UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Ingeniería de Sistemas y Automática

Diseño e implementación de una nube computacional basada en OpenStack

Trabajo Fin de Máster

Máster Universitario en Automática e Informática Industrial

AUTOR/A: Martinez Gonzalvez, Mario

Tutor/a: Posadas Yagüe, Juan Luís

CURSO ACADÉMICO: 2021/2022

ÍNDICE

ILUSTRACIONES

Índice de tablas

1. Introducción

Actualmente, nos encontramos en un momento de gran auge de las tecnologías en la nube, pero este término lleva utilizándose desde hace bastante tiempo.

El término *cloud computing* hace referencia a los servicios ofrecidos a través de la red, como el uso de aplicaciones, almacenamiento entre otros, pudiendo acceder a ellos mediante servidor web.

El *cloud computing* es considerado una de las tecnologías emergentes más importantes dentro del campo de la informática. Entre otras cosas, proporciona unos recursos computacionales muy altos sin necesidad de tener un hardware de alta calidad.

Los grandes proveedores de esta tecnología utilizan una infraestructura distribuida por todo el mundo para ayudar a prevenir cualquier interrupción, y también, conlleva una gran seguridad en el intercambio de datos. Esta nueva tendencia tecnológica es un concepto surgido de la necesidad de desplazar a servidores, dedicados a este fin, todas las aplicaciones y documentos que el usuario utiliza diariamente para su labor, con el objetivo de que el mismo pueda disponer de dichos elementos en el momento que se necesiten desde cualquier lugar del planeta para poder cumplir con su trabajo más allá de la situación en la que se encuentre.

Una parte importante de un *cloud* es la parte del manejo de recursos y aplicaciones, de esta forma se pueden instanciar las especificaciones necesarias para unas necesidades en concreto. Lo que se ha programado en este proyecto es un orquestador para el manejo de todos estos recursos y poder lanzar una consola con unas especificaciones desde un servidor web en la nube.

Hoy en día, para adquirir cualquier recurso informático por pequeño que sea, es complicado con los precios desorbitados que hay en el mercado, y todo esto a causa de la escasa cantidad de silicio para la fabricación de los pequeños transistores de cualquier microchip.

Una buena opción, sobre todo para las nuevas empresas en el ámbito industrial, optan por adquirir los servicios cloud para llevar a cabo las tareas que requieran mayores recursos, sin tener que invertir demasiado dinero en ello.

Como se irá viendo en el desarrollo de este proyecto, se ha diseñado e implementado un *cloud* privado, pero pudiendo convertirla en cloud pública con tan solo abriendo los puertos del servidor donde esté alojado, en este caso un ordenador personal. Al tenerlo instalado en un portátil convencional, han ido surgiendo algunos problemas con la instalación y el diseño, que posteriormente se mencionarán en esta memoria.

En términos de diseño e implementación, descartando el problema de recursos en el sistema, este proyecto es perfectamente aplicable y escalable para la ejecución en la industria.

Este servicio puede ser utilizado para la ejecución de programas en cualquier ámbito de la industria, ya sea para la compilación de código para la ejecución de **autómatas**, **brazos robóticos** e incluso sistemas de supervisión tales como ***SCADAS*** para para monitorizar y controlar los procesos industriales u otros dispositivos vinculados con la industria 4.0.

En los siguientes apartados se podrá ir viendo cómo funciona esta tecnología, en que se basa, cuantos tipos de tecnologías *cloud computing* hay en el mercado y por qué se ha optado por *OpenStack*. También, como se ha mencionado, se ha programado un orquestador con el que cualquier usuario pueda controlar los recursos que necesite para controlar aplicaciones en el *cloud*.

2. Objetivos

Nos encontramos en una época donde, por culpa de la pandemia y guerras innecesarias, es complicado crear una empresa en la industria e incluso mantenerla.

El objetivo general de este proyecto es reducir de manera sustancial la compra masiva de dispositivos electrónicos tanto en el campo de la industria como en cualquier otro campo realizando un *cloud* donde poder crear máquinas virtuales y utilizarlas desde cualquier dispositivo sin depender de ningún recurso electrónico.

Para conseguir dicho objetivo se ha desglosado en los siguientes subobjetivos:

1. **Instalación** *OpenStack* en un equipo.
2. Realización de las configuraciones necesarias.
3. Ampliar instalación añadiendo nuevos servicios que puedan ser útiles para la plataforma.
4. **Desarrollar una plataforma *cloud*** para reducir el coste en proyectos cualquier empresa.
5. Lanzar instancias de máquinas virtuales.
6. **Reducir la compra de dispositivos electrónicos** tales como ordenadores, servidores para empresas de la industria.
7. Reducir el **impacto medioambiental** del uso excesivo de aparatos electrónicos.
8. Aplicar conocimientos sobre **redes** y **programación**.

Aunque este proyecto está orientado al ámbito empresarial, se han aplicado conocimientos adquiridos sobre las asignaturas de **programación de sistemas** y **redes distribuidas para control** de este máster.

3. Cloud computing

Cloud computing (Computación en la nube) se basa en la utilización y desarrollo de un proceso informático en la red. Lo más importante que ofrece este tipo de tecnología es la manera de acceder a recursos y aplicaciones en cualquier parte del mundo con solo tener conexión a internet.

3.1 Historia

Antes de poder entender esta tecnología, es importante saber en el contexto en el que surge.

En los años 60, comienzan los planteamientos de computación y se crean los primeros servidores privados en las grandes empresas.

En 1990, como ya sabemos, comienzan las bases de lo que ahora llamamos internet, y comienza a expandirse por todo el mundo. A finales de la década, surgió la idea de *cloud computing* en un seminario del profesor **Ramnath K Chellappa**, y también se sumaría el pionero en la creación de internet **John Mccarthy.**

El impacto más notorio de esta tecnología en la sociedad fue en los primeros años de creación de *Amazon* a principios de siglo. La famosa empresa, se enfrentaba a los problemas de escala típicos de una compañía de comercio electrónico. Para hacerles frente, la compañía tuvo que construir sistemas internos sólidos para hacer frente al gran crecimiento que estaba experimentando. Fueron visionarios a la hora de apostar por el *cloud computing* para su cambio en la infraestructura de red.

A principios de 2006, se crea un servicio secundario por la empresa *Amazon* llamada ***Amazon Web Services (AWS),*** hoy, la empresa líder en *cloud computing.* Poco más tarde se sumarían las empresas líderes en tecnología como son *Google* e *IBM* en la investigación de esta técnica.

En 2008, surge *Eucalyptus*, una plataforma basada en API´s de AWS, crean la primera *cloud* privada. A mitades de este año, muchas empresas empiezan a cambiar su infraestructura a servicios en la nube.

A partir del 2010 hasta nuestros días, han surgido muchas plataformas de *cloud* muy importantes, como *Microsoft Azure,* EC2 de Amazon, que

permite alquilar consolas virtuales para ejecutar cualquier aplicación en la nube. En este mismo año, la *NASA* y *RackSpace* lanzaron conjuntamente ***OpenStack*** como un proyecto de software de *cloud* de código abierto.

OpenStack es una plataforma de software de código abierto para nubes privadas y públicas. Como plataforma que ofrece infraestructura como servicio (IaaS), permite a las empresas agregar servidores y componentes de redes y almacenamiento de manera fácil y eficiente a su nube.

3.2 Servicios en la nube

Existen tres modelos principales de *cloud computing*. Cada modelo muestra una parte distinta en la pila de informática en la nube.

