor forma para que nuevos usuarios se acerquen a la nube y para usuarios en general de experimentar con áreas poco familiares de ella. Presentando opciones y flujos de trabajo de forma gráfica solventa el descubrimiento de nuevas capacidades en forma de lecturas a través de la API o formas que la documentación del CLI no pueden. Una GUI también es la mejor forma para usuarios experimentados como usuarios de la nube y operadores de la nube de tener un resumen completo del estado de los recursos de la nube y de visualizar las relaciones entre ellas. Por estas razones, en adición de la API y cualquier otra interfaz de usuario, Openstack debe incluir la interfaz de usuario basado en la web.” [10] (OpenStack, 2021).

Muchos más son los objetivos que comprometen a hacer un sistema de computación más libre y al alcance de todos. Para poder seguir leyendo acerca de su visión se recomienda leer su documentación oficial. En general provee todos los beneficios históricos que nos ha dado el software libre al desarrollo de la tecnología a lo largo de los años.

### **3.3.1. Funcionamiento de OpenStack**

Según la documentación actual de OpenStack el sistema “consiste en varios servicios claves que son instalados separadamente. Estos servicios trabajan en conjunto dependiendo de las necesidades de tu nube e incluye los servicios de Compute, Identity, Networking, Image, Block Storage, Object Storage, Telemetry, Orchestration y servicios de bases de datos (siendo estos los básicos entre muchos más servicios opcionales). Puedes instalar cualquiera de estos proyectos separadamente y configurarlos de forma independiente o como entidades conectadas.” [11] (Documentación OpenStack, 2021)

Actualmente Openstack funciona sobre tres sistemas operativos diferentes: OpenSUSE, SUSE Linux Enterprise Server, Red Hat Enterprise Linux, CentOS y Ubuntu.

Para entender a detalle los componentes que componen OpenStack se describirán de forma resumida cada una de ellas.

Para apoyar la descripción en la siguiente figura se deja un esquema conceptual que resume en gran parte el funcionamiento básico de una instalación de OpenStack. En él podemos ver los componentes básicos de funcionamiento y se omiten varios de los servicios extras que suelen ser menos comunes y más de configuración adicional (ver figura 4). Para el caso especial de este despliegue se omitirá el Object Storage, también el Block Storage y se agregara Placement debido a que el proceso de instalación básica lo requiere. Por tanto se trabaja en los siguientes componentes:

- Keystone: Identity service
- Glance: Image service
- Placement service
- Nova: Compute service
- Neutron Networking service
- Horizon: Dashboard service

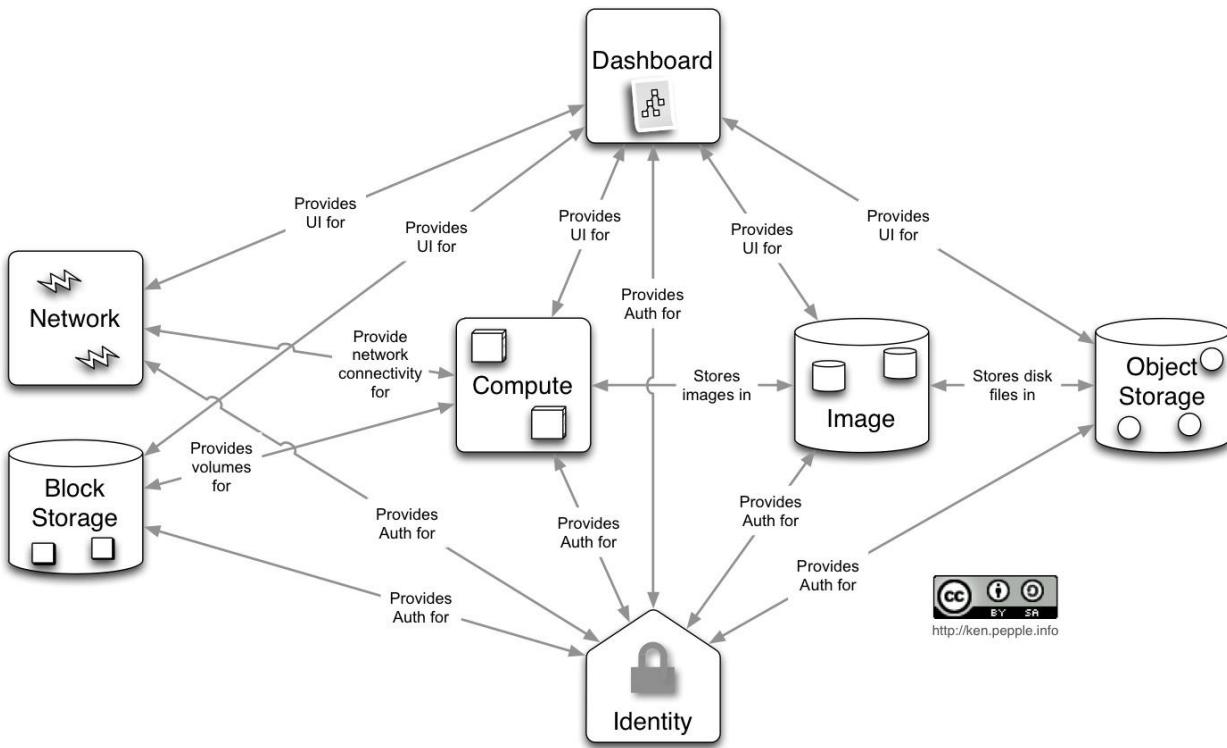


Figura 3 - Arquitectura conceptual de OpenStack, por Ken Pepple.

### Keystone: Identity service

“Keystone es un servicio de OpenStack que provee un cliente API de autorización, servicio de descubrimiento y autorización multi-arrendatario mediante la implementación de esta API.” [11] (Documentación OpenStack, 2021).

La función de Keystone es autenticar cada una de las operaciones necesarias del sistema. Por ejemplo, si un componente del sistema quisiera consultar información a otro servicio, este primero debe autenticarse

mediante Keystone para obtener autorización, luego podrá ejecutar las peticiones. En resumen se puede decir que Keystone es un autenticador para todo el sistema, lo que es crucial para la seguridad y la sinergia entre distintos componentes interrelacionados.

### **Glance: Image service**

“El servicio de imagen conocido como Glance es un proyecto que provee un servicio donde los usuarios pueden subir y explorar recursos que tienen como propósito ser usados con otros servicios. Esto actualmente incluye imágenes y definición de metadatos.”

Imágenes: Las imágenes de Glance incluyen descubrimiento, registro y recuperación de imágenes de máquinas virtuales (VM). Glance posee un API RESTful que permite consultar metadatos de máquinas virtuales así como también recuperar esa imagen.” [11] (Documentación OpenStack, 2021)

En resumen Glance se encarga de administrar los metadatos e imágenes de las máquinas virtuales que existen en el sistema de OpenStack. Esto es útil por ejemplo cuando se requiere un tipo de máquina virtual con una configuración preparada con el stack de desarrollo LAMP (Linux, Apache, MySQL, PHP) sobre un entorno Ubuntu, dentro del sistema glance puede registrar una máquina virtual Ubuntu y luego tomar una captura de la imagen una vez tenga el stack disponible, luego mediante esta captura, para luego compartir con los demás usuarios de OpenStack esta imagen.

### **Placement**

“Esta es un stack de API REST y modelado de información usado para hacer el seguimiento e inventario de los recursos proveedores y su uso, entre diferentes tipos de recursos. Por ejemplo, un recurso proveedor puede ser un nodo de cómputo, un grupo de almacenamiento compartido o un grupo de ubicaciones IP. El servicio Placement sigue el inventario de uso de cada proveedor. Por ejemplo, una instancia creada en un nodo de cómputo puede consumir recursos como la RAM y el CPU desde un nodo proveedor de cómputo, disco y desde un proveedor de grupo de almacenamiento compartido y direcciones IP de un proveedor de grupo de IP externo.”

Los tipos de recursos consumidos son seguidos como clases. El servicio provee un set de estructuras estándar (por ejemplo DISK\_GB, MEMORY\_MB y VPCU) y provee la habilidad de definir recursos personalizados y clases como sea necesario.

Cada proveedor de recursos también puede tener un conjunto de características que describen aspectos cualitativos del proveedor de recursos. Los rasgos describen un aspecto de un proveedor de recursos que no se puede consumir en sí mismo, pero que una carga de trabajo puede desear especificar. Por ejemplo, los discos disponibles pueden ser unidades de estado sólido (SSD).” [11] (Documentación OpenStack, 2021).

### **Nova: Compute service**

“Nova es el proyecto de OpenStack que provee una forma de aprovisionar instancias de cómputo (conocidas como servidores virtuales). Nova soporta la creación de máquinas virtuales, servidores bare metal, y posee un limitado soporte para sistemas de contenedores. Nova puede correr como servicio sobre un servidor Linux para proveer el servicio. Requiere de Keystone, Glance, Neutron, Placement para funcionar y Horizon, también del cliente de Nova para ser usuario por un usuario, también posee un API con la que se puede interactuar.” [11] (Documentación OpenStack, 2021).

Nova es el corazón del funcionamiento de OpenStack, en él finalmente se ejecutan las máquinas virtuales y se lleva la carga de trabajo, la cual es manejada por el servicio y por el aporte de los demás servicios en conjunto.

### **Neutron: Networking service**

“Neutron es un proyecto de OpenStack que provee ‘conectividad de red como servicio’ entre dispositivos interfaces (por ejemplo, vNICs) manejado por otro servicio (por ejemplo, Nova). Este implementa API de interconexión de redes de OpenStack.” [11] (Documentación OpenStack, 2021)

Neutron maneja toda la interacción que tiene el sistema con las redes entre sus componentes así también como las redes de las máquinas virtuales y sus configuraciones y reglas de funcionamiento. Esto se puede ver por ejemplo cuando se solicita un grupo de direcciones IP privadas para un número de máquinas virtuales, las cuales no son compartidas con otras máquinas virtuales del sistema y al mismo tiempo para la salida al internet pública de la máquina virtual existe otra dirección IP (fixed IPs), de todos estos trabajos de red está encargado Neutron.

### **Horizon: Dashboard service**

“Horizon es la implementación canónica del panel de administración de OpenStack, el cual provee una interfaz web de usuario para servicios como Nova, Swift, Keystone, etc.” [11] (Documentación OpenStack, 2021).

Horizon es la interfaz de usuario donde podemos administrar las partes esenciales del sistema, es la puerta de entrada de OpenStack y el servicio donde clientes y administradores pueden usar OpenStack con las facilidades que posee una interfaz de usuario y la conexión web.

En resumen es fundamental comprender el funcionamiento de cada uno de estos componentes para desarrollar el trabajo de implementación de OpenStack. Estos tienen sus funciones definidas y son sistemas individuales que trabajan en un ecosistema relacionado mediante mecanismo de interacción como lo son la conexión a internet y el servicio de autenticación. Ahora que todos los conceptos clave han sido introducidos se puede proceder a la implementación del sistema.

## **IV. Desarrollo**

El trabajo realizado en su desarrollo se compone por diversos pasos llevados uno sobre el otro para completar el objetivo final. La extensión en detalle del proceso de desarrollo se encuentra en **IX. Anexo del Desarrollo**, donde se revisa de forma técnica los comandos ocupados para realizar la implementación. En resumen y utilizando la metodología creada se realizó el trabajo de forma secuencial de la siguiente forma:

### **4.1. Diseño de arquitectura de sistemas para computación en la nube**

En este proceso se realiza la arquitectura computacional necesaria para operar el sistema de computación en la nube OpenStack. Esta sección se centra en disponer la capacidad computacional para poder instalar el sistema de computación en la nube, también acerca de los pre-requisitos del software necesario para que opere el sistema.

#### **4.1.1. Requerimientos de hardware**

Mediante la revisión de la infraestructura presente en el departamento de carrera y también sujeto a los requerimientos de hardware de OpenStack. A partir de esto se realiza una verificación de si los criterios necesarios están validados para confirmar la posibilidad de instalar OpenStack.

#### **4.1.2. Servidor de Control ó Nodo Controlador**

En este punto y el subsiguiente se revisa la fichas técnicas oficiales de los fabricantes de los servidores existentes en el departamento. Se realiza una tabla con los requerimientos presentados en documentación oficial de OpenStack con las especificaciones técnicas a modo de validación.

#### **4.1.4. Elección de redes de autoservicio**

Esta sección se centra en la elección de ambiente de red y posibilidades de interconexión de redes que ofrece OpenStack. Existen dos posibilidades de despliegue de redes virtuales dentro del sistema. Se revisan las opciones y se escoge la segunda la llamada ‘red de autoservicio’ la que ofrece mayor capacidades de configuración debido a la utilización de tecnologías que operan sobre la capa 3 del modelo OSI. Además esta opción abarca la primera, lo cual servirá de para revisar de forma exploratoria.

#### **4.1.5. Configuración del ambiente**

La configuración de ambiente es toda la configuración de software necesario para poder instalar OpenStack, requerido en su documentación oficial. Se procede a revisar la configuración de red recomendada en la documentación y se realiza un diseño en un diagrama de redes como será la red dentro

del departamento. Una vez definida las IP del segmento 172.24.250.0/24 como red de proveedores (definida por OpenStack como un enlace directo a internet y definido como un espacio de red para OpenStack) y el segmento 192.168.4.0/24 como red de administración donde se comparte el enlace con diversos servicios del departamento. Esto para obtener redundancia de interfaces, lo que permite tolerancia a fallas y mayor control de la instalación. Finalmente se revisa la conexión entre servidores mediante pruebas con el comando ping.

#### **4.1.6. Network Time Protocol (NTP)**

Como se indica en la documentación oficial los servidores de OpenStack deben estar sincronizados respecto al control de tiempo, esto para operar entre los distintos servidores. En esta sección se realiza una configuración central en el nodo de control y de esclavo en el nodo de cómputo. Esto mediante la configuración del servicio chrony.

#### **4.1.7. Paquetes de OpenStack**

En esta sección se configuran los repositorios de los sistemas operativos para poder obtener el software oficial de OpenStack en su versión 15 ‘Stein’. Mediante el gestor de paquetes zypper se configura y se verifica la integridad de los paquetes, se verifica esto mediante la instalación del cliente de OpenStack.

#### **4.1.8. Base de datos SQL**

Esta sección es acerca del servidor de bases de datos necesario para el sistema. Se procede a instalar mariadb que es uno de los recomendados entre otras opciones como MySQL. Se configura una instalación básica del sistema de bases de datos como validación.

#### **4.1.9. Cola de mensajes**

La cola de mensajes es un sistema para coordinar las operaciones y procesos requerido por OpenStack en diversos nodos. Se revisa la documentación de estos sistemas para proceder a instalar el servicio rabbitmq.

#### **4.1.10. Memcached**

Este es un sistema de memoria caché de objetos requerida por OpenStack para operar. Se instala el servicio memcached y se realiza una configuración básica para operar.

#### **4.1.11. Etcd**

Es un sistema de almacenamiento de llaves y valores utilizado por OpenStack. Se procede a configurar el servicio etcd que ya se encuentra instalado en el sistema. Se valida mediante la revisión de su estado en systemctl. Con este servicio se da por finalizada la configuración del ambiente y se procede a instalar el software relacionado con OpenStack.

### **4.2. Desplegar IaaS mediante OpenStack**

Esta sección y las siguientes subsecciones se centran en instalar y configurar los servicios de OpenStack necesarios para una instalación mínima. Para ello se procede a revisar la documentación y diversas fuentes con tal de que los sistemas configurados funcionen adecuadamente. Una vez finalizado uno se procede al siguiente hasta que el sistema esté completo en su versión mínima. Para detalles a mayor profundidad e instrucciones técnicas dirigirse a la sección IX. Anexo del Desarrollo.