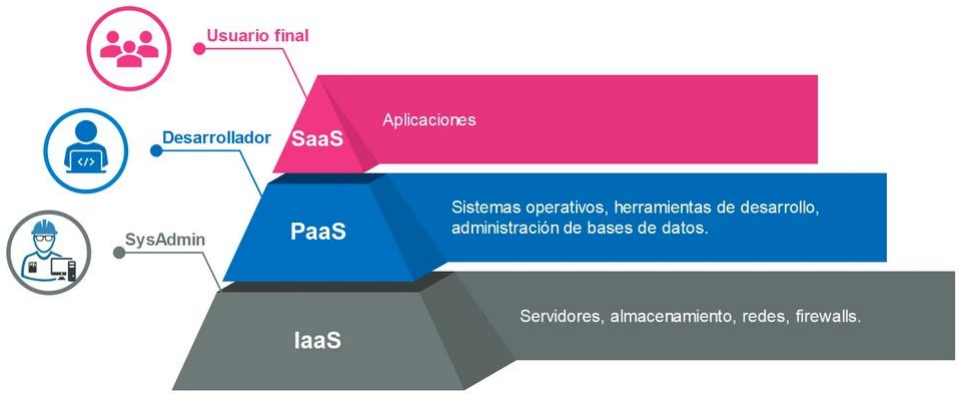

Ilustración 1 Jerarquía servicios cloud

3.2.1 Infraestructura como servicio (IaaS)

La infraestructura como servicio, que de forma abreviada se suele llamar *IaaS*, contiene los bloques de creación fundamentales para la TI en la nube.

Por lo general, permite acceder a las características de conexión en red, a los equipos (virtuales o en software dedicado) y al espacio de almacenamiento de datos.

La infraestructura como servicio ofrece el mayor nivel de flexibilidad y control de la administración en torno a recursos de TI y guarda el mayor parecido con los recursos de TI existentes con los que muchos departamentos de TI y desarrolladores están familiarizados.

La infraestructura como servicio (IaaS) da más recursos a la red existente. IaaS es básicamente una extensión de cualquier infraestructura de red actual. Se pueden agregar nuevas direcciones IP, almacenamiento, balanceadores de carga de virtualización (redes virtuales) y firewalls.

Todos estos recursos suelen costar miles de euros para una empresa, pero utilizando IaaS, se pueden agregar sólo los recursos que se necesitan y pagar por los recursos que se utilizan. El resultado es que se pueden escalar estos recursos hacia arriba y abajo según los requisitos que se necesiten. El ahorro de costes es enorme, sobre todo si la empresa está empezando y necesita una gran cantidad de recursos, pero no tiene el presupuesto para una red completa. Algunos ejemplos de IaaS son *Amazon EC2*, *DigitalOcean* y *OpenStack*.

3.2.2 Plataforma como servicio (PaaS)

Las plataformas como servicio eliminan la necesidad de las compañías de administrar la infraestructura subyacente (normalmente hardware y sistemas operativos) y permiten centrarse en la implementación y la administración de sus aplicaciones.

Esto contribuye a mejorar su eficacia, pues los clientes de estos servicios no tienen que preocuparse del aprovisionamiento de recursos, la planificación de la capacidad, el mantenimiento de software, los parches ni ninguna de las demás arduas tareas que conlleva la ejecución de cualquier aplicación.

3.2.3 Software como servicio (SaaS)

El software como servicio proporciona un producto completo que el proveedor del servicio ejecuta y administra. En la mayoría de los casos, quienes hablan de software como servicio en realidad se refieren a aplicaciones de usuario final.

Con la contratación de un SaaS, no hay que pensar en cómo se mantiene el servicio ni en cómo se administra la infraestructura subyacente. Solo habría que preocuparse por cómo utilizar ese sistema de software en concreto.

Con el auge de los smartphones, SaaS es cada vez más popular entre las empresas. La mayoría de las aplicaciones móviles tienen algún tipo de componente SaaS para trabajar con su red interna y la base de datos. La gran ventaja de SaaS es que las aplicaciones están disponibles en cualquier momento, y es el proveedor de la nube el que debe tener precauciones para proteger los datos de los usuarios. El proveedor SaaS debe también tener servidores de copia de seguridad y centros de datos para asegurarse de que su aplicación SaaS está siempre disponible y haya poco o ningún tiempo de inactividad.

Un ejemplo común de una aplicación SaaS es un programa de correo electrónico basado en la web que permite enviar y recibir mensajes sin tener que administrar la incorporación de características ni mantener los servidores y los sistemas operativos en los que se ejecuta el programa de correo electrónico.

3.3 Modelos de implementación

Teniendo en cuenta los modelos informáticos de *cloud*, hay tres posibles implementaciones de estas según las necesidades del cliente o la empresa.

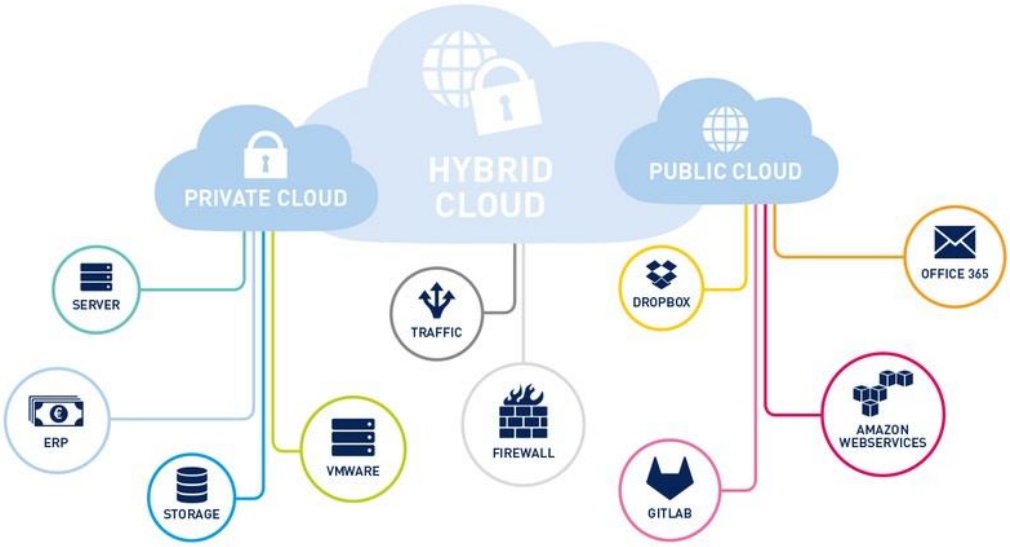

Ilustración 2 Modelos de cloud

3.3.1 Public cloud

Las nubes públicas están disponibles para el público en general y cualquier dato creado se almacena en servidores de terceros. Dado que los servidores pertenecen a proveedores de servicios que los gestionan y administran cualquier recurso del pool, se elimina el requisito de que una empresa deba comprar su propio hardware.

Las empresas proveedoras tienen sus propios recursos y los proporcionan como un servicio gratuito o, en el caso de Microsoft Azure, según el modelo de pago por uso.

Una aplicación basada en la nube se encuentra implementada totalmente en la nube, de modo que todas las partes de la aplicación se ejecutan en esta. Las aplicaciones en la nube se han creado directamente en la nube o se han transferido de la infraestructura existente para aprovechar los beneficios de la informática en la nube.

Las aplicaciones basadas en la nube se pueden construir en partes de infraestructura de bajo nivel o pueden utilizar servicios de nivel superior que proporcionan abstracción de los requisitos de administración, arquitectura y escalado de la infraestructura principal.

3.3.2 Hybrid cloud

Un modelo de despliegue híbrido está mejor optimizado ya que este modelo utiliza lo mejor de las nubes públicas y privadas. Permite la máxima combinación de las características para que una empresa pueda elegir lo que mejor se adapte a sus necesidades.

Una implementación híbrida es una manera de conectar la infraestructura y las aplicaciones entre los recursos basados en la nube y los recursos existentes situados fuera de la nube.

El método más común de implementación híbrida consiste en conectar la nube y la infraestructura existente en las instalaciones para ampliar e incrementar la infraestructura de la organización en la nube al mismo tiempo que se conectan estos recursos en la nube con el sistema interno.

3.3.3 Private cloud

La implementación local no aporta muchos de los beneficios de la informática en la nube, pero a veces se utiliza por su capacidad de ofrecer recursos dedicados.

En la mayoría de los casos, este modelo de implementación es idéntico al de la infraestructura de TI antigua, mientras que utiliza tecnologías de virtualización y administración de aplicaciones para intentar incrementar el uso de los recursos.

3.4 Beneficios

El cloud ofrece un acceso sencillo a cualquier tecnología para innovar y crear todo lo que se pueda imaginar

La nube brinda fácil acceso a una variedad de tecnologías para innovar más rápido y construir cualquier cosa que se pueda imaginar.

Es fácil activar rápidamente servicios de infraestructura como cómputo, almacenamiento y bases de datos hasta IoT, aprendizaje automático, lagos de datos, análisis y más.

Con la informática en la nube, ya no hace falta aprovisionar recursos en exceso con antelación para gestionar niveles pico de actividad comercial a futuro, muchas empresas ofrecen planes y servicios según las necesidades del usuario. Se puede ajustar la escala de estos recursos para aumentar o disminuir la capacidad instantáneamente a medida que cambien las necesidades de cualquier negocio.

4. Cloud Services

Cada vez son más las empresas que contratan servicios *cloud* para cubrir necesidades de IT, como almacenamiento de bases de datos, software en la nube o programas de gestión de recursos empresariales en la industria.

El *cloud computing* reduce mucho los costes en infraestructura IT y su mantenimiento dentro de la empresa, aunque implique contratar un servicio externo.

Las empresas dedicadas a ofrecer servicios *cloud computing* ponen a disposición de sus clientes una amplia gama de servicios, que además pueden ajustarse a las necesidades reales de cada uno de ellos en momentos determinados, permitiendo la ampliación o disminución de las funciones contratadas.

Actualmente, hay infinidad de servicios, pero como todo mercado, siempre lideran unos pocos.

4.1 Amazon Web Services

Aunque este proyecto surgió para ser utilizado dentro de la empresa, ha llegado hasta el público colocándose como empresa líder en cloud computing y actualmente ofrece la plataforma como servicio (PaaS).

Como todas las grandes empresas en este mercado, proporcionan una versión de prueba limitada durante un tiempo, y después se paga por los recursos y arquitectura que se utilizan.

Los servicios más populares que ofrece *Amazon* son:

- **Amazon lightsail**: ofrece instancias de servidor virtual privado (VPS) fáciles de usar, contenedores, almacenamiento, bases de datos.
- **Amazon Elastic Compute Cloud (EC2):** ofrece la plataforma de computación más amplia y profunda, con más de 500 instancias y la posibilidad de elegir el procesador, almacenamiento, redes, sistema operativo y modelo de compra más reciente para poder a ajustarla al máximo a las necesidades de su carga de trabajo.

- **Amazon Simple Storage Service (S3)**: servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, seguridad.

Aunque ofrecen muchos servicios y de gran calidad, estos se encarecen mucho a medida que se necesitan más recursos, y para cualquier empresa resultaría una gran inversión.

A continuación, una tabla de ventajas y desventajas sobre el servicio que ofrecen:

Ventajas	Desventajas
Actualmente, la empresa de cloud más fuerte del mercado, lo que conlleva una gran confianza al obtener sus servicios.	La plataforma es genérica, no es adaptable totalmente para tus necesidades.
Ofrece más herramientas que cualquier otra plataforma.	Difícil de entender al principio.
No es excesivamente caro.	

Tabla 1 Ventajas y desventajas AWS

4.2 Google Cloud

Como AWS, esta plataforma tiene muchos servicios que ofrecer, y como cualquier otro, ofrecen un periodo de prueba gratuita.

Google cloud en su magnitud, se podría denominar de tipo PaaS

El servicio más utilizado en su plataforma es el *computer engine,* este proporciona servidores virtuales donde correr software.

Una de las innovaciones que ofrecen actualmente es *Google Kubernetes Engine,* este se basa en *Google cloud* con una distribución en **contenedores,** ayuda a en el desarrollo para tener una mayor rapidez, e implementa software de manera eficaz.

Como AWS, hay que tener en cuenta lo mejor y lo peor que tiene esta plataforma:

Ventajas	Desventajas
Alto nivel de seguridad y protección de datos.	Tiene menos funciones que cualquier otra plataforma.
Al ser una plataforma de Google, tendremos el rendimiento y calidad de una compañía grande.	Nada en la plataforma es gratuito
Precios competitivos, aunque no los más baratos	

Tabla 2 Ventajas y desventajas Google cloud

4.3 Microsoft Azure

La plataforma *Azure*, es conocida por su gran seguridad en el servicio, ofrece una amplia gama de servicios, pudiendo con estos realizar un servicio al cliente como **IaaS** y **PaaS,** y adicionalmente, cuenta con un SaaS en su gama de productos *Office 365* de los que se pueden aprovechar sin contratar un *cloud* en específico.

Los servicios de *Azure* son de gran ayuda si lo que se necesita es un *cloud híbrido*. Ofrecen prácticamente los mismos servicios que las anteriores compañías, con el mismo mecanismo de precios, solo se paga lo que se consume.

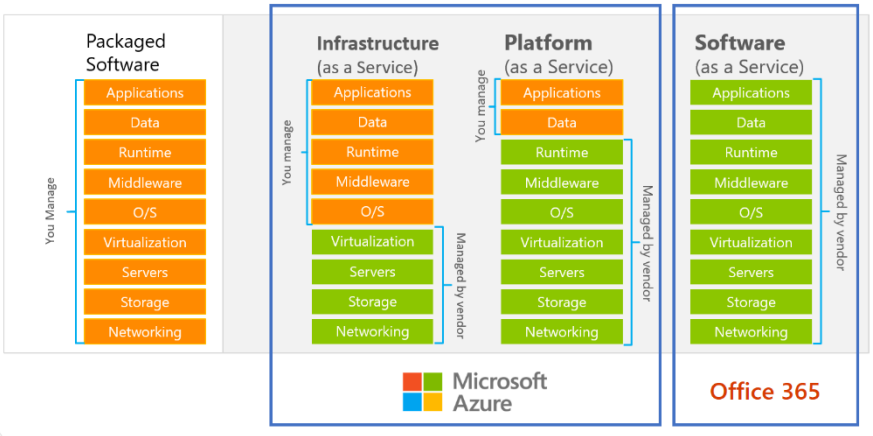

Ilustración 3 Servicios Microsoft Azure

Siguiendo la dinámica de las anteriores plataformas, en la siguiente tabla podemos ver los pros y contras de esta plataforma:

Ventajas	Desventajas
Los recursos se piden bajo demanda, se puede activar o desactivar fácilmente.	Se necesita instalar aplicaciones en local para su utilización.
Ofrece la posibilidad de tener una infraestructura mundial.	Plataforma poco intuitiva
Tiene una buena asistencia técnica.	Hace falta una supervisión continua de los costes de contratación, ya que estos varían.

Tabla 3 Ventajas y desventajas Azure

4.4 OpenStack

OpenStack es un proyecto *Cloud Computing* cuyo fin es crear una plataforma *open source* que cumpla con las necesidades de los proveedores de nubes públicas y privadas, independientemente de su tamaño, que sea fácil de implementar y escalable.

Comercialmente se define como un sistema operativo en la nube que permite controlar recursos de computación, almacenamiento y redes a través de un *dashboard* y API´s estándares de programación.

Muchas empresas actualmente utilizan esta tecnología para crear nubes privadas, aunque es complicado de implementar al principio, es de las mejores plataformas que se pueden utilizar, ya que continuamente hay actualizaciones y soporte de esta, pero puede llegar a ser un problema en algunos casos.

Openstack es similar a *Vmware vCloud* el cual ha quedado muy retrasado respecto a *Openstack* y cuyas licencias salen muchísimo más caras.

No solo es gratis en su totalidad, este tiene una licencia apache 2.0. Pero hay empresas como Red Hat que sí cobran por el soporte y uso de este sistema.

Ventajas	Desventajas
Autonomía para cualquier usuario para crear instancias, no es necesario una cuenta de administrador para ello.	Se actualiza cada seis meses, por lo que hay que estar al tanto de las nuevas funciones y de otras que se eliminan.
Escalable.	No hay mucho soporte técnico, hay que recurrir a manuales poco intuitivos para.
Es de código abierto, cualquiera puede aportar en su desarrollo.	Solo funciona con Linux.

Tabla 4 Ventajas y desventajas OpenStack

4.5 Otros servicios

Como se ha visto, existen muchos proveedores de *cloud*, y todos ofrecen más o menos los mismos servicios, aunque cada uno con alguna diferencia.

Si nos ponemos a investigar el código base de cualquiera de las otras plataformas podríamos ver que la mayoría de estas podría estar basada en ***OpenStack*** ya que este puede ofrecer los mismos servicios que cualquier otro con la ventaja de ser código abierto.