#### **4.2.1. Keystone: Identity service**

Keystone es el sistema de autenticación de OpenStack, es el primero en instalarse. Este se encarga de autorizar e identificar a los usuarios, así también como las operaciones realizadas por los demás servicios. Debido a que cada operación dentro del sistema es autenticada es el primer servicio a instalar.

#### **4.2.2. Glance: Image service**

Glance es el sistema de imágenes de máquinas virtuales. Estos son recursos utilizados por el sistema en especial por Nova para crear máquinas virtuales. Glance se encarga de registrar y disponer de imágenes de sistema operativo, así como también de su configuraciones, copias y respaldos.

#### **4.2.3. Placement service**

El servicio placement es un sistema de seguimiento de inventario para el sistema OpenStack. Al igual como todos los servicios funciona mediante un modelo de API. Se encarga de revisar los recursos de hardware, inventarios, usos del sistema entre otros.

#### **4.2.4. Neutron Networking service**

Neutron es un sistema de redes para OpenStack. Se encarga de todos los aspectos relacionados a la interconexión de redes así también como de las redes virtuales para las máquinas virtuales. Es uno de los sistemas más complejos y su instalación se revisa en profundidad. Su instalación es valida mediante comandos de OpenStack y también en la sección V. Resultados.

#### **4.2.5. Nova: Compute service**

Nova es el sistema que lleva a cabo las virtualizaciones y mediante QEMU/KVM se encarga de virtualizar y llevar a cabo las operaciones sobre máquinas virtuales. Este está interrelacionado con todos los servicios de OpenStack y requiere de configuración. Se revisa su configuración a profundidad y se valida su instalación mediante la creación de una máquina virtual de prueba. Además se revisa su funcionamiento en la sección V. Resultados.

#### **4.2.6. Horizon: Dashboard service**

Horizon es el cliente de OpenStack. Este accede mediante un API a los demás servicios para ir mostrando una interfaz web que permite acceder a las configuraciones más generales. Provee de un sistema de fácil acceso y permite utilizar el sistema de forma automatizada. Se puede revisar su utilización en la sección IX. Anexo del Desarrollo y también se puede probar en la IP 192.168.4.47.

### **V. Resultados y discusión**

#### **5.1. Prueba de funcionamiento de la implementación**

Esta sección es una prueba del funcionamiento del sistema implementado. Probaremos la creación de los dos tipos de redes, pero se usará solo el primer tipo a modo de prueba, debido a problemas con la configuración de este tipo de red que se revisará en la sección V. Resultados.

Primero crearemos las redes virtuales. Primero la red de proveedores y en segundo lugar la de autoservicio.

“Antes de lanzar una instancia, debes crear la infraestructura de red virtual necesaria. Para operación interconectividad 1 una instancia usa la red del proveedor (externo) para conectarse a la infraestructura física mediante la capa 2 (bridging/switching). Esta red incluye un servidor DHCP que provee direcciones IP a las instancias” [13] (OpenStack Documentation, 2021).

## Networking Option 1: Provider Networks

Overview

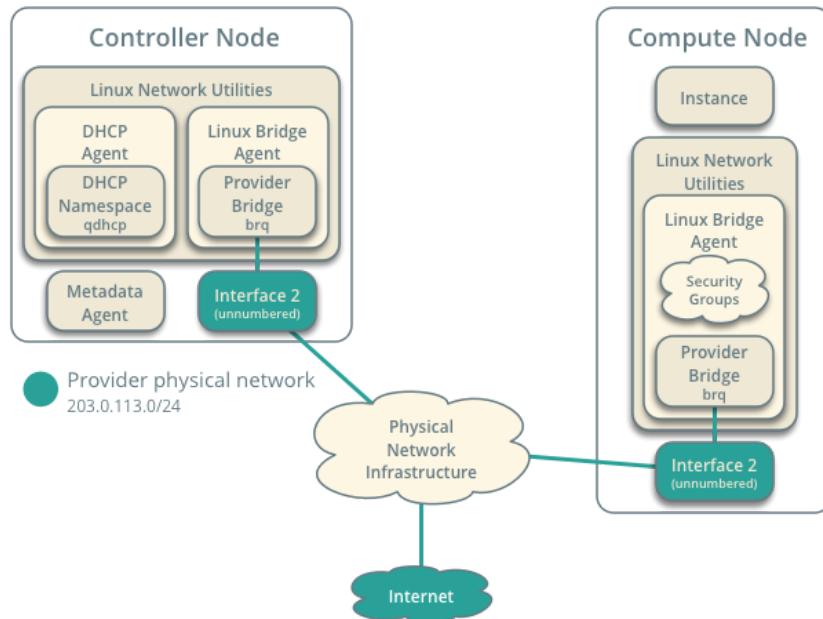


Figura 4 - Captura de pantalla funcionamiento de redes de proveedor. Fuente: documentación OpenStack.

Creamos la red de proveedor:

```
. admin-openrc

openstack network create --share --external \
--provider-physical-network provider \
--provider-network-type flat provider

openstack subnet create --network provider \
--allocation-pool start=172.24.250.104 ,end=172.24.250.254 \
--dns-nameserver 8.8.8.8 --gateway 172.24.250.1 \
--subnet-range 172.24.250.0/24 provider
```

Ahora procederemos a crear la red de autoservicio. “Las redes de autoservicio se conectan a la infraestructura de red física mediante NAT. Esta red incluye un servidor DHCP que provee direcciones a las instancias”. [13] (OpenStack Documentation, 2021).

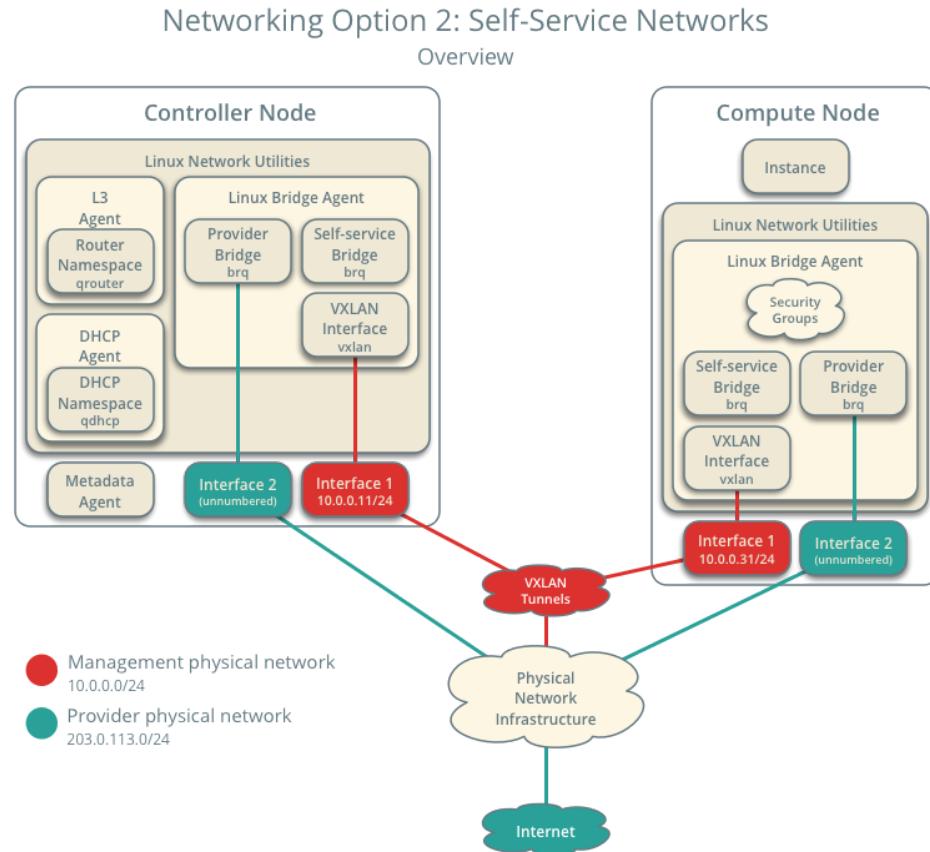


Figura 5 - Captura de pantalla funcionamiento de redes de autoservicio. Fuente: documentación OpenStack.

Creamos la red de autoservicio:

```
openstack network create selfservice
openstack subnet create --network selfservice \
--dns-nameserver 8.8.8.8 --gateway 172.24.252.1\
--subnet-range 172.24.252.0/24 selfservice
```

Creamos un router “Las redes de autoservicio se conectan a las redes de proveedor usando routers virtuales que realizan un NAT bidireccional. Cada router contiene una interfaz y al menos una red de autoservicio y un gateway a la red de proveedor”. [13] (OpenStack Documentation, 2021)

```
openstack router create router
openstack router add subnet router selfservice
openstack router set router --external-gateway provider
```

Verificamos la operación:

ip netns

```
yondu:~ # ip netns
qrouter-5989ba66-4eb4-417a-bb41-d498cd9b5c2c (id: 2)
qdhcp-bfab86f3-6ae1-4a44-96f3-d444cfbbe6a5 (id: 0)
qdhcp-a4faaa24e-6a06-4306-bfcfcd-4bc0cb38afaf (id: 1)
```

Figura 6 - Captura de pantalla verificación funcionamiento de la red de autoservicio. Fuente propia.

```
yondu:~ # openstack network list
+-----+-----+-----+
| ID | Name | Subnets |
+-----+-----+-----+
| a4faaa24e-6a06-4306-bfcfcd-4bc0cb38afaf | provider | 8819e188-2007-4d78-ab86-4a45debb0c6a |
| bfbab86f3-6ae1-4a44-96f3-d444cfbbe6a5 | selfservice | b0d08acc-45d0-4224-adf6-2b720d4ba956 |
+-----+-----+-----+
```

Figura 7 - Captura de pantalla verificación funcionamiento de la red de autoservicio. Fuente propia.

Listamos los puertos de los router para determinar la IP del gateway en la red de proveedor:

openstack port list --router router

```
yondu:~ # openstack port list --router router
+-----+-----+-----+
| ID | Name | MAC Address | Fixed IP Addresses
|     |       | Status |
+-----+-----+-----+
| ad3324dd-1603-4c66-9517-1f0a1fcff956 | fa:16:3e:27:37:24 | ip_address='172.24.250.106', su
bnet_id='8819e188-2007-4d78-ab86-4a45debb0c6a' | ACTIVE |
| ec1dc236-4546-480a-8087-91ec986d2bd3 | fa:16:3e:33:83:81 | ip_address='172.24.252.1', subn
et_id='b0d08acc-45d0-4224-adf6-2b720d4ba956' | ACTIVE |
+-----+-----+-----+
yondu:~ # ping 172.24.250.106
PING 172.24.250.106 (172.24.250.106) 56(84) bytes of data.
64 bytes from 172.24.250.106: icmp_seq=1 ttl=64 time=0.053 ms
64 bytes from 172.24.250.106: icmp_seq=2 ttl=64 time=0.040 ms
64 bytes from 172.24.250.106: icmp_seq=3 ttl=64 time=0.034 ms
^C
--- 172.24.250.106 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2041ms
rtt min/avg/max/mdev = 0.034/0.042/0.053/0.009 ms
```

Figura 8 - Captura de pantalla verificación funcionamiento de los routers virtuales. Fuente propia.

Ahora que están configurada las redes virtuales procederemos a configurar las características de la máquina virtual.

Para poder utilizar el sistema implementado se desplegará un sistema operativo ubuntu 18.04. “La forma más fácil de obtener una imagen virtual que funciona en OpenStack es descargando alguna que alguien ya haya creado. La mayor parte de estas imágenes contienen el paquete cloud-init que soporta pares de llaves SSH e inyección de datos de usuarios.” [16] (OpenStack Image Guide, 2021).

Además de obtener la imagen es necesario crear la configuración de acceso y seguridad para las instancias. “Antes de lanzar una instancia, deberías agregar al grupo de seguridad para habilitar a los

usuarios realizar pruebas con ping y conexión SSH.” [17] (OpenStack, Configure access and security for instances, 2021).

Procedemos a descargar la imagen adecuada para OpenStack de Ubuntu 18.04:

```
wget https://cloud-images.ubuntu.com/bionic/current/bionic-server-cloudimg-amd64.img
```

Creamos la imagen con el servicio de Imágen Glance:

```
openstack image create "ubuntu 18.04 LTS" --file bionic-server-cloudimg-amd64.img --disk-format qcow2 --container-format bare --public
```

Creamos el flavor, que vendrían a ser las especificaciones técnicas del hardware virtual:

```
openstack flavor create --vcpus 1 --ram 1024 --disk 10 basic
```

Generamos los accesos por par claves de pares. Para acceder es necesario obtener llaves pública y privada:

```
ssh-keygen -q -N ""
openstack keypair create --public-key ~/.ssh/id_rsa.pub ubuntu-wordpress
```

Verificamos la creación de las llave:

```
openstack keypair list
```

yondu:~ # openstack keypair list	
Name	Fingerprint
centos	4b:31:14:36:f8:af:bb:73:f9:cd:d3:ed:25:40:d7:0f
cloudkey	16:d3:c8:20:20:88:74:5d:f5:c9:55:7c:ff:64:4f:3e
first-test-ssh	4c:9a:77:36:1c:d2:96:60:48:01:ac:c2:3c:68:6e:32
mykey	79:e9:4a:07:3b:f9:04:81:c7:93:16:de:15:3d:b5:c2
ubuntu-wordpress	f2:4a:5b:6b:6c:dd:23:cb:40:5c:64:89:7a:bc:5f:93
ubuntu-wordpress-2	33:34:bc:4f:ff:1e:27:2c:94:9a:61:f8:50:aa:c5:74
yondu	79:e9:4a:07:3b:f9:04:81:c7:93:16:de:15:3d:b5:c2

Figura 9 - Captura de pantalla verificación funcionamiento de los pares de claves. Fuente propia.

Agregamos las reglas de seguridad necesarias para probar la conexión mediante PING a la máquinas virtuales y el puerto destinado a la conexión SSH:

```
openstack security group rule create --proto icmp default
openstack security group rule create --proto tcp --dst-port 22 default
```

Lanzamos la instancia:

```
openstack server create --flavor basic --image ubuntu 18.04 \
--nic net-id=a4faa24e-6a06-4306-bfcd-4bc0cb38afaf \
--security-group default --key-name ubuntu-wordpress
```

Revisamos el estado del servidor:

```
openstack server list
```

ID	Name	Status	Networks	Image	Flavor
e6bc181-c9f4-41a4-b52c-e037219f8270	ubuntu-wordpress	ACTIVE	provider=172.24.250.130	ubuntu 18.04 LTS	basic
dfbb3f9c-89bc-4c13-9f59-4b0280e29e53	selfservice-instance	ACTIVE	selfservice=172.24.252.124	cirros	m1.nano

Figura 10 - Captura de pantalla verificación funcionamiento de los servidores virtuales. Fuente propia.

Accedemos al servidor mediante el link de conexión VNC:

```
openstack console url show ubuntu-wordpress
```

Field	Value
type	novnc
url	http://192.168.4.47:6080/vnc_auto.html?path=%3Ftoken%3D61dc50e7-17dc-4ae0-852a-42229cc188cd

Figura 11 - Captura de pantalla verificación funcionamiento de VNC. Fuente propia.

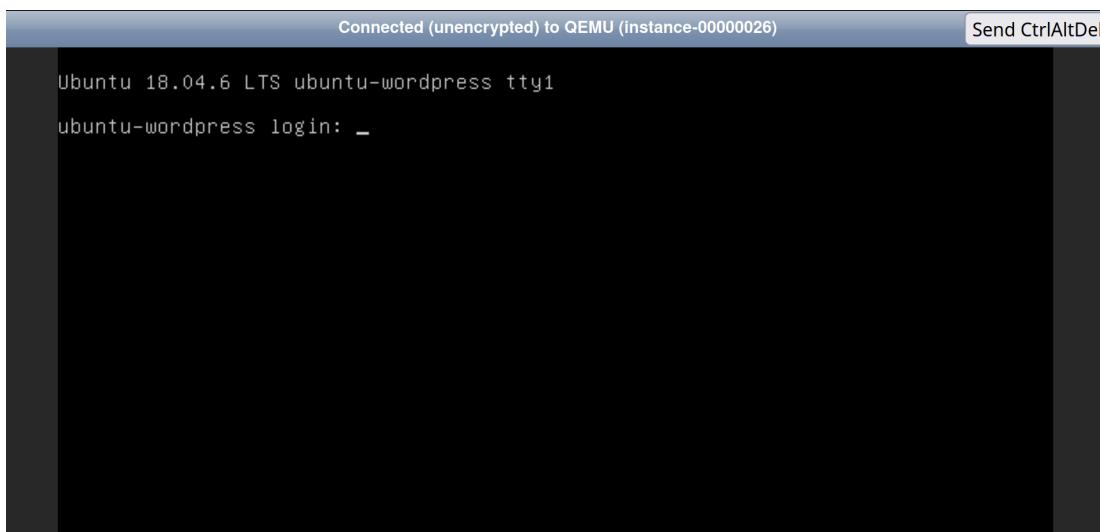


Figura 12 - Captura de pantalla verificación funcionamiento de VNC. Fuente propia.

Verificamos la conexión hacia la máquina virtual:

```
yondu:~ # ping 172.24.250.130
PING 172.24.250.130 (172.24.250.130) 56(84) bytes of data.
64 bytes from 172.24.250.130: icmp_seq=1 ttl=64 time=0.756 ms
64 bytes from 172.24.250.130: icmp_seq=2 ttl=64 time=0.671 ms
64 bytes from 172.24.250.130: icmp_seq=3 ttl=64 time=0.665 ms
^C
--- 172.24.250.130 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2055ms
rtt min/avg/max/mdev = 0.665/0.697/0.756/0.046 ms
```

Figura 13 - Captura de pantalla verificación funcionamiento de red máquina virtual. Fuente propia.

Accedemos con las llaves creadas anteriormente hacia el servidor:

```
yondu:~ # ssh -i ubuntu-wordpress.pem ubuntu@172.24.250.130
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-158-generic x86_64)

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

 System information as of Tue Sep 28 15:39:23 UTC 2021

 System load:  0.52           Processes:          85
 Usage of /:   18.3% of 9.52GB  Users logged in:   0
 Memory usage: 32%
 Swap usage:   0%
 IP address for ens3: 172.24.250.130

4 updates can be applied immediately.
4 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Thu Sep 23 14:37:32 2021 from 172.24.250.100
ubuntu@ubuntu-wordpress:~$ echo "hello openstack world"
hello openstack world
ubuntu@ubuntu-wordpress:~$ █
```

Figura 14 - Captura de pantalla verificación funcionamiento y acceso a servidor virtual ubuntu. Fuente propia.

Los resultados de la implementación y del desarrollo son un sistema de computación en la nube, basado en la infraestructura como servicio con OpenStack. Este da las posibilidades de ser usado dentro del departamento con los fines que sean pertinentes. Por ahora el resultado es la capacidad de usar el sistema de servidores virtuales dentro de la red del departamento. La capacidad de utilizar un entorno web para aprovisionar recursos y el explorar un gran abanico de posibilidades que ofrece la computación en la nube.

Si bien la prueba de la red de autoservicio no ha sido exitosa, se ha decidido que la red de proveedor es suficiente para alcance de este proyecto. Debido a que los objetivos se han cumplido y las posibilidades de la red de autoservicio serán características adicionales que escapan al proyecto.

Por tanto el resultado de la implementación considera OpenStack como una solución que expande la infraestructura del departamento de carrera. Que debe ser probada y utilizada, al margen de que cumpla su función y además de servir de objeto de estudio para la comunidad de la carrera.

## VI. Conclusiones

Los resultados obtenidos para la Implementación de Sistema de Computación en la Nube han sido completados para los objetivos que se buscaban. Si bien en un principio la finalidad era expandir la infraestructura del departamento de carrera utilizando OpenStack como servicio, este se ha transformado en una solución bastante robusta para cumplir tales funciones. Al ser una solución tan compleja requiere también de bastante trabajo el que opere correctamente, por lo cual cada servicio y característica adicional requerirá de tiempo adicional para poder configurar, desplegar y probar la característica para asegurar el adecuado funcionamiento. Por ejemplo el uso de IP elásticas es una característica que podría ser muy útil para que la implementación pueda ser ofrecida a través de internet pública, pero esta característica requiere configuración en diversos servicios, pruebas y finalmente despliegue el cuál puede tomar semanas o meses.

Si bien el alcance del objetivo de implementación se logró, este presentó bastantes dificultades que impidieron el desarrollo esperado de la implementación. Este sistema se comenzó a implementar en septiembre del 2019, previo al estallido social de octubre de ese año, posterior a ello, cuando se retomó el trabajo llegaron las dificultades de la pandemia, ya que en marzo cerraron la universidad y el acceso físico al datacenter. Pasó el 2020 con poco desarrollo y con muchas dificultades de acceso a las dependencias de la universidad, varias fueron las veces que era necesario reiniciar el servidor, ya que fallaban las interfaces de red, las fuentes de poder debido a los cortes de suministro eléctrico. Este tipo de trabajo requiere presencialidad, por lo cual, cuando el acceso estuvo más expedito y se coordinó la una restauración de los servidores con el profesor guía en diciembre del 2020 se retomó de lleno la implementación de OpenStack, teniendo los mismos problemas de acceso y de dificultades debido al estado de excepción y las consecuencias de la pandemia. Finalmente luego de meses de configuración y viajes de trabajo en terreno se cumplió con el objetivo dado en septiembre del 2021.

Para un sistema tan robusto y complejo, existe una necesidad de infraestructura más completa, contar por ejemplo con un uptime superior (tiempo que se encuentran en funcionamiento los servidores, idealmente 24 horas al dia, 7 dias a la semana o al menos 99% del tiempo al año) y de soporte de personas presencial que se encuentren físicamente cerca de los servidores. También de la necesidad de redundancia de fuentes de poder y de generadores de energía, debido a que cada vez que se cortaba el suministro eléctrico fallan las fuentes de poder. OpenStack debe ser implementado en una infraestructura de datacenter más completa, mantenida y en revisión constante, debido a que las operaciones que este realiza pueden ser cruciales o pueden ser muchas para alguna organización que la implemente como solución de producción. Y este al fallar requiere trabajo de solución de problemas adecuada.

Finalmente se concluye que OpenStack es una solución de computación de la nube al alcance de todos. Esta solución puede expandir la infraestructura tecnológica del departamento de carrera de Ingeniería Civil Informática. Pero para un funcionamiento adecuado es necesario contar con un datacenter más equipado y a prueba de fallos para su implementación.

## VII. Bibliografía

La bibliografía se compone de 17 recursos, de diversos tipos: libros, papers, documentación, guías y foros que se pueden encontrar en internet. Dentro de ello se destacan principalmente la documentación oficial de OpenStack que está bastante completa.

- [1] Jeff Daniels. (2009). Server Virtualization Architecture and Implementation. ACM Crossroads, 16, 8-12.
- [2] Omar Sefraoui, Mohammed Aissaoui, Mohsine Eleuldj. (2012, Octubre). OpenStack: Toward an Open-Source Solution for Cloud Computing. International Journal of Computer Applications, 55, 38-42.
- [3] Sushil Bhardwaj, Leena Jain, Sandeep Jain. (2010). Cloud Computing: A Study of Infrastructure as a Service (IAAS). International Journal of Engineering and Information Technology, 2, 60-63.
- [4] Ken Pepple. (2011). Deploying OpenStack. Sebastopol, CA 95472, United States of America: O'Reilly Media, Inc.
- [5] Xen Project. (2021). Virtualization. The beginnings of the hypervisor and Xen. 2021, abril, Virtualization - Xen Project Recuperado de <https://xenproject.org/users/virtualization/>
- [6] Daniel P. Berrangé. (2021). Manage virtual machines with virt-manager. 2021, abril, de Virtual Machine Manager Recuperado de <https://virt-manager.org/>
- [7] Openstack documentación oficial. (2021). Documentación de Openstack. 2021, abril, de OpenStack Docs Recuperado de <https://docs.openstack.org>
- [8] Óscar Ávila Mejía. (19 de mayo de 2011). Computación en la Nube. ContactoS, 80, 45 - 52.
- [9] Shivaji P. Mirashe, Dr. N.V. Kalyankar. (2010, Marzo). Cloud Computing. Journal of Computing, 2, 78 - 82.
- [10] OpenStack. (2021). Vision for OpenStack Clouds. 2021, mayo 18, de Openstack Governance Recuperado de <https://governance.openstack.org/tc/reference/technical-vision.html>
- [11] OpenStack. (2021). OpenStack Documentation. 2021, mayo 20, de OpenStack Docs: Preface Recuperado de <https://docs.openstack.org/install-guide/preface.html>
- [12] David Watts. (2015). System x3250 M4 Product Guide. 2021, mayo 28, de lenovopress.com Recuperado de <https://lenovopress.com/tips0812-system-x3250-m4>
- [13] OpenStack Documentation. (2021). OpenStack Docs: Install OpenStack Services. 2021, mayo 28, de OpenStack Documentation Recuperado de <https://docs.openstack.org/install-guide/overview.html>

[14] RabbitMQ. (2021). RabbitMQ is the most widely deployed open source message broker. 2021, junio 9, de RabbitMQ Recuperado de <https://www.rabbitmq.com/>

[15] Memcached. (2021). A distributed memory object caching system. 2021, junio 9, de What is Memcached? Recuperado de <https://www.memcached.org/>

[16] OpenStack. (2021). Get images. 2021, de OpenStack Documentation Recuperado de <https://docs.openstack.org/image-guide/obtain-images.html>

[17] OpenStack. (2021). Configure access and security for instances. 2021, de OpenStack Documentation Recuperado de

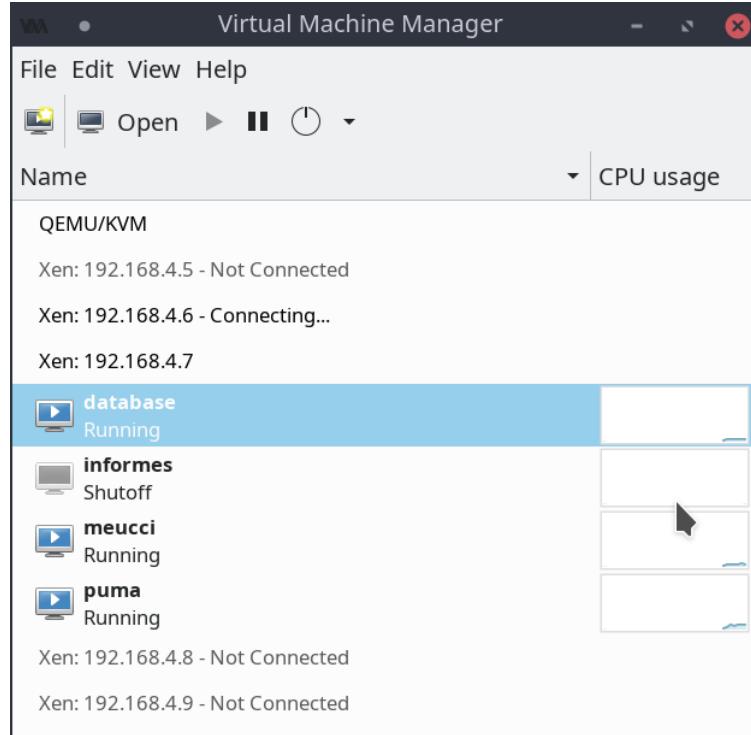
<https://docs.openstack.org/horizon/latest/user/configure-access-and-security-for-instances.html>

## VIII. Anexos

1.- Fotos del datacenter del Departamento de Ingeniería Civil Informática. En él se pueden apreciar los dos racks y los servidores.



2.- Utilización de Virtual Machine Manager en conjunto con hipervisor Xen. Se puede apreciar por ejemplo la base de datos de la carrera databases y el servidor de telefonía IP meucci.



```

Sep 13 12:38:37 db last message repeated 2 times
Sep 13 12:38:34 db sshd[31341]: error: PAM: authentication error for illegal use
r censei from 192.168.8.80
Sep 13 12:38:56 db last message repeated 2 times
Sep 13 12:40:03 db sshd[31347]: error: PAM: authentication error for illegal use
r exfuir from 192.168.8.80
Sep 13 12:40:03 db last message repeated 2 times
Sep 13 12:43:13 db sshd[31359]: error: PAM: authentication error for root from 1
92.168.8.80
Sep 13 12:43:17 db sshd[31359]: error: PAM: authentication error for root from 1
92.168.8.80
Sep 13 13:47:12 db sshd[31594]: error: PAM: authentication error for root from 1
92.168.8.80
Sep 13 13:47:13 db last message repeated 2 times
Sep 13 13:56:45 db sshd[31628]: error: PAM: authentication error for illegal use
r admin from 192.168.8.80
Sep 13 13:56:52 db last message repeated 2 times
Sep 13 13:57:05 db sshd[31633]: error: PAM: authentication error for illegal use
r admin from 192.168.8.80
Sep 13 13:57:11 db last message repeated 2 times
Sep 13 14:33:55 db sshd[31744]: error: PAM: authentication error for mmunoz from
192.168.8.168
Sep 13 14:34:10 db su: mmunoz to root on /dev/pts/0
Sep 13 14:35:14 db su: censei to root on /dev/pts/0

```

3.- Todos los servicios de OpenStack están listados en el siguiente enlace:  
<https://docs.openstack.org/wallaby/projects.html>

4.- Imágen comercial del servidor System x3250 M4



## IX. Anexo del Desarrollo

### Indice

9.1. Arquitectura de sistemas para computación en la nube	45
9.1.1. Requerimientos de hardware	45
9.1.2. Servidor de Control ó Nodo Controlador	46
9.1.3. Servidor Compute ó Nodo de Cómputo	47
9.1.4. Elección de redes de autoservicio	48
9.1.5. Configuración del ambiente	49
9.1.6. Network Time Protocol (NTP)	52
9.1.7. Paquetes de OpenStack	53
9.1.8. Base de datos SQL	54
9.1.9. Cola de mensajes	55
9.1.10. Memcached	56
9.1.11. Etcd	56
9.2. Desplegar IaaS mediante OpenStack	58
9.2.1. Keystone: Identity service	58
9.2.2. Glance: Image service	64

9.2.3. Placement service	68
9.2.4. Neutron Networking service	71
9.2.5. Nova: Compute service	79
9.2.6. Horizon: Dashboard service	89

## **9.1. Arquitectura de sistemas para computación en la nube**

El diseño de un sistema para computación en la nube está sujeto tanto a las necesidades de los usuarios finales, como a los requisitos del software que se implementará. Para lo primero y por la naturaleza de este trabajo, se realizará una configuración básica que nos sirva de forma exploratoria y pueda mostrar los potenciales de la computación en la nube de forma rápida y accesible. Esta arquitectura es conocida como la arquitectura mínima de ejemplo. Para lo segundo se ha revisado la documentación de OpenStack y se ha encontrado los requisitos de hardware existentes.