Aquí se muestra un gráfico sobre la popularidad de los servicios *cloud* durante el pasado año:

Ilustración 4 Diagrama de Gartner servicios cloud 2021

5. Solución adoptada

Gracias a los servicios que se pueden ofrecer mediante la implementación de ***Openstack*** se ha realizado un orquestador con el que poder lanzar máquinas virtuales desde cualquier parte.

Los *cloud* que ofrecen las grandes empresas no se pueden moldear al gusto, solo se puede utilizar el software que ofrecen con un precio definido.

En este caso, se ha realizado la instalación y diseño del software desde una máquina virtual en un portátil personal con recursos limitados.

Obviamente, estamos ante un software de *cloud* privada, ya que solo se va a utilizar para la realización del proyecto. En cambio, si queremos convertir el orquestador en *cloud* pública, solo haría falta añadir el software a un dominio en internet y así poder utilizarlo en cualquier parte, aunque es poco recomendable si el ordenador en el que está instalado no tiene los suficientes recursos.

Para una correcta implementación, lo más óptimo sería utilizar un servidor dedicado con los recursos suficientes para la buena ejecución del software, o si se quiere ir a más, disponer de un ***Data Center*** donde poder almacenar más máquinas virtuales y obtener más velocidad de procesamiento.

Para una ejecución inmediata, se ha llevado a cabo el orquestador mediante *Django,* donde se ha desarrollado la web mediante Python para la programación del *Back-End*, con llamadas mediante API a *OpenStack*, y para una visualización de las consolas de las máquinas virtuales y los diferentes recursos que se necesitan para ello, la programación ha sido mayoritariamente con *JavaScript,* y minoritariamente *HTML* y *CSS* para el *Front-End.*

6. Economía

Gracias a la infinidad de opciones que ofrece ***OpenStack,*** una de las más importantes es la monitorización de los recursos empleados para la ejecución de instancias y almacenamiento que se utiliza en cada máquina virtual se pueden calcular los costes que estos conllevan, ya sean de electricidad o de desgaste de hardware.

En este caso, se propone como caso práctico una empresa privada de ámbito tecnológico, nueva en el mercado, que precisa de varios ordenadores para sus empleados o para ejecutar software. Para monitorizar todos los recursos que utiliza la empresa, *OpenStack* ofrece un servicio que mide las métricas utilizadas que posteriormente se explicará, así se podrían realizar los cálculos de los recursos utilizados por esta empresa y poder realizar los recibos correspondientes.

Como posteriormente se verá, se ha creado un ***dashboard*** para visualizar los recursos que se están utilizando en el momento, así la empresa suministrada podrá tener todos los recursos que se utilizan totalmente controlados.

Estas estadísticas solo pueden ser pedidas por el administrador, todo mediante llamadas con API a ***OpenStack,*** y este devuelve, según la petición, más o menos detalles sobre los recursos que se están utilizando y el tiempo que llevan en ejecución las máquinas virtuales.

7. Arquitectura de OpenStack

OpenStack es un proyecto *open source* enfocado al *Cloud Computing*, diseñado para desplegar cualquier tipo de nube, orientadas a ofrecer infraestructuras como servicio a los usuarios, escrito en Python, con arquitectura modular, y compuesto por varios módulos diferentes, de ellos unos nueve son los más importantes. Cada módulo proporciona herramientas para que los administradores puedan gestionar los recursos a través de *dashboards* o por línea de comandos, para que los usuarios puedan hacer uso de los servicios a través de una interfaz web o por llamadas al sistema.

Actualmente hay importantes empresas que emplean *OpenStack* para su infraestructura de nube privada. A pesar del aumento tanto en popularidad como en utilización, aún no da soporte a versiones muy antiguas del software. Muchas de las empresas que precisan de un cloud, aún no se han atrevido a implementar *OpenStack*, ya que resulta complicada esta tarea, y también configurarlo y mantenerlo, haciendo que muchas empresas prefieran quedarse con AWS u otras más famosas.

7.1 Servicios principales

La plataforma OpenStack se puede dividir en varios servicios con los que podemos hacer funcionar la estructura de un *cloud.*

Para conseguir una arquitectura bien estructurada de lo que es *OpenStack*, se suelen instalar todos los servicios a la vez, y luego se utilizan los necesarios. Cada servicio depende de otro normalmente, por lo que, si no instalamos el servicio que depende del otro, probablemente no funcione ese servicio correctamente.

En los siguientes apartados se explican los servicios más importantes de *OpenStack.*

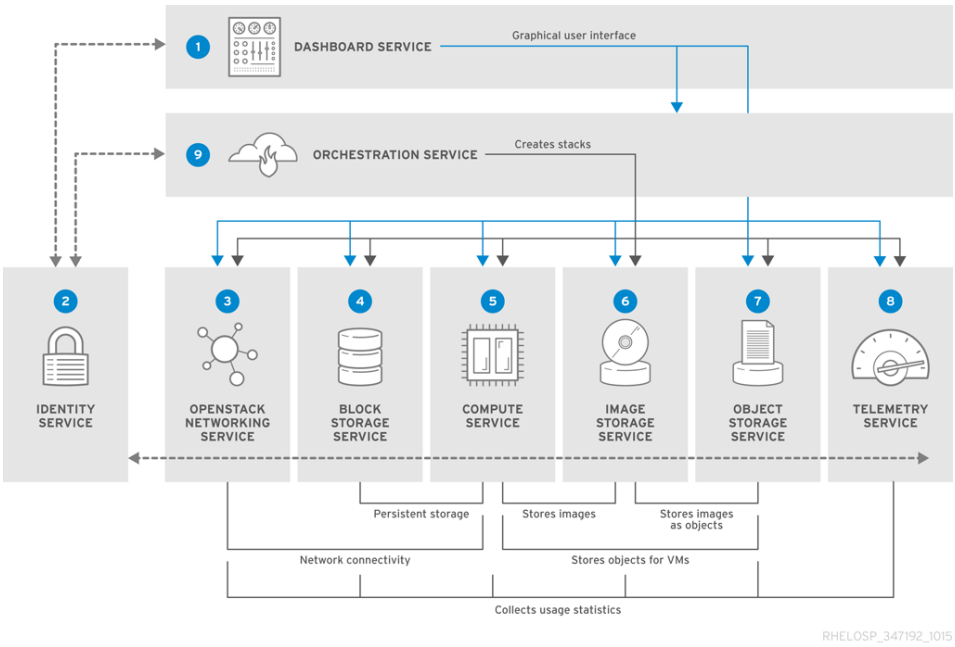

Ilustración 5 Servicios OpenStack ©

7.1.1 Keystone

Este módulo se encarga de la autenticación de la plataforma y los demás servicios.

Por cada petición que se haga dentro de la plataforma, es necesario un token que este servicio provee, este módulo hace más seguro cada rincón dentro de la nube.

Dentro de *keystone,* existen unos subservicios con los que trabaja conjuntamente este módulo:

- **Identity**: Valida las credenciales de los usuarios o grupos que puedan entrar a la plataforma.

- **Resources**: Proporciona información sobre proyectos de los usuarios y dominios.

- **Token**: Se utiliza para acceder a la plataforma y hacer peticiones a cualquier otro módulo de *OpenStack*.

- **Endpoint**: Es la URL con la que podemos hacer las peticiones a los demás servicios.

- **Tenant**: Es un contenedor para agrupar o aislar recursos. Dependiendo del servicio, un *tenant* puede ser un cliente, una cuenta, una organización o un proyecto.

Estos son los subservicios más comunes, son los que se van a utilizar para este proyecto en concreto, a parte de estos, podemos encontrar muchos más.