En resumen se puede decir que OpenStack necesita dos servidores relativamente modernos, una configuración de la red adecuada, conexión a internet y la infraestructura necesaria para alojar estos servidores y mantenerlos bien ventilados (como se puede ver en el anexo 1 el data center posee parte de estas capacidades). En los siguientes puntos se detalla los requerimientos y el equipo disponible para implementar la arquitectura necesaria para la implementación de computación en la nube.

### **9.1.1. Requerimientos de hardware**

Los requerimientos de hardware son dos servidores, dentro de las definiciones de OpenStack conocidos como ‘nodos’. El primero es el nodo controlador el cual debe poseer una CPU de 1 o 2 núcleos, 8GB de RAM, 100 GB de almacenamiento y 2 interfaces de red (más adelante se detalla la necesidad de contar con dos interfaces). El segundo nodo es el nodo de cómputo el cual debe tener 2, 4 o más núcleos de CPU, 8 o más GB de RAM, 100 o más GB de almacenamiento.

De forma adicional están los nodos adicionales los cuales sirven de Block Storage o de Object Storage según estos sean necesarios, poseen menos capacidades pero aportan almacenamiento de forma escalable. Estos serán omitidos debido que no son necesarios para la instalación mínima y pueden ser desplegados dentro del nodo de cómputo o de controlador (debido a la naturaleza de OpenStack podemos distribuir los

componentes del sistema en cualquier nodo a elección) y si fueran necesarios se agregarán al sistema fácilmente ya que OpenStack está preparado para agregar o borrar componentes.

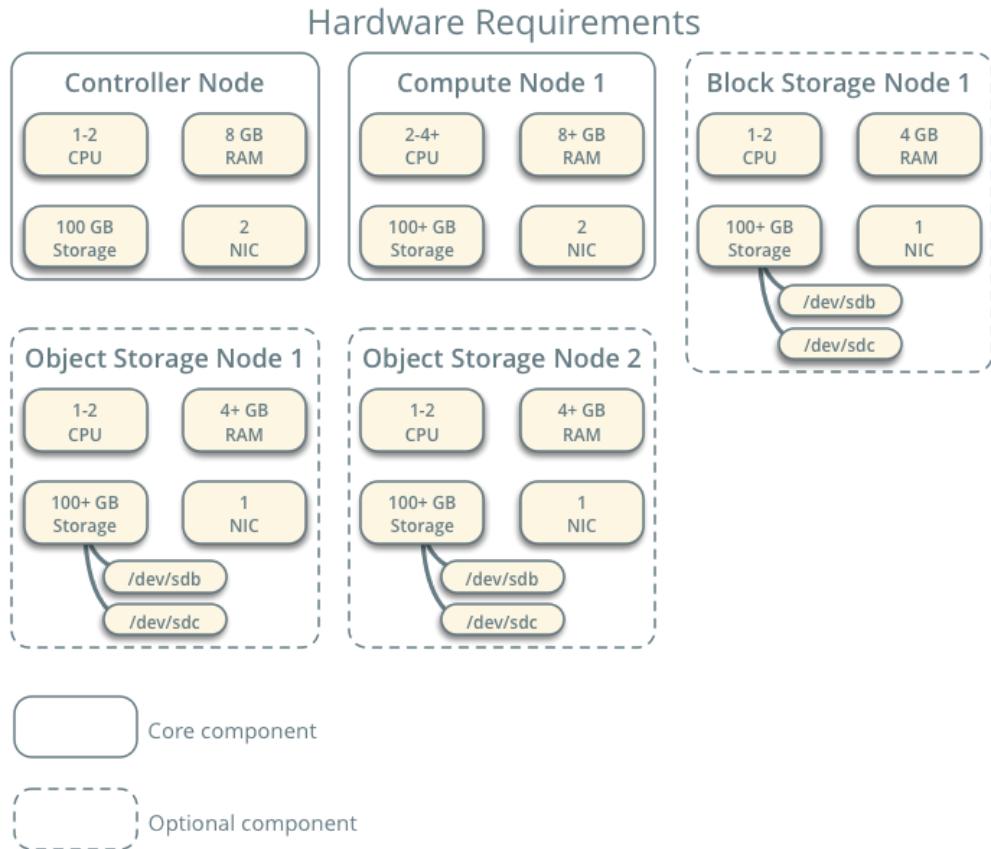


Figura 15 - Requerimientos básicos de OpenStack, por documentación oficial Openstack.

Los dos servidores a disposición que serán utilizados son dos equipos disponibles en el centro de datos del departamento de la carrera. Estos a pesar de no ser de última generación poseen la potencia computacional y el equipo de hardware suficiente para ejecutar el sistema OpenStack.

### 9.1.2. Servidor de Control ó Nodo Controlador

Este servidor se caracteriza por tener menos prestaciones, pero de albergar la mayor parte de componentes de OpenStack, esto porque como su nombre lo indica será utilizado como controlador, mientras que el servidor Compute será utilizado para realizar a cabo la virtualización, que es la tarea más demandante de recursos computacionales. “El nodo controlador ejecutará Keystone (Identity service), Glance (Image service), Placement service, porciones de administración de Neutron (Networking service), varios agentes de redes y Horizon (Dashboard). También incluirá servicios del ambiente como una base de datos SQL, la cola de mensajes y el servicio NTP”. [13] (OpenStack Documentation, 2021).

El servidor Lenovo System X3250 M4 (Anexo 3) (anteriormente IBM) “Es un servidor rack de 1U de un solo socket diseñado para pequeños negocios y compradores iniciales buscando una solución para mejorar la eficiencia de su negocio. Este entrega varias características innovadoras en una base compacta de 1U con un precio competitivo.” [12] (David Watts, 2015).

Componentes	Especificaciones	Requerimiento	Revisión
Nombre de producto	IBM System x3250 M4	-	-
Factor de forma	Sistema de rack 1U	-	-
Procesador	Xeon E3-1220 v2 (hasta 3.2 GHz/8 MB/1600 MHz) de cuatro núcleos	1 ó 2 núcleos de CPU	
Memoria RAM	8 GB DDR3 a 1600 Mhz	8 GB	
Almacenamiento	480 GB HDD	100 GB ó más GB	

Tabla 1 - Especificaciones técnicas Nodo Controlador. Fuente David Watts (Lenovo). Edición propia.

### 9.1.3. Servidor Compute ó Nodo de Cómputo

Para el servidor Compute se utilizará el mismo modelo con mayores prestaciones, en cuanto al aumento de la memoria RAM y del almacenamiento en disco duro. “Este ejecutará la porción del servicio de Nova (Compute) donde se encuentra el hipervisor y opera las instancias. Este nodo también ejecutará el servicio de los agentes de Neutron (Networking) que conecta las instancias a las redes virtuales y provee de servicios de firewall a través de los grupos de seguridad.” [13] (OpenStack Documentation, 2021).

Componentes	Especificaciones	Requerimiento	Revisión
Nombre de producto	IBM System x3250 M4	-	-
Factor de forma	Sistema de rack 1U	-	-
Procesador	Xeon E3-1220 v2 (hasta 3.2 GHz/8 MB/1600 MHz) de cuatro núcleos	2, 4 ó más núcleos de CPU	

Memoria RAM	16 GB DDR3 a 1600 Mhz	8 ó más GB	
Almacenamiento	1 TB HDD	100 GB ó más GB	

Tabla 2 - Especificaciones técnicas Nodo Controlador. Fuente David Watts (Lenovo). Edición propia.

#### 9.1.4. Elección de redes de autoservicio

“La opción de redes de autoservicio aumenta las opciones de redes de proveedor con el servicio de capa 3 (enrutamiento) que permite redes de autoservicio usando métodos de segmentación revestida como VXLAN. Esencialmente, enruta las redes virtuales a las redes físicas usando NAT. Adicionalmente, esta opción provee la base para servicios como LBaaS (Balanceador de Carga como Servicio) y FWaaS (Cortafuegos como Servicio).

El usuario de OpenStack puede crear redes virtuales sin el conocimiento de la infraestructura subyacente o la red de datos. Esto puede incluir redes VLAN si el plug-in de la capa 2 está configurado de esta forma.” [13] (OpenStack Documentation, 2021).

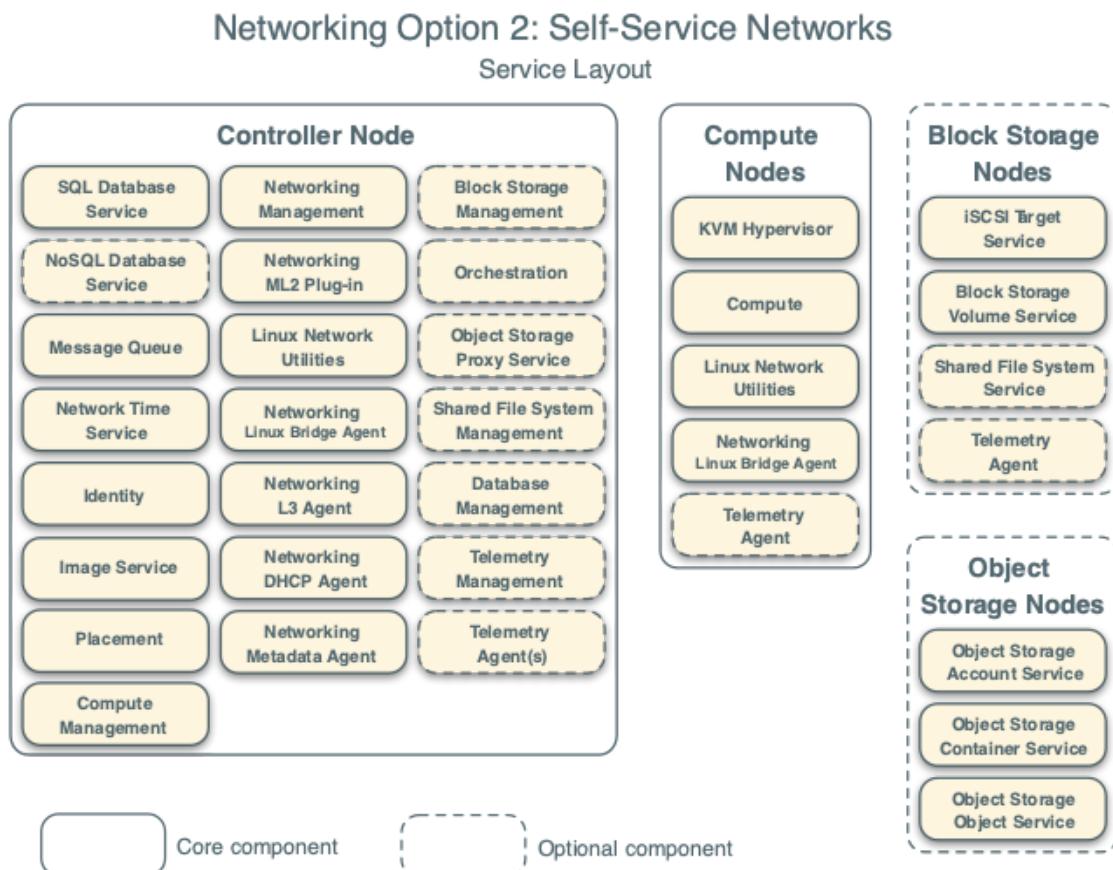


Figura 16 - Opción de redes de autoservicio, por documentación oficial Openstack.

Esta opción fue escogida debido a las características adicionales que provee la capa 3, además permite crear redes virtuales de manera más flexible para el usuario sin que este sepa el funcionamiento de la red subyacente donde se establece el sistema (lo que entrega facilidad de uso). Además por la posibilidad de poder probar todas las funcionalidades de OpenStack. Las consecuencias de esta elección para el desarrollo del trabajo se revisan en el punto correspondiente a Neutron y a Nova, que son los servicios cruciales donde se realiza la configuración de red y su elección de autoservicio.

### **9.1.5. Configuración del ambiente**

La elección del sistema operativo fue OpenSUSE Leap 15.1 x64 debido a que la instalación de las máquinas anfitrionas de la implementación anterior con Xen utilizan OpenSUSE. Por tanto se mantiene el sistema operativo de base para la implementación de la nube como una forma de hacer común el sistema de los servidores anfitriones.

#### **Red anfitriona**

Una vez ya instalado el sistema operativo con una configuración básica se procede en primer lugar a configurar las interfaces de red. Con el fin de que obtengan internet para instalar paquetes de software, actualizaciones de seguridad, DNS y NTP. Se definirá la red básica de anfitriones. Esta tendrá dos conexiones definidas las cuales usarán espacios de direcciones privadas y tendrán conexión a internet.

En primer lugar la red de proveedores usará el segmento 172.24.250.100.0/24 y estará conectada directamente a la red física (más información en configuración en switch Cisco del departamento de carrera). Esto hace que las IP de este segmento están destinadas solo a OpenStack y no incluyan ninguna dirección de servidores externos, además esto da la posibilidad a OpenStack de definir su propio DHCP y configuraciones de autoservicio.

En segundo lugar estará la red de administración la cual estará conectada por VLAN al segmento 192.168.4.0/24 (externo) el cual es compartido con los demás servidores del datacenter. Esta red servirá de redundancia en caso de fallar la primera interfaz.

## Network Layout

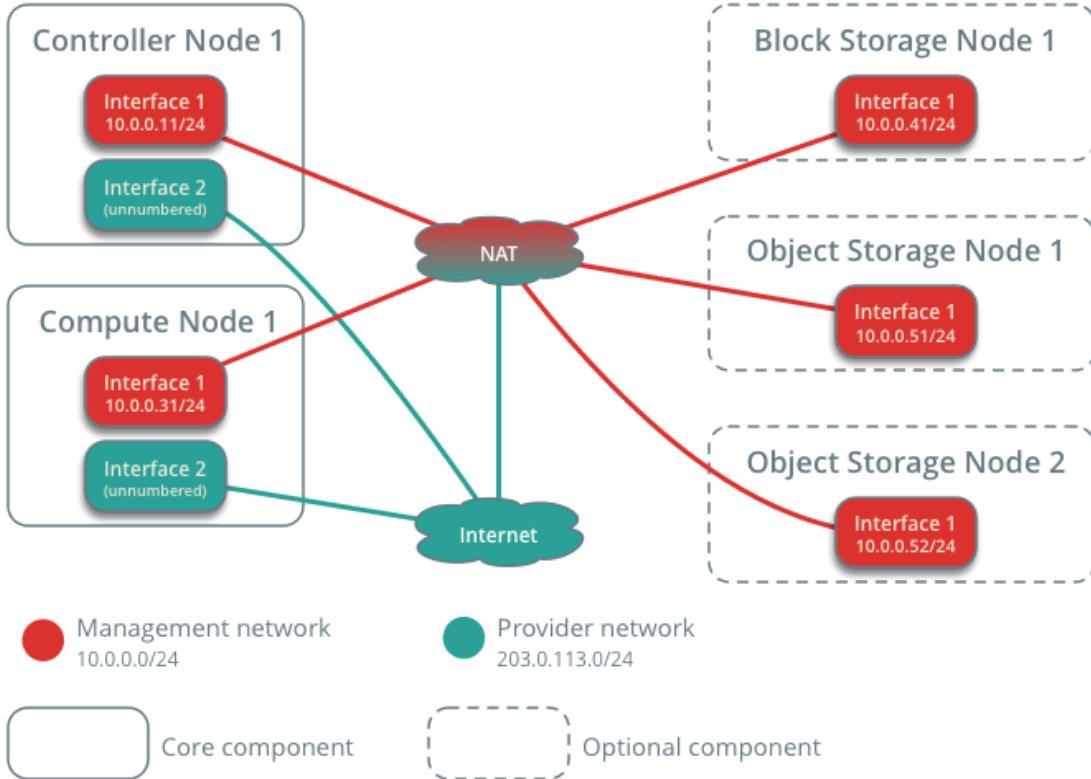


Figura 17 - Esquema de red de ambiente. Por Documentación OpenStack.

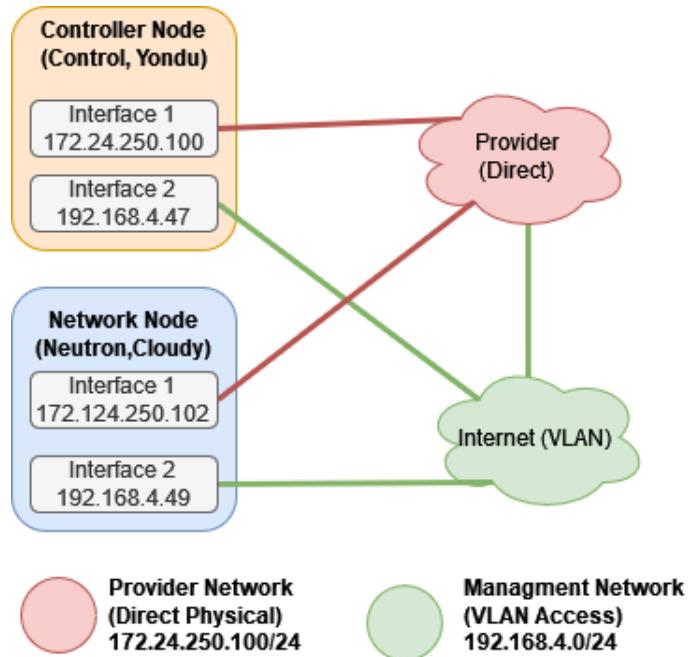


Figura 18 - Esquema de red propuesta para la implementación. Fuente propia.

La arquitectura definida posee las siguientes características:

- Red de proveedor y de máquinas virtuales en el segmento 172.24.250.100.0/24 con gateway en 172.24.250.100.1. Esta red posee acceso a internet.
- Red de administración en 192.168.4.0/24 con gateway en 192.168.4.1. Esta red también posee acceso a internet.

#### Configuración de red en nodos

Procedemos a configurar las interfaces de red del nodo de control. Para ello podemos utilizar la utilidad de configuración del sistemas incluida en OpenSuse llamada Yast:

```
sudo yast
```

Navegamos a sistema → Ajustes de red → Interfaz eth0 → F4 para editar y configuramos con las siguientes opciones:

Nodo/Configuración	Dirección IP interface eth0	Dirección IP interface eth1	Máscara	Nombre de host
Control	172.24.250.100	192.168.4.47	24	control
Compute	172.24.250.102	192.168.4.49	24	rocket

Tabla 3 - Configuración de red de nodos. Fuente propia.

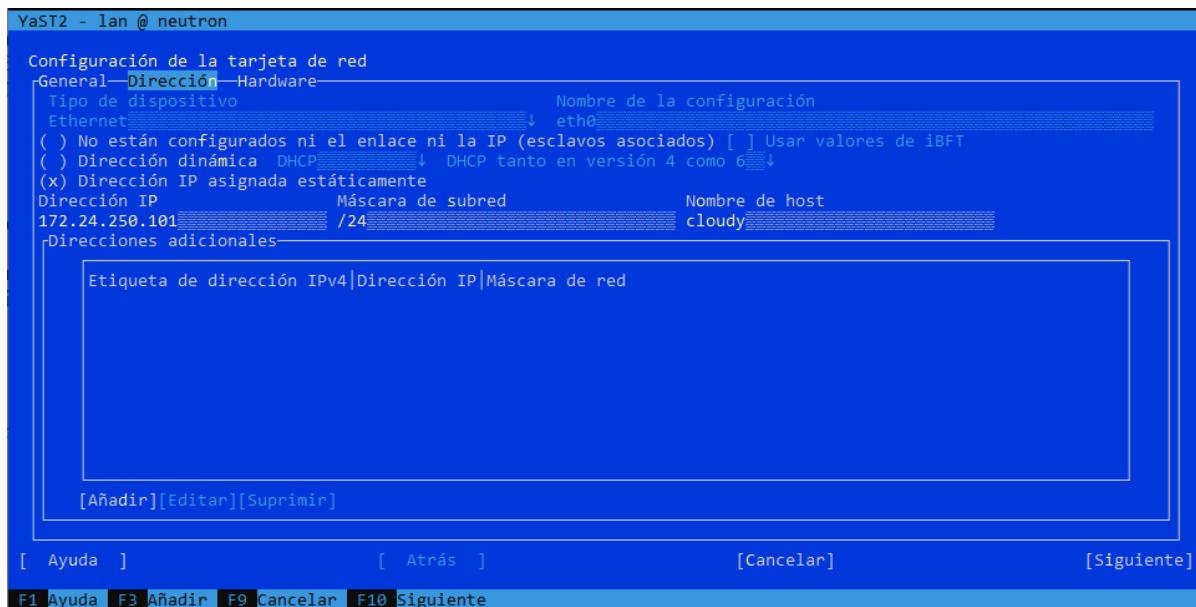


Figura 19 - Captura de configuración de red mediante Yast. Fuente propia.

Verificación de conectividad

```
yondu:~ # ping compute
PING rocket (172.24.250.102) 56(84) bytes of data.
64 bytes from rocket (172.24.250.102): icmp_seq=1 ttl=64 time=0.213 ms
64 bytes from rocket (172.24.250.102): icmp_seq=2 ttl=64 time=0.235 ms
^C
--- rocket ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
rtt min/avg/max/mdev = 0.213/0.224/0.235/0.011 ms
```

Figura 20 - Captura de verificación de conectividad. Fuente propia.

### 9.1.6. Network Time Protocol (NTP)

Configuramos el nodo de Control:

```
zypper install chrony
```

Editamos el archivo /etc/chrony.conf y agregamos los siguientes permisos para clientes NTP:

```
allow 172.24.250.102/24
```

Reiniciamos el servicio para surtir los cambios:

```
systemctl enable chronyd.service
systemctl start chronyd.service
```

Configuramos el nodo de Cómputo:

Modificamos el archivo para que el servidor NTP sea el servidor de control

```
server 172.24.250.100 iburst
```

Verificación de conectividad

```
yondu:/home/elvis # chronyc sources
210 Number of sources = 5
MS Name/IP address          Stratum Poll Reach LastRx Last sample
=====
===
^* 200.27.106.116           1   10    377    111    +132us[ +150us] +/- 656
4us
^- time.cloudflare.com     3   10    377    130    -21us[-2977ns] +/-   6
8ms
^- 200-89-75-197-LIBRE.uchi> 2   10    377    324    -497us[ -480us] +/-   4
0ms
^- 200-89-75-198-LIBRE.uchi> 2   10    377    913    -1631us[-1614us] +/-   3
2ms
^- time.cloudflare.com     3   10    377    118    -1111us[-1094us] +/-   7
0ms
```

Figura 21 - Captura de verificación de funcionamiento de NTP en nodo controlador. Fuente Propia.

```
compute:/home/elvis # chronyc sources
210 Number of sources = 5
MS Name/IP address          Stratum Poll Reach LastRx Last sample
=====
===
^* yondu                   2    7    377     53    -6768ns[-9225ns] +/- 654
5us
^- time.cloudflare.com     3   10    377     72    -2335us[-2337us] +/-   6
9ms
^- 200-89-75-197-LIBRE.uchi> 2   10    377    718    +685us[ +699us] +/-   3
5ms
^- time.cloudflare.com     3   10    377    586    +130us[ +130us] +/-   6
8ms
^- 200-89-75-198-LIBRE.uchi> 2   10    377    709    -928us[ -915us] +/-   3
7ms
```

Figura 22 - Captura de verificación del funcionamiento de NTP en nodo de cómputo. Fuente Propia.

### 9.1.7. Paquetes de OpenStack

Los paquetes de software donde vienen todos los servicios del sistema OpenStack deben ser configurados correctamente para obtener la versión Stein que es el lanzamiento número 19 que fue el último estable cuando se comenzó este proyecto. Realizamos esta configuración en los tres nodos con OpenSuse Leap 15.

```
zypper addrepo -f obs://Cloud:OpenStack:Stein/openSUSE_Leap_15.0 Stein
```

Podemos verificar la integridad de los paquetes con el siguiente certificado GPG

```
Key Name: Cloud:OpenStack OBS Project <Cloud:OpenStack@build.opensuse.org>
Key Fingerprint: 35B34E18 ABC1076D 66D5A86B 893A90DA D85F9316
Key Created: 2015-12-16T16:48:37 CET
Key Expires: 2018-02-23T16:48:37 CET
```

Para finalizar la instalación

```
zypper refresh && zypper dist-upgrade
```

Instalamos el cliente OpenStack

```
zypper install python-openstackclient
```

### 9.1.8. Base de datos SQL

“La mayoría de los servicios de OpenStack utilizan bases de datos para almacenar información. La base de datos usualmente se ejecuta en el nodo controlador” [13] (OpenStack Documentation, 2021). Para este caso en concreto se utilizó MariaDB como gestor de base de datos. Pero también se pueden utilizar otras bases de datos como MySQL y PostgreSQL.

```
zypper install mariadb-client mariadb python-PyMySQL
```

Se creó el archivo de configuración en /etc/my.cnf.d/openstack.cnf y se configuró la base de datos para que sea accesible en la red de administración por los demás nodos.

```
[mysqld]
bind-address = 172.24.250.100

default-storage-engine = innodb
innodb_file_per_table = on
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
```

Se habilitó y se inició el servicio de bases de datos:

```
systemctl enable mysql.service
systemctl start mysql.service
```

Se habilitó la seguridad básica con

```
mysql_secure_installation
```

### 9.1.9. Cola de mensajes

La función de la cola de mensajes es según el glosario de la guía “pasar peticiones desde clientes a los trabajadores apropiados y devolver la salida al cliente después de que esté completo el trabajo.” [13] (OpenStack Documentation, 2021).

Openstack usa estas colas de mensajes para “coordinar operaciones y el estado de información entre servicios. El servicio de cola de mensajes usualmente se ejecuta en el nodo controlador.” [13] (OpenStack Documentation, 2021).

Para la guía y este proyecto se utilizó el agente de mensajes RabbitMQ, según su documentación oficial “RabbitMQ es el agente de mensajes de código libre más desplegado. RabbitMQ es liviano y fácil de desplegar en instalaciones y en la nube. Soporta múltiples protocolos de mensajes. Puede ser desplegado en configuraciones distribuidas y federadas para acercarse a los requerimientos alta-escalabilidad y alta-disponibilidad.”[14] (RabbitMQ, 2021).

Instalamos los paquetes en el nodo controlador:

```
zypper install rabbitmq-server
```

Iniciamos el servicio de cola de mensajes y configuramos para que inicie cuando el sistema encienda:

```
systemctl enable rabbitmq-server.service
systemctl start rabbitmq-server.service
```

Crear el usuario openstack:

```
rabbitmqctl add_user openstack pez1988stack
```

Dar permisos de escritura, lectura y acceso para el usuario openstack:

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

### 9.1.10. Memcached

En la página oficial Memcached se define como “Sistema de memoria caché de objetos distribuida, libre y de código abierto. (...). Memcached es un almacenamiento en memoria para llave-valor (arreglos, objetos) desde resultados de llamadas de la base de datos, llamadas a las API, o renderizado de páginas.” [15] (Memcached, 2021).

El servicio de identidad (Keystone) utiliza servicios de mecanismo autentificación como Memcached para almacenar tokens. Este servicio se ejecuta en el nodo controlador.

Instalamos los paquetes:

```
zypper install memcached python-pyton-memcached
```

Editamos en /etc/sysconfig/memcached y configuramos el servicio para usar la IP de administración del nodo controlador. Esto permitirá a los otros nodos conectarse:

```
MEMCACHED_PARAMS="-l 172.24.250.100"
```

Habilitamos y reiniciamos el servicio:

```
systemctl enable memcached.service
systemctl start memcached.service
```

### 9.1.11. Etcd

“Los servicios de OpenStack pueden utilizar Etcd, un almacenamiento confiable y distribuido de llave-valor para bloques de llave, almacenamiento de configuración, mantener monitoreo de los servicios activos y otros datos útiles.” [13] (OpenStack Documentation, 2021).

Instalamos etcd, para ello creamos el usuario:

```
groupadd --system etcd
useradd --home-dir "/var/lib/etcd" \
--system \
--shell /bin/false \
-g etcd \
etcd
```

Creamos los directorios necesarios:

```
mkdir -p /etc/etcd
chown etcd:etcd /etc/etcd
mkdir -p /var/lib/etcd
chown etcd:etcd /var/lib/etcd
```

Descargamos e instalamos los comprimidos:

```
ETCD_VER=v3.2.7

rm -rf /tmp/etcd && mkdir -p /tmp/etcd

curl -L \
https://github.com/coreos/etcd/releases/download/${ETCD_VER}/etcd-${ETCD_VER}-linux-
amd64.tar.gz \
-o /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz
tar xzvf /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz \
-C /tmp/etcd --strip-components=1

cp /tmp/etcd/etcd /usr/bin/etcd
cp /tmp/etcd/etcdctl /usr/bin/etcdctl
```

Creamos y editamos en /etc/etcd/etcd.conf.yaml para definir las siguientes configuraciones:

```
name: controller
data-dir: /var/lib/etcd
initial-cluster-state: 'new'
initial-cluster-token: 'etcd-cluster-01'
initial-cluster: controller=http://172.24.250.100:2380
initial-advertise-peer-urls: http://172.24.250.100:2380
advertise-client-urls: http://172.24.250.100:2379
listen-peer-urls: http://0.0.0.0:2380
listen-client-urls: http://172.24.250.100:2379
```

Creamos y editamos en /usr/lib/systemd/system/etcd.service:

```
[Unit]
After=network.target
Description=etcd - highly-available key value store

[Service]
# Uncomment this on ARM64.
```

```
# Environment="ETCD_UNSUPPORTED_ARCH=arm64"
LimitNOFILE=65536
Restart=on-failure
Type=notify
ExecStart=/usr/bin/etcd --config-file /etc/etcd/etcd.conf.yml
User=etcd

[Install]
WantedBy=multi-user.target
```

Reiniciamos el servicio de archivos de systemd y habilitamos e iniciamos el servicio:

```
systemctl daemon-reload
systemctl enable etcd
systemctl start etcd
```

Con esto hemos finalizado la configuración del ambiente para instalar OpenStack, todos estos servicios serán cruciales para su funcionamiento. En el siguiente punto se procederá a trabajar con el software de OpenStack en concreto.

## 9.2. Desplegar IaaS mediante OpenStack

El despliegue del software de OpenStack se realiza mediante la instalación independiente de servicios. La interrelación que existe entre ellos se irá entendiendo a medida de que estos sean expuestos, especialmente los casos de Keystone, Nova y Neutron. Estos son los servicios que más se interrelacionan y los cuales también requieren más configuración para su correcto funcionamiento.

Como se explicó en la metodología el proceso de trabajo es seguir la documentación oficial de forma que esta se vaya completando correctamente. Adicionalmente se agregara información relacionada con la guía para ir completando, esta proviene de diferentes fuentes en internet, las cuales han sido probadas y aceptadas por su utilidad en la implementación.