En la siguiente figura podemos ver el proceso que sigue el módulo *keystone* para la identificación de un usuario:

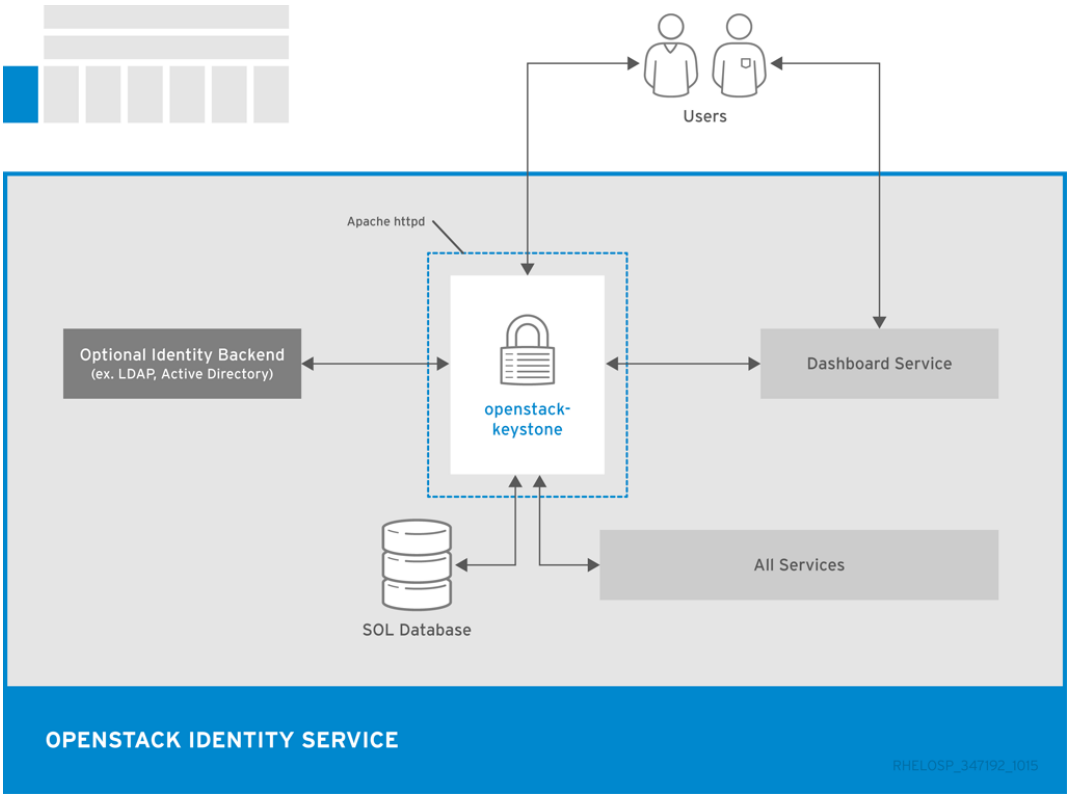

Ilustración 6 Proceso identificación keystone ©

7.1.2 Horizon

Este módulo es la interfaz gráfica por defecto que proporciona *OpenStack*. En este *dashboard* se pueden acceder a algunos recursos de una manera básica. Este no es muy recomendable de utilizar ya que, si no tenemos unos recursos muy altos, como es este caso, puede fallar. Es el módulo que más recursos utiliza de todos.

Este nos puede servir al principio para saber que toda la implementación de la nube ha sido correcta y está en funcionamiento. Además, poder crear usuarios una vez esté funcionando.

En el caso de este proyecto, nuestro servicio *horizon* sería la plataforma cloud desarrollada.

7.1.3 Heat

Heat es el proyecto principal de la plataforma *OpenStack*.

Implementa un motor de orquestación para lanzar múltiples aplicaciones compuestas en la nube basadas en plantillas en forma de archivos de texto que pueden tratarse como código.

Un formato nativo de plantillas de *Heat* está en desarrollo, pero *Heat* también se esfuerza por proporcionar compatibilidad con el formato de plantillas de *AWS Cloud Formation*, de modo que muchas plantillas existentes de *AWS* puedan ser lanzadas en *OpenStack*. Heat proporciona tanto una API ReST nativa de *OpenStack* como una API de consulta compatible con *AWS*.

Básicamente, este módulo convierte toda la infraestructura en código.

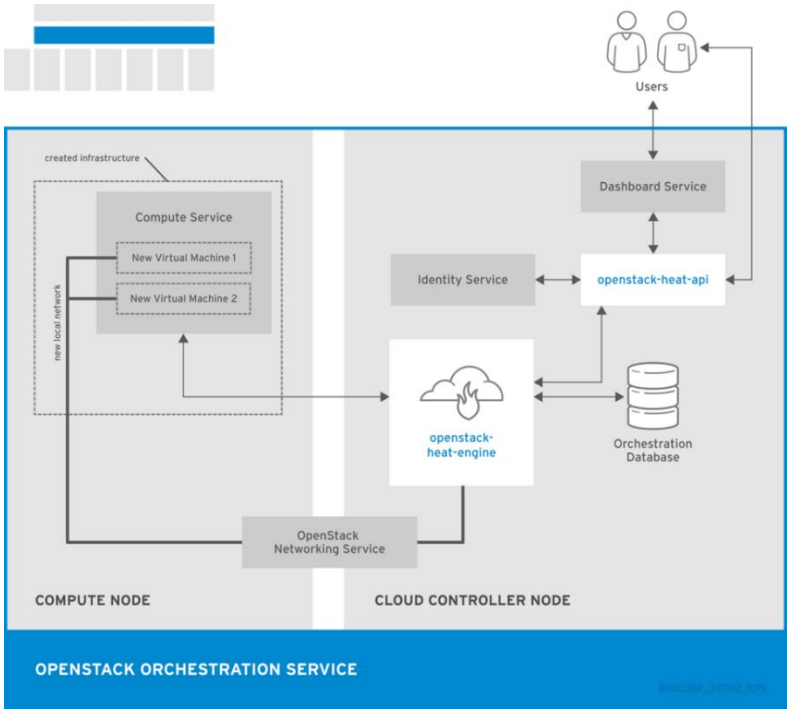

Ilustración 7 Diagrama funcionamiento Heat ©

7.1.4 Nova

Este es el módulo que administra las instancias. Ya sea para crearlas, apagarlas o realizar cualquier acción sobre ellas.

Nova soporta la creación de máquinas virtuales, servidores baremados, y tiene un soporte limitado para contenedores de sistema. Este se ejecuta como un conjunto de "*Daemons*" sobre servidores Linux existentes para proporcionar ese servicio.

Nova es un controlador de estructura *cloud*, que es la parte principal de un sistema de IaaS. Está diseñado para gestionar y automatizar las peticiones de los usuarios y/o grupos, y trabaja con tecnologías de visualización.

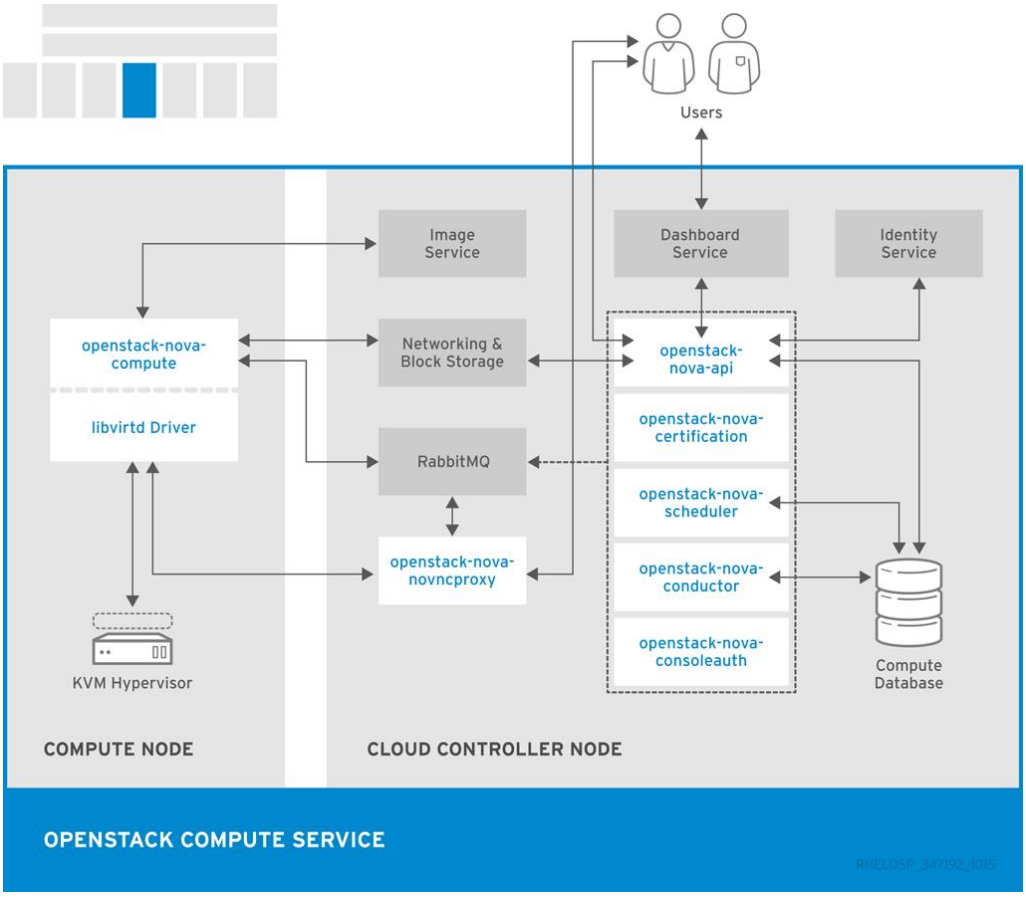

Ilustración 8 Diagrama funcionamiento Nova ©

7.1.5 Glance

Este servicio se encarga de la parte más importante para la creación de las máquinas virtuales, la gestión de imágenes.