### 9.2.1. Keystone: Identity service

“El sistema de OpenStack consiste en varios servicios clave que son instalados separadamente. Estos servicios trabajan en conjunto dependiendo de las necesidades de la nube(...). Esta sección describe cómo instalar y configurar el servicio de identidad de OpenStack llamado Keystone en el nodo controlador. Por propósitos de escalabilidad, esta configuración despliega tokens Fernet y el servidor HTTP Apache para manejar peticiones.” [13] (OpenStack Documentation, 2021).

“El servicio de identidad provee un único punto de integración para administrar la autenticación, autorización y un catálogo de servicios.

El servicio de identidad es típicamente el primer servicio con el que el usuario interactúa. Una vez autenticado un usuario final puede usar su identidad para acceder a otros servicios de OpenStack. Igualmente, otros servicios de OpenStack aprovechan el servicio de identidad para asegurar quienes son los usuarios y que harán con los otros servicios. Este servicio también se puede integrar con otros métodos de autenticación externos como LDAP.

Usuarios y servicios pueden ubicar otros servicios usando el catálogo de servicios, el cual es manejado por el servicio de identidad. (...). Cada servicio puede tener uno o varios puntos finales y cada uno de los puntos finales puede ser de tres tipos: administración, interna y pública. (...). OpenStack soporta múltiples regiones para escalabilidad. Por simplicidad, esta guía usa la red de administración para todos los puntos finales y la región por defecto será RegionOne. El servidor de identidad contiene estos tres componentes:

**Servidor:** Un servidor centralizado provee servicios autenticación y autorización usando una interfaz RESTful.

**Drivers:** Los driver o el servicio back-end están integrados al servidor centralizado. Estos son usados para acceder a la información de identidad en repositorios externos a OpenStack, y pueden existir en la infraestructura donde OpenStack es desplegado (por ejemplo una base de datos SQL o servidores LDAP).

**Modulos:** Modulos middleware se ejecutan en el espacio de dirección del componente de OpenStack que es usado por el servicio de identidad. Estos módulos interceptan peticiones de servicios, extraen credenciales de usuario, y envían al servidor centralizado para la autorización. Esta integración entre los módulos middleware y los componentes de OpenStack usan el Python Web Gateway Interface.” [13] (OpenStack Documentation, 2021).

Prerrequisitos:

Crear la base de datos del servicio de identidad:

```
mysql -u root -p
```

```
MariaDB [(none)]> CREATE DATABASE keystone;
```

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
IDENTIFIED BY 'pez1988stack';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
IDENTIFIED BY 'pez1988stack';
```

Estos pasos serán necesarios para cada uno de los servicios de OpenStack y serán configuraciones que se irán repitiendo constantemente en la base de datos.

Instalamos los paquetes de OpenStack Keystone, Apache y el módulo WSGI mediante zypper:

```
zypper install openstack-keystone apache2 apache2-mod_wsgi
```

Editamos el archivo de configuración en `/etc/keystone/keystone.conf` y realizamos las siguientes configuraciones:

```
[database]
connection = mysql+pymysql://keystone:pez1988stack@control/keystone

[token]
provider = fernet
```

Cargamos de datos la base de datos del servicio de identidad con los scripts por defecto:

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

Inicializamos el repositorio de claves Fernet:

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

Arrancamos el servicio de identidad con los scripts incluidos:

```
keystone-manage bootstrap --bootstrap-password pez1988stack \
--bootstrap-admin-url http://control:5000/v3/ \
--bootstrap-internal-url http://control:5000/v3/ \
--bootstrap-public-url http://control:5000/v3/ \
--bootstrap-region-id RegionOne
```

Configuramos el servidor HTTP Apache:

Editamos en `/etc/sysconfig/apache2`:

```
APACHE_SERVERNAME="control"
```

Creamos el archivo `/etc/apache2/conf.d/wsgi-keystone.conf` con la siguiente configuración, que conectará el servicio HTTP con el WSGI:

```

Listen 5000
Listen 35357
<VirtualHost *:5000>
    WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone
    group=keystone display-name=%{GROUP}
        WSGIProcessGroup keystone-public
        WSGIScriptAlias / /usr/bin/keystone-wsgi-public
        WSGIApplicationGroup %{GLOBAL}
        WSGIPassAuthorization On
        ErrorLogFormat "%{cu}t %M"
        ErrorLog /var/log/apache2/keystone.log
        CustomLog /var/log/apache2/keystone_access.log combined

    <Directory /usr/bin>
        Require all granted
    </Directory>
</VirtualHost>

#agregamos el servicio en el puerto 35357 por problema presentados en la
compatibilidad
<VirtualHost *:35357>
    WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone
    group=keystone display-name=%{GROUP}
        WSGIProcessGroup keystone-admin
        WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
        WSGIApplicationGroup %{GLOBAL}
        WSGIPassAuthorization On
        ErrorLogFormat "%{cu}t %M"
        ErrorLog /var/log/apache2/keystone.log
        CustomLog /var/log/apache2/keystone_access.log combined

    <Directory /usr/bin>
        Require all granted
    </Directory>
</VirtualHost>

```

Cambiamos los permisos de posesión al usuario keystone recursivamente:

```
chown -R keystone:keystone /etc/keystone
```

Finalizamos la instalación. Iniciamos el servidor HTTP Apache y lo configuramos para que inicie cuando el sistema inicie:

```
systemctl enable apache2.service
systemctl start apache2.service
```

Configuramos una cuenta administrativa en un archivo de texto llamado `/root/admin-openrc`. Esta nos permitirá autenticarse y acceder rápidamente al cliente de comandos OpenStack:

```
export OS_USERNAME=admin
export OS_PASSWORD=pez1988stack
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://control:5000/v3
export OS_IDENTITY_API_VERSION=3
```

Creamos un dominio, proyectos, usuarios y roles: El servicio de identidad provee autentificación para cada servicio de OpenStack. El servicio de autentificación usa una combinación de dominios, proyectos, usuarios y roles.

Creamos un proyecto ‘admin’ que contenga un único usuario para cada servicio que se agregue al ambiente. Creamos el proyecto ‘admin’:

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

Para las funciones no administrativas deberíamos usar un proyecto y un usuario sin privilegios. En esta caso usaremos ‘myproject’ y ‘myuser’:

```
openstack project create --domain default \
--description "Demo Project" myproject
```

Creamos el usuario ‘myuser’:

```
openstack user create --domain default \
--password-prompt myuser
```

Creamos el rol ‘myrole’:

```
openstack role create myrole
```

Unimos estas combinaciones:

```
openstack role add --project myproject --user myuser myrole
```

Verificamos la operación después de haber configurado. Esto es vital para probar el correcto funcionamiento del servicio de identidad. Para ello quitamos temporalmente las variables de entorno OS\_AUTH\_URL y OS\_PASSWORD:

```
unset OS_AUTH_URL OS_PASSWORD
```

Como el usuario admin, solicitamos un token de autenticación:

```
openstack --os-auth-url http://control:5000/v3 \
--os-project-domain-name Default --os-user-domain-name Default \
--os-project-name admin --os-username admin token issue
```

Si la configuración es correcta debería devolvernos la siguiente información:

Field	Value
expires	2021-09-27T20:05:12+0000
id	gAAAAABhUhXoilFzSfGzU81--oTbn1xDEEmk_MSbdiTvu8YqqnHoycaI210kT UeAImIGrUWkiCLqBMZThk1I9SI8X7TW-DI2ZVtRHYYb4Jnrv2vnZKdyTYbzxfcDy84dZgil_ESb8 Zw8nrA096PoXiIst0db1u-N367SGB8BwCCpkhXpI316Q7Q
project_id	5adf803a38ba4c5590879a8545fcf5d1
user_id	b30621dd1d604530b37fe29f80968aec

Figura 23 - Captura de pantalla de verificación del funcionamiento de Keystone. Fuente propia.

Como paso final también podemos probar el script `admin-openrc`. En este podemos cargar todas las variables de entorno descritas anteriormente, para poder utilizarlas simplemente hay que ejecutar:

```
. admin-openrc
```

Y luego solicitamos el token de autenticación con:

```
openstack token issue
```

El resultado debería responder con la siguiente información:

```

yondu:~ # . admin-openrc
yondu:~ # openstack token issue
+-----+
| Field      | Value
+-----+
| expires    | 2021-09-27T20:07:07+0000
| id         | gAAAAABhUhZbsaX3LUtH4EdZ4qPuDEQtM1CVvx28dTj0oO6iBM8e-inphmYZ
0jSqycu1BiDBFZ7K9ZZ9zyxF8B7BdX1wsSteerAezf5eLo2uYx_qH7sfUaLUL5WdYr_EVaesPGsr
xbR_jYxbOMEYQGXENQqHhInR1kuywY-WKD13I59KDiwi84 |
| project_id | 5adf803a38ba4c5590879a8545fcf5d1
| user_id    | b30621dd1d604530b37fe29f80968aec
+-----+

```

Figura 24 - Captura de pantalla verificación del funcionamiento de Keystone. Fuente propia.

### 9.2.2. Glance: Image service

“El servicio de imagen (glance) permite al usuario descubrir, registrar y obtener imágenes de máquinas virtuales. Este ofrece un API REST que te permite consultar metadatos de imágenes de máquinas virtuales y obtener una imagen actual. Puedes almacenar imágenes de máquinas virtuales disponibles a través del servicio de imágenes en diferentes ubicaciones, desde sistema de archivos simples hasta sistemas de almacenamiento como el Object Storage de OpenStack.

El servicio de imagen de OpenStack es central a la Infraestructura como Servicio. Este acepta peticiones de API para discos o imágenes de servidores y definiciones de metadatos desde los usuarios finales del componente de Openstack Compute (Nova). (...)

El servicio de imagen de OpenStack incluye los siguientes componentes:

Glance-api: Acepta llamadas de API de imagen para descubrir, obtener y almacenar.

Glance-registry: Almacena, procesa y obtiene metadatos de las imágenes. Los metadatos incluyen ítems como el tamaño y el tipo.

Bases de datos: Almacena metadatos de las imágenes, estas pueden estar en la base de datos que se desee, usualmente MySQL o SQLite.

Repositorio y almacenamiento para los archivos de imagen: Varios tipos de repositorio están soportados incluyendo sistemas de archivos normales. Object Storage, RADOS block devices, VMware datastore y HTTP son otras posibilidades.

Servicio de definición de metadatos: Una API común para los proveedores, administradores, servicios y usuarios para definir sus propios metadatos.” [13] (OpenStack Documentation, 2021).

Prerrequisitos:

Crear la base de datos del servicio de imagen:

```
mysql -u root -p
```

```
MariaDB [(none)]> CREATE DATABASE glance;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
    IDENTIFIED BY 'pez1988stack';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
    IDENTIFIED BY 'pez1988stack';
```

Obtenemos las variables de entorno en el archivo admin-openrc:

```
. admin-openrc
```

Creamos el usuario glance con las credenciales:

```
openstack user create --domain default --password-prompt glance
```

Agregamos el rol de admin al usuario glance y al proyecto servicio:

```
openstack role add --project service --user glance admin
```

Creamos la entidad del servicio glance:

```
openstack service create --name glance \
    --description "OpenStack Image" image
```

Creamos los puntos finales del API del servicio de imagen:

```
openstack endpoint create --region RegionOne \
    image public http://control:9292
```

```
openstack endpoint create --region RegionOne \
    image internal http://control:9292
```

```
openstack endpoint create --region RegionOne \
    image admin http://control:9292
```

Instalamos los paquetes:

```
zypper install openstack-glance openstack-glance-api openstack-glance-registry
```

Editamos el archivo `/etc/glance/glance-api.conf` y configuramos la base de datos:

```
[database]
connection = mysql+pymysql://glance:pez1988stack@controller/glance
```

Configuramos la conexión a keystone en `[keystone_auth_token]` y en `[paste_deploy]`:

```
[keystone_auth_token]
www_authenticate_uri = http://control:5000
auth_url = http://control:5000
memcached_servers = control:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = pez1988stack

[paste_deploy]
# ...
flavor = keystone
```

En las sección `[glance_storage]` configuramos el sistema de archivos local y la ubicación de archivos de imagen:

```
[glance_store]
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
```

Editamos `/etc/glance/glance-registry.conf` y configurnos:

```
[database]
connection = mysql+pymysql://glance:pez1988stack@control/glance
```

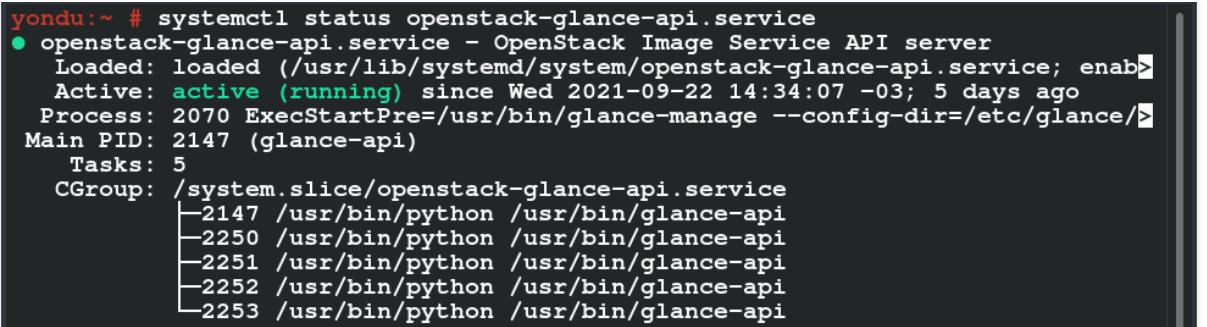
En las secciones de [keystone\_authhtoken] y [paste\_deploy] configuramos el servicio de identidad:

```
[keystone_authhtoken]
# ...
www_authenticate_uri = http://control:5000
auth_url = http://control:5000
memcached_servers = control:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = admin
username = glance
password = pez1988stack

[paste_deploy]
flavor = keystone
```

Finalizamos la instalación:

```
systemctl enable openstack-glance-api.service openstack-glance-registry.service
systemctl start openstack-glance-api.service openstack-glance-registry.service
```



```
yondu:~ # systemctl status openstack-glance-api.service
● openstack-glance-api.service - OpenStack Image Service API server
   Loaded: loaded (/usr/lib/systemd/system/openstack-glance-api.service; enabled; relo
   Active: active (running) since Wed 2021-09-22 14:34:07 -03; 5 days ago
     Process: 2070 ExecStartPre=/usr/bin/glance-manage --config-dir=/etc/glance/ >
    Main PID: 2147 (glance-api)
      Tasks: 5
     CGroup: /system.slice/openstack-glance-api.service
             └─2147 /usr/bin/python /usr/bin/glance-api

systemctl status openstack-glance-api.service
```

Figura 25 - Captura de pantalla comprobación del funcionamiento de Glance. Fuente propia.

### 9.2.3. Placement service

“El servicio de colocación provee un API HTTP para hacer seguimiento de los recursos del proveedor, inventarios y usos. Placement opera como un servicio web sobre un modelo de datos. La instalación involucra crear las bases de datos necesarias e instalar y configurar el servicio web. Esto es un proceso sencillo, pero requiere unos pocos pasos para integrar con el resto de la nube OpenStack.” [13] (OpenStack Documentation, 2021).

Prerrequisitos: Antes de instalar el servicio placement, debes crear una base de datos con las credenciales del servicio y las API endpoints.

Creamos la base de datos:

```
mysql -u root -p

MariaDB [(none)]> CREATE DATABASE placement;

MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'localhost' \
    IDENTIFIED BY 'pez1988stack';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'%' \
    IDENTIFIED BY 'pez1988stack';
```

Configuramos los usuarios y los endpoints:

Obtenemos las variables de entorno en el archivo admin-openrc:

```
. admin-openrc
```

Creamos el usuario del servicio placement:

```
openstack user create --domain default --password-prompt placement
```

Agregamos el usuario placement al proyecto servicio con rol de administrador:

```
openstack role add --project service --user placement admin
```

Creamos la entrada del placement API en servicio de catálogo:

```
openstack service create --name placement --description "Placement API" placement
```

Creamos los endpoints de la API:

```
openstack endpoint create --region RegionOne \
placement public http://control:8778

openstack endpoint create --region RegionOne \
placement internal http://control:8778

openstack endpoint create --region RegionOne \
placement admin http://control:8778
```

Instalamos y configuramos los componentes:

Instalamos los paquetes:

```
zypper install openstack-placement-api
```

Editamos en /etc/placement/placement.conf y configuramos:

Configuramos [placement\_database] el acceso a la base de datos:

```
[placement_database]

connection = mysql+pymysql://placement:pez1988stack@control/placement
```

En la sección [api] y [keystone\_auth\_token] configuramos el servicio de identidad:

```
[api]

auth_strategy = keystone

[keystone_auth_token]

auth_url = http://control:5000/v3
memcached_servers = control:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = admin
username = placement
password = pez1988stack
```

Llenamos la base de datos placement con el script incluido en el paquete:

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

Finalizamos la instalación y configuramos el vhost del API apache:

```
mv /etc/apache2/vhosts.d/placement-api.conf.sample \
/etc/apache2/vhosts.d/placement-api.conf
```

```
systemctl reload apache2.service
```

Verificamos la instalación:

Obtenemos las variables de entorno en el archivo admin-openrc:

```
. admin-openrc
```

Hacemos un chequeo de estado para revisar:

```
placement-status upgrade check
```

```
yondu:~ # placement-status upgrade check
+-----+
| Upgrade Check Results
+-----+
| Check: Missing Root Provider IDs
| Result: Success
| Details: None
+-----+
| Check: Incomplete Consumers
| Result: Success
| Details: None
+-----+
```

Figura 26 - Captura de pantalla de la verificación del funcionamiento del servicio Placement. Fuente propia.

#### 9.2.4. Neutron Networking service

En la documentación oficial de OpenStack la instalación de Neutron es posterior a la de Nova. Sin embargo Nova al necesitar también de Neutron para funcionar adecuadamente. Por ello en este documento recomiendo comenzar con Neutron y así también de paso simplificar la instalación de OpenStack reduciendo el número de pasos a realizar significativamente, ya que Neutron y Nova son los servicios que más configuración requieren y los que más tiempo toman.

“OpenStack Networking (Neutron) te permite crear y añadir interfaces a red manejadas por otros servicios de OpenStack. Los plugins pueden ser implementados para acomodar diferente equipamiento de red y software, proveyendo flexibilidad a la arquitectura y al despliegue de OpenStack.

Incluye los siguientes componentes:

Neutron-server: Acepta y enruta las peticiones de API a los plugins de networking adecuados, para la acción.

OpenStack Networking plugins y agentes: Conecta y desconecta puertos, crea redes o subredes, y provee direccionamiento IP. Estos plugins y agentes difieren dependiendo de las marcas y tecnologías dependiendo de la nube. OpenStack Networking viene con los plugins y agentes para switches físicos y virtuales Cisco, productos NEC OpenFlow, Open vSwitch, Linux bridging y el producto NSX de VMware. Los agentes comunes para todas las instalaciones de OpenStack son L3 (Capa 3), DHCP (direcciónamiento de IP dinámico) y un agente de plugins.

Cola de mensajes: Usado por la mayoría de las instalaciones de OpenStack Networking para enrutar información entre el neutron-server y varios agentes. También actúa como base de datos para almacenar información entre el estado de la red y plugins particulares.

OpenStack Networking interactúa principalmente entre OpenStack Compute y provee de redes y conectividad para las instancias.” [13] (OpenStack Documentation, 2021).

También para entender más a fondo a Neutron es necesario revisar sus conceptos teóricos. La forma en la cual se configura está muy relacionada a la infraestructura de red que se le da la IaaS. Por ello es necesario manejar los conceptos de OpenStack para adecuar correctamente la infraestructura a las necesidades finales de la nube.

“OpenStack Networking (Neutron) administra todas las facetas de interconexión de redes de la Virtual Networking Infrastructure (VNI) y las capas de acceso respecto a la Physical Networking Infrastructure (PNI) en el ambiente de OpenStack. Neutron permite a los proyectos crear topologías de redes virtuales avanzadas, las cuales pueden incluir servicios como firewall, balanceadores de carga y redes virtuales privadas (VPN).

Neutron provee redes, subredes y routers como abstracciones de objetos. Cada abstracción tiene su funcionalidad que imita su contraparte física: las redes contienen subredes, y los routers enrutan el tráfico entre diferentes redes y subredes.

Cualquier setup de Neutron debe tener al menos una red externa. A diferencia de otras redes, la red externa no es simplemente una red definida virtualmente. En lugar de eso representa la vista dentro de la red externa accesible desde fuera de la instalación de OpenStack. Las direcciones IP de las redes externas son accesibles por cualquiera físicamente fuera de la red.

En adición a las redes externas, cualquier setup de Neutron tiene una o más redes internas, estas redes definidas por software se conectan directamente a las máquinas virtuales. Solo las VMs en cualquier red interna, o aquellas en las subredes conectadas a través de interfaces al mismo router pueden acceder a las redes de esas máquinas virtuales directamente.

Para acceder desde el exterior a las VMs y viceversa, los routers entre estas redes son necesarios. Cada router tiene un gateway que se conecta a una red externa y uno o más interfaces conectadas a las redes internas. Como un router físico, las subredes pueden acceder máquinas u otras subredes que están conectadas al mismo router y las máquinas pueden acceder desde redes exteriores a través del gateway para el router.

Adicionalmente se pueden utilizar direcciones IP de redes externas a puertos en la red interna. Donde sea que algo esté conectado a una subred, esta conexión es llamada un puerto. Se pueden asociar direcciones IP de direcciones externas a puertos de las VMs. De esta forma una entidad fuera de la red puede acceder a las VMs.

Neutron también soporta los grupos de seguridad. Esto permite al administrador definir las reglas de firewall en grupos. Una VM puede pertenecer a uno o más grupos de seguridad, y Neutron aplica esa reglas en aquellos grupos de seguridad para bloquear desbloquear puertos, rango de puertos o tipos de tráfico a las VM.

Cada plugin que Neutron usa tiene sus propios conceptos. Mientras que no son vitales para operar la VNI y el ambiente de OpenStack, entender estos conceptos pueden ayudarte a levantar un setup de Neutron. Todas las instalaciones de Neutron usan un plugin principal y un plugin de grupos de seguridad. Adicionalmente los plugins de Firewall as a Service (FWaaS) y el Load Balancer as a Service (LBaaS) están disponibles.” [13] (OpenStack Documentation, 2021).

Para empezar la configuración será necesario que el ambiente de red visto en la sección 4.1.5 esté operacional correctamente. Luego pasaremos a realizar la siguiente configuración:

**Prerrequisitos:** Antes de instalar el servicio placement, debes crear una base de datos con las credenciales del servicio y las API endpoints.

Creamos la base de datos:

```
mysql -u root -p

MariaDB [(none)]> CREATE DATABASE neutron;

MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
    IDENTIFIED BY 'pez1988stack';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
    IDENTIFIED BY 'pez1988stack';
```

Configuramos los usuarios y los endpoints:

Obtenemos las variables de entorno en el archivo admin-openrc:

```
. admin-openrc
```

Creamos el usuario del servicio neutron:

```
openstack user create --domain default --password-prompt neutron
```

Agregamos el usuario neutron al proyecto servicio con rol de administrador:

```
openstack role add --project service --user neutron admin
```

Creamos la entrada de neutron servicio de catálogo:

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

Creamos los endpoints de la API:

```
openstack endpoint create --region RegionOne \
    network public http://control:9696
```

```
openstack endpoint create --region RegionOne \
    network internal http://control:9696
```

```
openstack endpoint create --region RegionOne \
    network admin http://control:9696
```

Realizaremos la instalación de opciones de red 2, que abarca la opción 1.

Configuramos el nodo controlador en primer lugar.

Instalamos los paquetes:

```
zypper install --no-recommends openstack-neutron \
  openstack-neutron-server openstack-neutron-linuxbridge-agent \
  openstack-neutron-l3-agent openstack-neutron-dhcp-agent \
  openstack-neutron-metadata-agent bridge-utils
```

Configuramos los componentes del servidor:

```
[DEFAULT]
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = true
transport_url = rabbit://openstack:pez1988stack@control
auth_strategy = keystone

notify_nova_on_port_status_changes = true
notify_nova_on_port_data_changes = true

[database]
connection = mysql+pymysql://neutron:pez1988stack@control/neutron

[keystone_authhtoken]

www_authenticate_uri = http://control:5000
auth_url = http://control:5000
auth_uri = http://control:5000
memcached_servers = control:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = admin
username = neutron
password = pez1988stack

[nova]

auth_url = http://control:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = admin
username = nova
password = pez1988stack
```

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

Configuramos el Modular Layer 2 plug-in (ML2)

“El plugin ML2 usa el mecanismo de Linux Bridge para construir infraestructura de redes virtuales para las instancias.” [13] (OpenStack Documentation, 2021).

Editamos el archivo `/etc/neutron/plugins/ml2/ml2_conf.ini`:

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

[ml2_type_flat]
flat_networks = provider

[ml2_type_vxlan]
vni_ranges = 1:1000

[securitygroup]
# ...
enable_ipset = true
```

Procedemos a configurar Linux Bridge Agent:

“El agente Linux Bridge construye en la capa 2, infraestructura de redes virtuales para las instancias y maneja grupos de seguridad.” [13] (OpenStack Documentation, 2021).

Este agente permite interconectar diferentes nodos de la implementación, en especial conecta las redes virtuales entre el nodo controlador y el nodo de cómputo.

Editamos la configuración en `/etc/neutron/plugins/ml2/linuxbridge_agent.ini`:

```
[linux_bridge]
physical_interface_mappings = provider:eth0

[vxlan]
enable_vxlan = true
local_ip = 192.168.4.47
l2_population = true

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

Nos aseguramos de que para utilizar este agente tengamos el núcleo Linux preparado para soportar la capacidad de bridging. Esto mediante los valores `sysctl`.

Si no estuviesen disponible deberíamos asegurarnos de cargarlos en el núcleo esto mediante el módulo de kernel `br_netfilter`. Podemos utilizar el comando `modprobe` o `insmod` para realizar esta tarea.

```
yondu:/home/elvis # modprobe br_netfilter
yondu:/home/elvis # sysctl -a | grep 'net.bridge.*'
net.bridge.bridge-nf-call-arptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
```

Figura 27 - Captura de pantalla de comprobación del módulo del kernel `br_netfilter`. Fuente Propia

Ahora procedemos a configurar agente de Capa 3 (L3 Agent). “El agente de capa 3 provee enrutamiento y servicios de NAT para redes virtuales de autoservicio”. [13] (OpenStack Documentation, 2021).

Editamos `/etc/neutron/l3_agent.ini` y completamos con las siguientes acciones:

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

Configuramos el agente DHCP que provee de direcciones IP para las redes virtuales. Esto en `/etc/neutron/dhcp_agent.ini`:

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

Ahora procedemos a configurar el nodo de cómputo:

Instalamos los paquetes necesarios.

```
zypper install --no-recommends openstack-neutron-linuxbridge-agent bridge-utils
```

Editamos la configuración de Neutron para el nodo de cómputo en /etc/neutron/neutron.conf.

```
[DEFAULT]
transport_url = rabbit://openstack:pez1988stack@control
auth_strategy = keystone

[keystone_authtoken]
www_authenticate_uri = http://control:5000
auth_url = http://control:5000
memcached_servers = control:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = admin
username = neutron
password = pez1988stack

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

Configuramos el Linux Bridge Agent que es el agente en común esto en /etc/neutron/plugins/ml2/linuxbridge\_agent.ini:

```
[linux_bridge]
physical_interface_mappings = provider:eth0

[vxlan]
enable_vxlan = true
local_ip = 192.168.4.49
l2_population = true

[securitygroup]
# ...
```

```
enable_security_group = true
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

Configuramos el servicio de computo para que utilice el servicio de redes.

Editamos en /etc/nova/nova.conf:

```
[neutron]
url = http://control:9696
auth_url = http://control:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = admin
username = neutron
password = pez1988stack
service_metadata_proxy = true
metadata_proxy_shared_secret = pez1988stack
```

Finalizamos la instalación, para ello configuramos una variable de entorno necesaria para el funcionamiento de neutrón. Para ello agregamos la siguiente línea en /etc/sysconfig/neutron:

```
NEUTRON_PLUGIN_CONF="/etc/neutron/plugins/ml2/ml2_conf.ini"
```

Reiniciamos los servicios de OpenStack involucrados en el nodo de cómputo para surtir los cambios:

```
systemctl restart openstack-nova-compute.service
systemctl enable openstack-neutron-linuxbridge-agent.service
systemctl start openstack-neutron-linuxbridge-agent.service
```

Verificamos el correcto funcionamiento de los dos nodos con el comando que nos lista los agentes de red disponibles y su estado. En el se puede ver el Linux Bridge Agent del nodo de cómputo funcionando adecuadamente, así como los agentes del nodo controlador:

```
openstack network agent list
```

ID	Agent Type	Host	Availability Zone	Alive	State	Binary
18fba23e-fabd-45ea-afb7-8b7f56dc6069	Linux bridge agent	yondu	None	:-)	UP	neutron-linuxbridge-agent
43ce3f81-33a0-4f13-8484-e487012a979c	Metadata agent	yondu	None	:-)	UP	neutron-metadata-agent
4a69f55c-1bff-4ff2-8b7d-f48431516821	L3 agent	yondu	nova	:-)	UP	neutron-l3-agent
87bbc5da-914f-4ee2-842e-d3d8ec66579d	Linux bridge agent	compute	None	:-)	UP	neutron-linuxbridge-agent
d695711f-59ba-4e13-b13d-561087ca25d6	DHCP agent	yondu	nova	:-)	UP	neutron-dhcp-agent

Figura 28 - Captura de pantalla verificación del funcionamiento de Neutron. Fuente propia.

### 9.2.5. Nova: Compute service

“Nova es el proyecto de OpenStack que ofrece una forma de aprovisionar instancias (conocidas también como servidores virtuales). Nova soporta la creación de máquinas virtuales, servidores bare metal (a través del uso de Ironic) y tiene un soporte limitado para sistemas contenedores. Nova ejecuta un set de demonios sobre servidores Linux para proveer de ese servicio. Este requiere de los siguientes servicios adicionales para un funcionamiento básico:

Keystone: Provee identidades y autenticación para todos los servicios de OpenStack.

Glance: Provee el repositorio de imágenes de servidores. Todas las instancias en Compute (Nova) se lanzan desde imágenes de Glance.

Neutron: Éste es responsable por aprovisionar las redes físicas o virtuales que las instancias de Compute se conectan al iniciar.

Placement: Es responsable por el seguimiento de inventario de recursos disponible en la nube y de asistir en escoger qué proveedor de esos recursos serán usados para crear máquinas virtuales.” [13] (OpenStack Documentation, 2021)

Prerrequisitos: Configuramos en el nodo controlador en primer lugar.

Creamos la base de datos nova\_api, nova, nova\_cell0 y damos los accesos necesarios:

```
mysql -u root -p

MariaDB [(none)]> CREATE DATABASE nova_api;
MariaDB [(none)]> CREATE DATABASE nova;
MariaDB [(none)]> CREATE DATABASE nova_cell0;