Con este módulo se puede subir cualquier tipo de imagen, especificando su extensión, y según los permisos que se tengan dentro de la plataforma, se pueden utilizar cualquier imagen guardada.

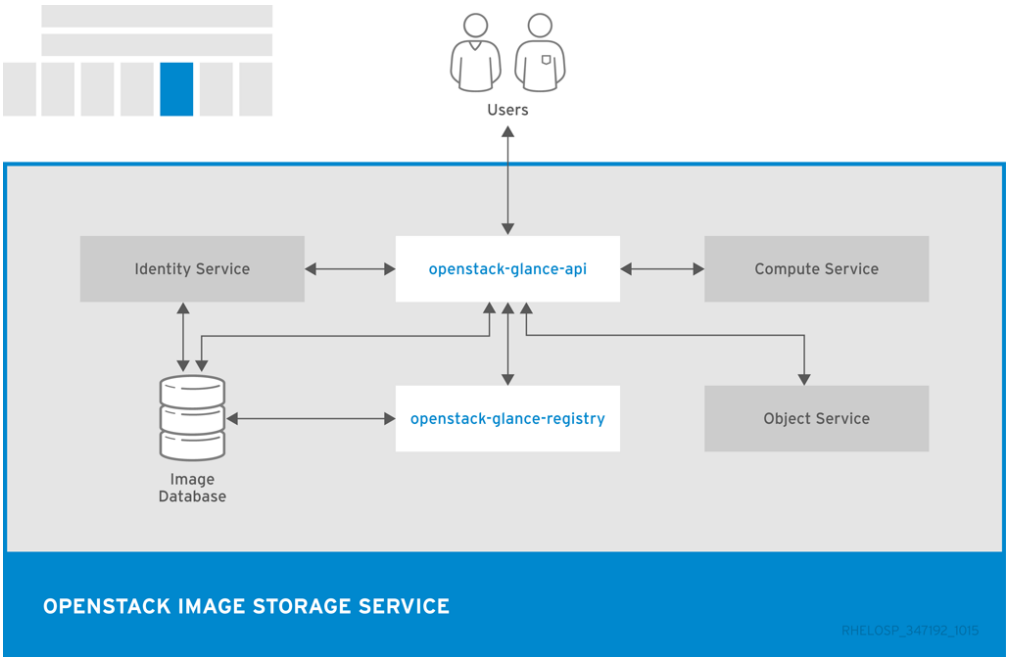

Ilustración 9 Diagrama de funcionamiento Glance ©

7.1.6 Cinder

Esta parte de *OpenStack* es la que se encarga de administrar el almacenamiento de cada máquina virtual.

Este pone a disposición un catálogo de dispositivos de almacenamiento basados en bloques con diferentes características.

Cinder presenta capacidades de almacenamiento básicas, como replicación, administración de instantáneas y clones de volumen. Sin embargo, debido a que *cinder* es una capa de abstracción para la administración del almacenamiento, es posible que un usuario no tenga acceso a las características especiales y la funcionalidad de un dispositivo o sistema de almacenamiento determinado, a menos que el administrador del *cloud* haga que esas capacidades estén disponibles a través de controladores específicos del producto.

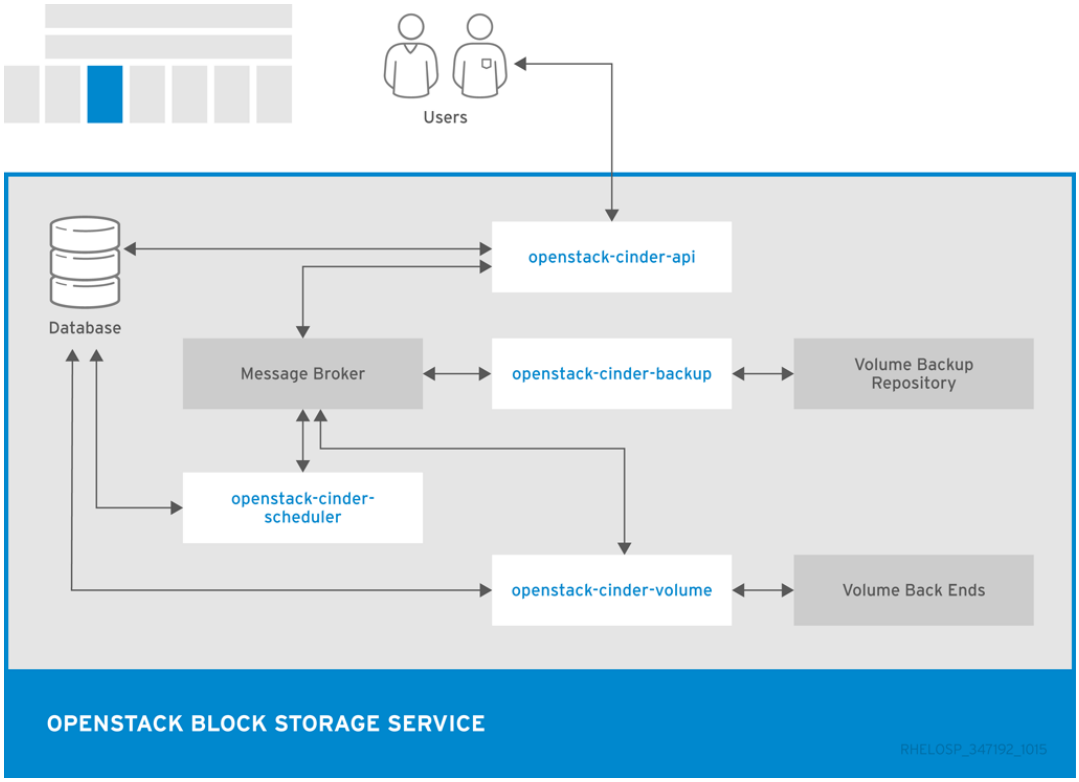

Ilustración 10 Diagrama almacenamiento Cinder ©

7.1.7 Swift

Se entiende como el módulo de almacenamiento que administra datos como objetos en contraposición al almacenamiento de archivos, que usa una estructura de directorios jerárquica. Un objeto contiene datos, metadatos y un identificador global único.

Este módulo es necesario una vez se disponga de un *Data Center* lo suficientemente grande para el almacenamiento doble de datos, ya que este se encarga de realizar *Backup* de todos los datos de la nube.

Swift es un sistema de almacenamiento redundante y escalable. Los objetos y los archivos se escriben en varias unidades de disco repartidos por los servidores del *Data Center*, con *OpenStack* responsable de asegurar la replicación y la integridad de los datos. En caso de que un servidor o disco falle, *OpenStack* replica su contenido desde otros nodos activos a nuevas ubicaciones.