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
    IDENTIFIED BY 'pez1988stack';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
    IDENTIFIED BY 'pez1988stack';
```

```

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
    IDENTIFIED BY 'pez1988stack';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
    IDENTIFIED BY 'pez1988stack';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' \
    IDENTIFIED BY 'pez1988stack';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%' \
    IDENTIFIED BY 'pez1988stack';

```

Configuramos los usuarios y los endpoints:

Obtenemos las variables de entorno en el archivo admin-openrc:

```
. admin-openrc
```

Creamos el usuario del servicio de nova:

```
openstack user create --domain default --password-prompt nova
```

Agregamos el usuario placement al proyecto servicio con rol de administrador:

```
openstack role add --project service --user placement admin
```

Creamos la entidad del servicio nova:

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

Creamos los endpoints de la API:

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

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

```
openstack endpoint create --region RegionOne \
    compute admin http://control:8774/v2.1
```

Instalamos y configuramos los componentes:

Instalamos los paquetes:

```
zypper install openstack-nova-api openstack-nova-scheduler \
openstack-nova-conductor openstack-nova-consoleauth \
openstack-nova-novncproxy iptables
```

Editamos en `/etc/nova/nova.conf` y configuramos:

En la sección [DEFAULT] habilitamos solo las APIs de compute y metadatos:

```
[DEFAULT]
# ...
enabled_apis = osapi_compute,metadata
```

En la sección [api\_database] y [database] configuramos el acceso a la base de datos:

```
[api_database]
# ...
connection = mysql+pymysql://nova:pez1988stack@control/nova_api

[database]
# ...
connection = mysql+pymysql://nova:pez1988stack@control/nova
```

En la sección [DEFAULT] configuramos RabbitMQ como cola de mensajes

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:pez1988stack@control
```

En las secciones [api] y [keystone\_auth\_token] configuramos el acceso al servicio de identidad:

```
[api]
# ...
auth_strategy = keystone

[keystone_auth_token]
```

```
# ...
auth_url = http://control:5000/v3
memcached_servers = control:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = admin
username = nova
password = pez1988stack
```

En las sección [DEFAULT] configuramos la opción my\_ip con la dirección de la interfaz de administración en el nodo controlador:

```
[DEFAULT]
my_ip = 172.24.250.100
```

En la sección [DEFAULT] habilitamos el soporte para el servicio de Networking:

```
[DEFAULT]
use_neutron = true
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

En la sección [vnc] configuramos el proxy VNC para usar la dirección de la interfaz de administración en el nodo controlador:

```
[vnc]
enabled = true
# ...
server_listen = 172.24.250.101
server_proxyclient_address = 172.24.250.101
```

En la sección [glance] configñuramos la ubicación de la API del servicio de imagen:

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

En la sección [oslo\_concurrency] configuramos la ruta de bloqueo:

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

En la sección [placement] configuramos el acceso al servicio placement:

```
[placement]
region_name = RegionOne
project_domain_name = Default
project_name = admin
auth_type = password
user_domain_name = Default
auth_url = http://control:5000/v3
username = placement
password = pez1988stack
```

Llenamos con tablas y datos en la base de datos utilizando los scripts incluidos:

```
su -s /bin/sh -c "nova-manage api_db sync" nova
```

Registraremos la base de datos cell0:

```
su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova
```

Creamos la celda cell1:

```
su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose" nova
```

Llenamos con tablas y datos en la base de datos utilizando los scripts incluidos:

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

Verificamos que las celdas de nova cell0 y cell1 están registrados correctamente

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

Finalizamos la instalación:

```
systemctl enable openstack-nova-api.service \
openstack-nova-consoleauth.service openstack-nova-scheduler.service \
openstack-nova-conductor.service openstack-nova-novncproxy.service
systemctl start openstack-nova-api.service \
openstack-nova-consoleauth openstack-nova-scheduler.service \
openstack-nova-conductor.service openstack-nova-novncproxy.service
```

Ahora configuramos en el nodo de cómputo que en nuestro caso es Rocky:

“Este servicio soporta varios hipervisores para desplegar máquinas virtuales. Para simplicidad esta configuración Quick Emulator Hypervisor (QEMU) con la extensión kernel-based (KVM) en el nodo de cómputo que soporta aceleración por hardware para las máquinas virtuales.” [13] (OpenStack Documentation, 2021).

Instalamos los paquetes:

```
zypper install openstack-nova-compute genisoimage qemu-kvm libvirt
```

Editamos en `/etc/nova/nova.conf` y configuramos:

En la sección [DEFAULT] habilitamos las APIs de compute y metadatos:

```
[DEFAULT]
# ...
enabled_apis = osapi_compute, metadata
```

En la sección [DEFAULT] establecemos el compute\_driver:

```
[DEFAULT]
compute_driver = libvirt.LibvirtDriver
```

En la sección [DEFAULT] configuramos la cola de mensajes RabbitMQ:

```
[DEFAULT]
transport_url = rabbit://openstack:pez1988stack@control
```

En la sección [api] y [keystone\_auth\_token] configuramos el acceso al servicio de identidad:

```
[api]

auth_strategy = keystone

[keystone_authtoken]

auth_url = http://control:5000/v3
memcached_servers = control:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = admin
username = nova
password = pez1988stack
```

En la sección [DEFAULT] configuramos la opción my\_ip con la IP de la interfaz de la red de administración en el nodo de cómputo:

```
[DEFAULT]

my_ip = 172.24.250.102
```

En la sección [DEFAULT] habilitamos el soporte para el servicio de Networking y deshabilitamos el firewall obsoleto ya que este se mudo a neutron :

```
[DEFAULT]

use_neutron = true
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

En la sección [vnc] habilitamos y configuramos el acceso de consola remota:

```
[vnc]
# ...
enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = $my_ip
novncproxy_base_url = http://192.168.4.47:6080/vnc_auto.html
```

Respecto a VNC podemos ver que: “Este servicio escucha en todas las direcciones IP y el componente de proxy solo escucha en la dirección IP de la interfaz de red de administración. La URL base indica la

locación donde puede usar un navegador web para acceder remotamente a las consolas de las instancias que residen en el nodo de cómputo.” [13] (OpenStack Documentation, 2021).

En la sección [glance] configuramos la locación de la API del servicio de imagen:

```
[glance]
# ...
api_servers = http://control:9292
```

En la sección [oslo\_concurrency] configuramos la ruta de bloqueo:

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

En la sección [placement] configuramos la locación de la API:

```
[placement]
region_name = RegionOne
project_domain_name = Default
project_name = admin
auth_type = password
user_domain_name = Default
auth_url = http://control:5000/v3
username = placement
password = pez1988stack
```

Revisamos que el módulo de kernel nbd se encuentra cargado:

```
modprobe nbd
```

No aseguramos que el módulo cargue en cada inicio del sistema añadiendo nbd a /etc/modules-load.d/nbd.conf

Finalizamos la instalación. Inicializamos los servicios incluyendo sus dependencias y lo configuramos para que se inicie automáticamente cuando el sistema inicie.

```
systemctl enable libvirtd.service openstack-nova-compute.service
systemctl start libvirtd.service openstack-nova-compute.service
```

Verificamos la operación:

Revisamos los logs en `/var/log/nova-compute.log` para ver si los accesos AMQP u otro pueda estar fallando. Usualmente a causa de los firewall de los sistemas operativos.

Revisamos si el nodo de cómputo se agregó a la tabla cell en la base de datos:

Obtenemos las variables de entorno en el archivo `admin-openrc`:

```
. admin-openrc

openstack compute service list --service nova-compute
```

Descubriremos nodos de cómputo

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

Verificamos la operación entre los dos nodos:

```
compute:~ # openstack compute service list --service nova-compute
+---+-----+-----+-----+-----+-----+
| ID | Binary | Host | Zone | Status | State | Updated At |
+---+-----+-----+-----+-----+-----+
| 14 | nova-compute | compute | nova | enabled | up | 2021-09-28T15:15:47.000000 |
+---+-----+-----+-----+-----+-----+
```

Figura 29 - Captura de pantalla funcionamiento Nova. Fuente propia.

Listamos los componentes de servicio para verificar la exitoso lanzamiento y registro de cada proceso:

```
openstack compute service list
```

```
compute:~ # openstack compute service list
+---+-----+-----+-----+-----+-----+
| ID | Binary | Host | Zone | Status | State | Updated At |
+---+-----+-----+-----+-----+-----+
| 1 | nova-consoleauth | yondu | internal | enabled | up | 2021-09-28T15:17:12.000000 |
| 2 | nova-scheduler | yondu | internal | enabled | up | 2021-09-28T15:17:19.000000 |
| 4 | nova-conductor | yondu | internal | enabled | up | 2021-09-28T15:17:13.000000 |
| 14 | nova-compute | compute | nova | enabled | up | 2021-09-28T15:17:17.000000 |
+---+-----+-----+-----+-----+-----+
```

Figura 30 - Captura de pantalla funcionamiento Nova. Fuente propia.

Listamos los endpoint de la API para saber si el servicio de identidad puede conectarse:

```
openstack catalog list
```

Name	Type	Endpoints
placement	placement	RegionOne
nova	compute	RegionOne internal: http://control:8774/v2.1 RegionOne public: http://control:8774/v2.1 RegionOne admin: http://control:8774/v2.1
neutron	network	RegionOne public: http://control:9696 RegionOne admin: http://control:9696 RegionOne internal: http://control:9696
glance	image	RegionOne internal: http://control:9292 RegionOne public: http://control:9292 RegionOne admin: http://control:9292
keystone	identity	RegionOne internal: http://control:5000/v3/ RegionOne public: http://control:5000/v3/ RegionOne admin: http://127.0.0.1:35357/v3/
placement	placement	RegionOne

Figura 31 - Captura de pantalla verificación funcionamiento de Nova. Fuente propia.

Listamos la imágenes en el servicio de imagen para verificar la conectividad:

```
openstack image list
```

ID	Name	Status
bcaddcb2-86e0-48bc-a343-e28da42f5d5d	centos7	active
52b951c5-cc5b-4e92-8b29-ef2089e441a2	cirros	active
50d3ce29-87e3-473a-9d00-33052db13ac5	ubuntu 18.04 LTS	active
25406172-aa6d-447a-a276-8ad53ab10e31	working network	active

Figura 32 - Captura de pantalla verificación funcionamiento de Nova. Fuente propia.

Revisamos que la celdas y el API de placement están trabajando correctamente y otros requisitos necesarios estan en su lugar:

```
nova-status upgrade check
```

```
compute:~ # nova-status upgrade check
+-----+
| Upgrade Check Results
+-----+
| Check: Cells v2
| Result: Success
| Details: None
+-----+
| Check: Placement API
| Result: Success
| Details: None
+-----+
| Check: Ironic Flavor Migration
| Result: Success
| Details: None
+-----+
| Check: Request Spec Migration
| Result: Success
| Details: None
+-----+
| Check: Console Auths
| Result: Success
| Details: None
+-----+
```

Figura 33 - Captura de pantalla verificación funcionamiento de Nova. Fuente propia.

## **.2.6. Horizon: Dashboard service**

El servicio final que instalaremos será el dashboard el cual permite tener una administración de OpenStack mediante una interfaz web. Este se instala en el nodo controlador. “El único servicio requerido por el dashboard es el servicio de identidad. Puedes usar el dashboard con la combinación de otros servicios, como el servicio de imagen, Compute y Neutron. Puedes también usarlo con servicios independientes como el Object Storage.” [13] (OpenStack Documentation, 2021).

Es interesante también instalar este servicio luego de Keystone para poder ver el progreso que se va realizando con la instalación de OpenStack. Es la forma más usada y parte del resultado final. A medida que los proyectos independientes se van concretando este va tomando una forma más completa.

Instalamos los paquetes:

```
zypper install openstack-dashboard
```

Configuramos el servidor web

```
cp /etc/apache2/conf.d/openstack-dashboard.conf.sample \
    /etc/apache2/conf.d/openstack-dashboard.conf
a2enmod rewrite
```

Configuramos en

```
/srv/www/openstack-dashboard/openstack_dashboard/local/local_settings.py
```

Configuramos el dashboard para usar servicio de OpenStack en el nodo controlador:

```
OPENSTACK_HOST = "control"
```

Permite a acceder al dashboard desde cualquier anfitrión:

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

Configuramos la sesión de dispositivo de almacenamiento memcached:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'

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

Habilitamos el API de identidad Keystone en la versión 3:

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

Habilitamos el soporte para dominios:

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

Configuramos las versiones del API:

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

Configuramos Default como el dominio defecto para los usuarios creados a través del dashboard:

```
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"
```

Configuramos user como el rol por defecto para los usuarios que serán creados en el dashboard:

```
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
```

Si ocupamos la opción de red , deshabilitamos el soporte para servicios de la capa 3:

```
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
```

Configuramos el time zone:

```
TIME_ZONE = "CL"
```

Finalizamos la instalación:

```
sudo systemctl restart apache2.service memcached.service
```

Para verificar el funcionamiento del dashboard accedemos a través del navegador web en <http://172.24.250.100> o <http://192.168.4.47>. Usamos las credenciales de admin o demo para el dominio default.

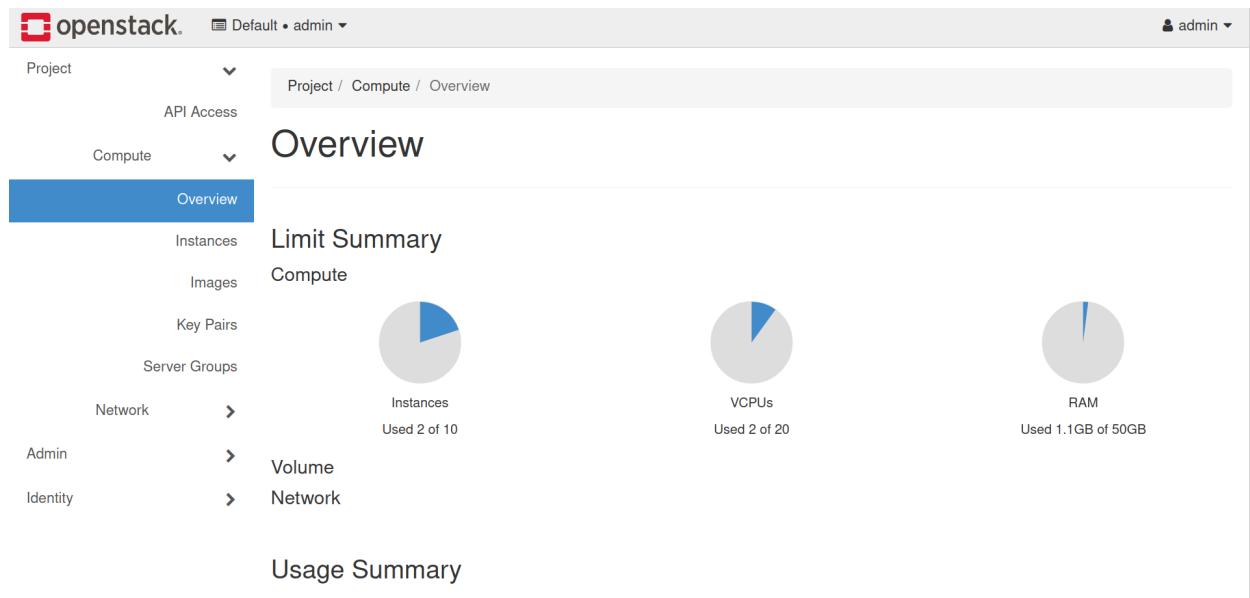


Figura 34 - Captura de pantalla verificación funcionamiento de Horizon. Fuente propia.