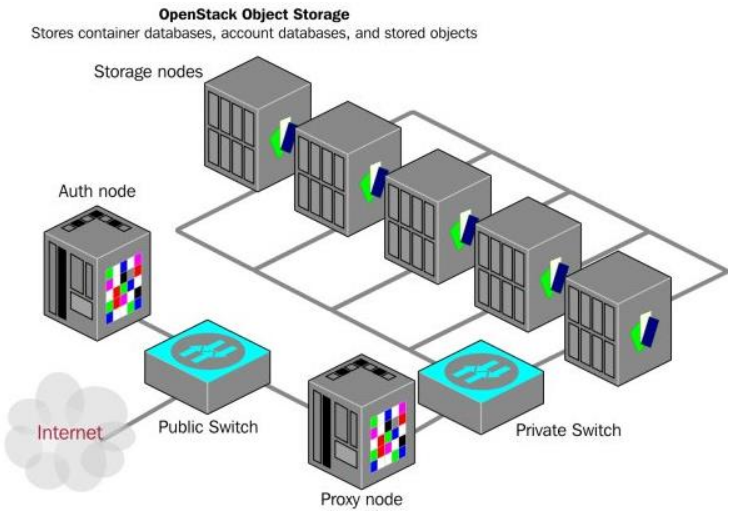

Ilustración 11 Sistema de almacenamiento Swift

7.1.8 Neutron

Neutron es el componente de Openstack que gestiona las redes definidas por software.

Incluye muchas de las tecnologías que usamos hoy en cualquier *Data center*, tal como *switches*, routers, balanceadores de carga, VPN y firewalls.

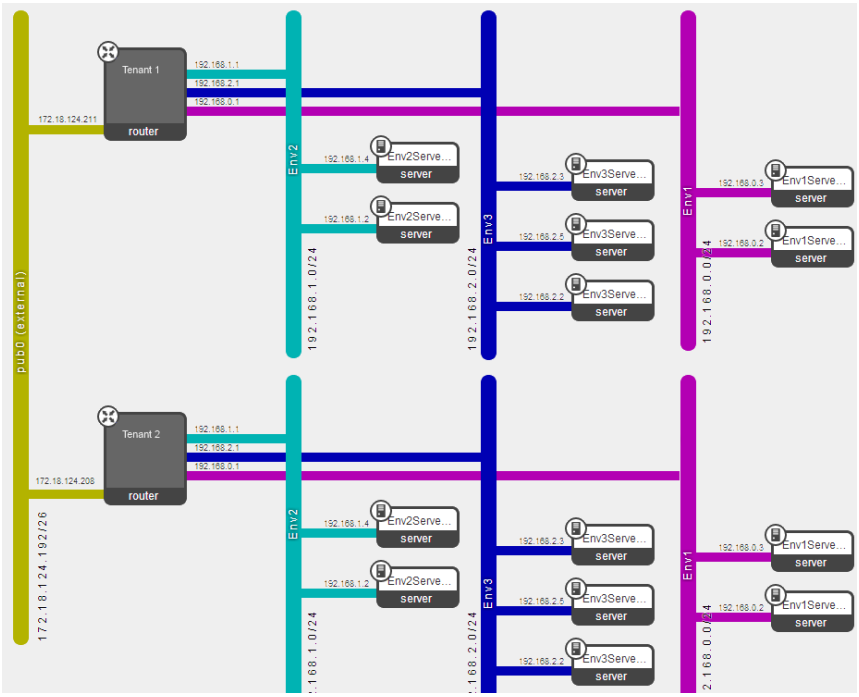

Ilustración 12 Ejemplo topología red Neutron

Esta parte de la nube se podría implementar para el desarrollo de la plataforma *cloud* dentro del *dashboard*, pero harían falta conocimientos amplios sobre el lenguaje de programación *Angular*.

7.1.9 Ceilometrer

Este módulo nos permite recolectar todos los recursos utilizados en el cloud para poder utilizarlos en la parte económica del proyecto. Estos como se verá más adelante en la memoria se han ordenado en un *dashboard* donde se verán todos los recursos en ejecución al momento.

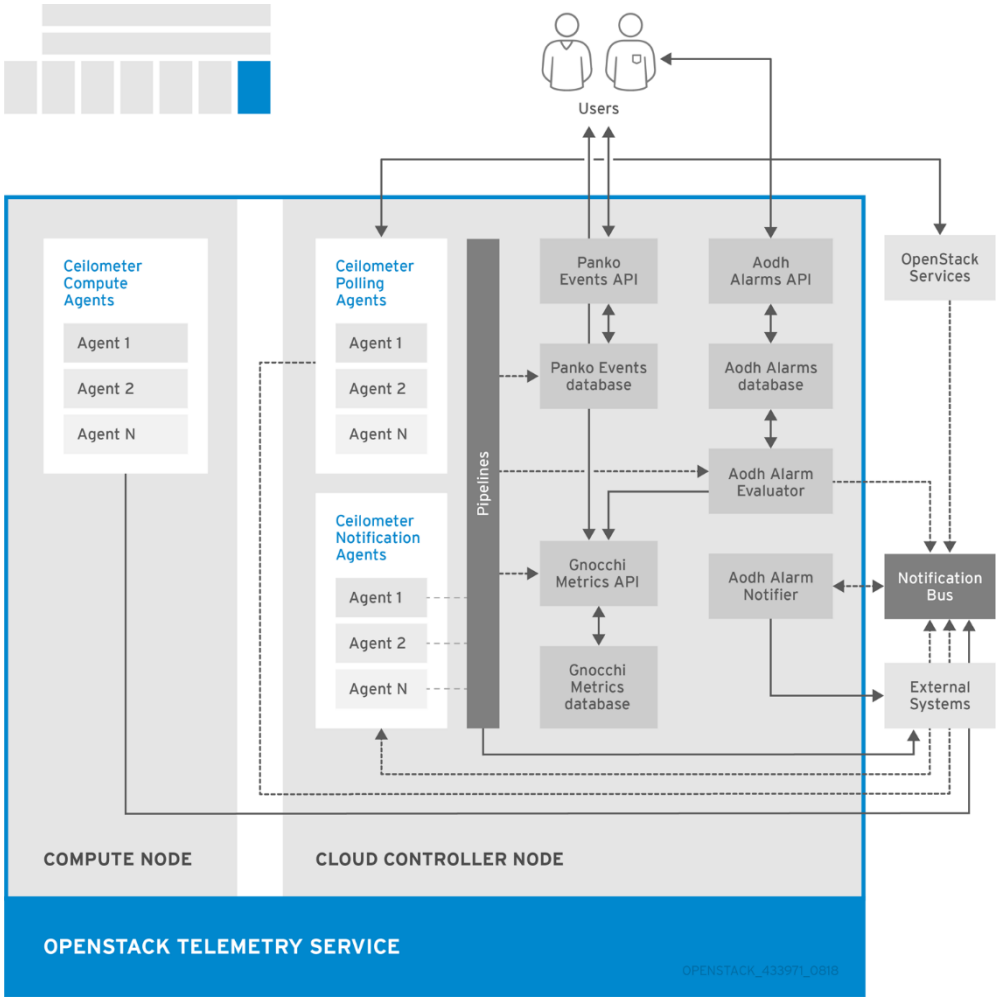

Ilustración 13 Diagrama de funcionamiento de Ceilometrer ©

7.2 Arquitectura general

OpenStack proporciona en sus documentos un diagrama conceptual para entender toda la relación que tienen sus servicios en la capa más básica, y es la siguiente:

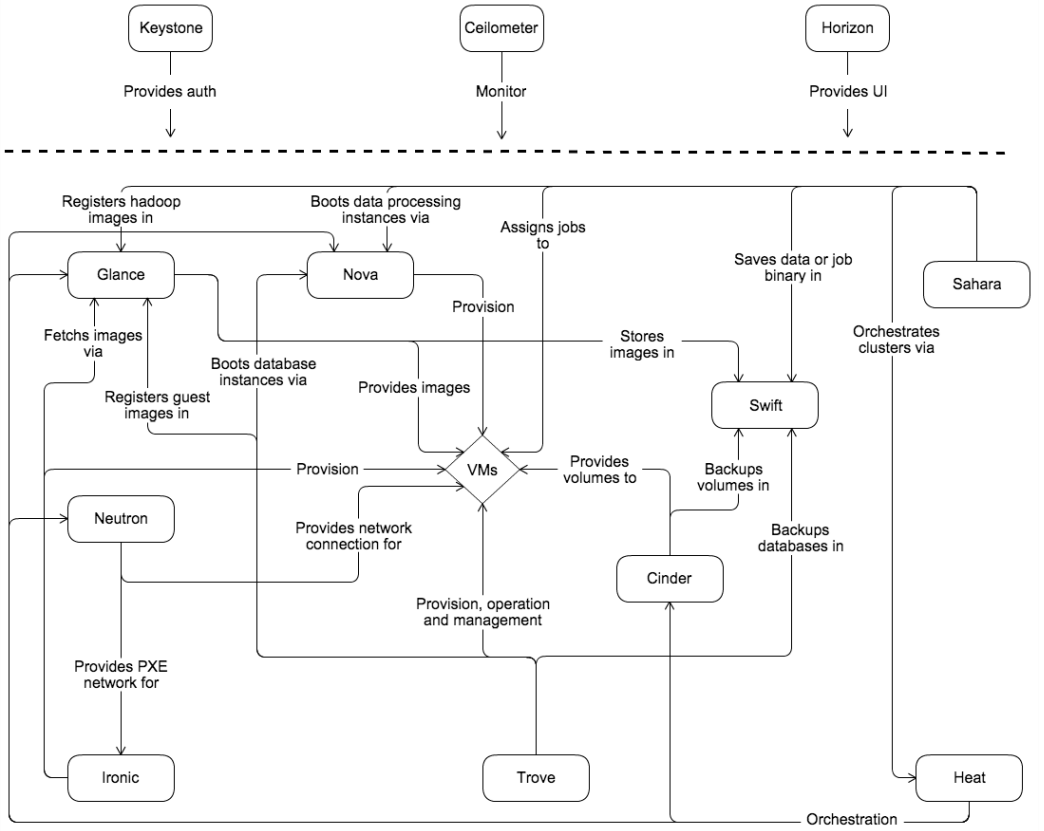

Ilustración 14. Diagrama de servicios

En cambio, esta es solo la arquitectura más básica que se puede entender. Para entender cómo funciona la nube de *OpenStack*, tendríamos el siguiente diagrama, que, aunque es muy común este funcionamiento, no es el único posible:

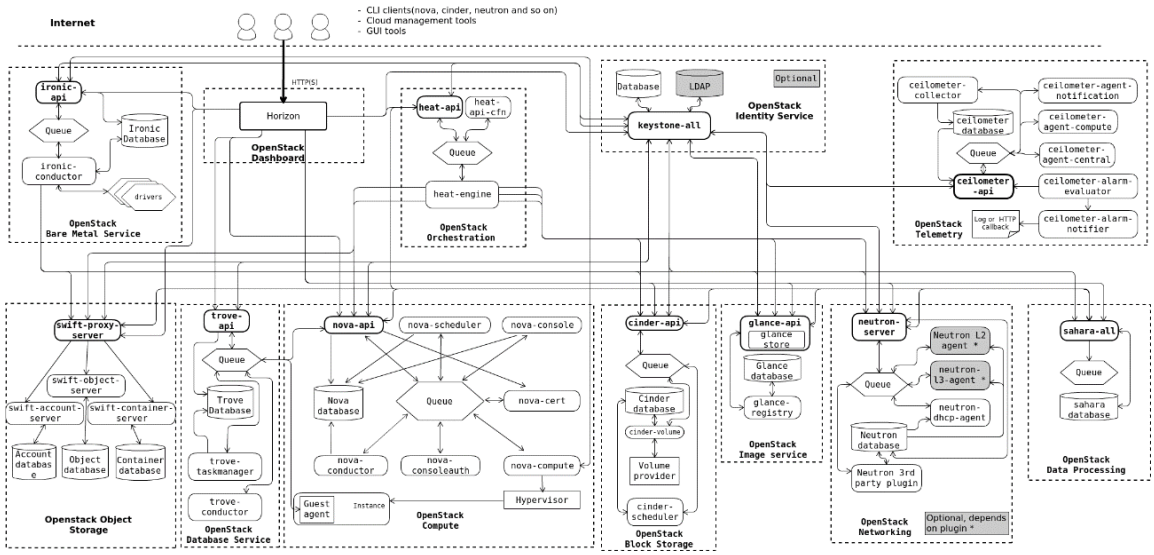

Ilustración 15. Diagrama ejemplo de un Cloud

Como se puede ver, cada servicio precisa de otros para funcionar, y, sobre todo, cada uno de ellos depende del más importante, *keystone*, sin este servicio no se podría realizar ninguna tarea dentro del *cloud*.

8. Implementación

Para la puesta en marcha de los servicios, se ha realizado una máquina virtual de ***Linux***, más concretamente el sistema operativo ***CentOS 7***, que es el mejor sistema operativo con el que se puede llevar a cabo la instalación de la nube. Hay unos pocos sistemas operativos que son compatibles con ***OpenStack,*** pero no son realmente buenos para su instalación, prácticamente todos tienen muchos errores de dimensionamiento y pueden dejar sin funcionamiento la máquina virtual principal. Uno de ellos es el ***Red Hat CentOS***, que es prácticamente el mismo sistema operativo que el utilizado para este proyecto, pero actualmente tiene soporte a errores, y como es habitual, conlleva un pago por estos servicios.

La implementación de OpenStack, ha sido probada e instalada en ***CentOS 8***, ya que este tiene soporte actualmente, pero viendo la cantidad de errores que siguen apareciendo en este nuevo SO, se ha decidido implementarlo en la versión anterior, como se ha nombrado anteriormente.

Una de las mejores formas de implementar este software no es realmente con una máquina virtual, si esta se instala en un ordenador personal, la mejor forma es haciendo una partición en el ordenador y tener instalado Linux en una de las partes.

Las mejores versiones de *OpenStack,* para la versión de Linux que se ha comentado, son *Rocky* y *Wallaby*, pero la que menos errores se han reportado es de la versión de *Rocky*, que es la que se ha elegido. Luego están las versiones más actualizadas como es *Yoga*, pero en la versión de *CentOS 7* no correría esta versión.

En los siguientes apartados se podrá ir viendo los requisitos del sistema y como configurar la máquina virtual para una buena ejecución, aunque de ella se hayan ido encontrando problemáticas a medida que se ha ido realizando el proyecto.

8.1 Requisitos del sistema

Como se ha comentado a lo largo de la memoria, la implementación se ha realizado en un portátil personal, que, aun teniendo buenas especificaciones, no es un entorno óptimo para alojar un *cloud* como tal.

Como se ha mencionado, es necesario un sistema operativo *Linux* para la implementación del software, por lo que se ha preparado una máquina virtual para ello.

Las especificaciones del sistema son los siguientes:

	Totales	**Disponibles (VM)**
Almacenamiento	1Tb	30 Gb
Ejecución	16 Gb RAM	10 Gb RAM
CPU	Intel i7 (10G)	Intel i7 (10G)
Grafica	NVIDIA 2060	NVIDIA 2060

Tabla 5 Especificaciones del sistema

Para un perfecto funcionamiento de las instancias lanzadas dentro de la nube, y que estas puedan ser lanzadas sin ningún error, es necesario un servidor con unas altas especificaciones, al contrario de lo que se dispone para este proyecto.

	Necesarios	**Disponibles**
Almacenamiento	100 Gb	30 Gb
Ejecución	32 Gb RAM	10 Gb RAM
Cloud	Servidor	PC personal

Tabla 6 Especificaciones del sistema necesarias

8.2 Problemática

Antes de explicar cómo se han implementado estos servicios, hay que comentar las limitaciones que se han ido encontrando en la puesta en marcha y que problemas implica en el desarrollo de este proyecto.

OpenStack, en base a su arquitectura, esta nos limita a la hora de crear máquinas virtuales, si este se encuentra instalada dentro de una máquina virtual, como es el caso de este proyecto.

Uno de los problemas que tiene *OpenStack* es que no permite la creación de máquinas virtuales anidadas, y como se ha podido ver, el servicio más importante que ofrece este software son la creación de las máquinas virtuales en la nube, si este está instalado de primeras en una máquina virtual, no permitirá la creación e inicio de instancias que requieran de muchos recursos, lo que dificulta la implementación de este software, y dejando solo la posibilidad de la instalación en un sistema operativo único.

A parte de esta limitación, podemos encontrar el problema de almacenamiento y procesado, si *OpenStack* comprueba que en el sistema no hay suficientes recursos para lanzar una máquina virtual, aunque en las especificaciones de esta hayamos puesto los mínimos recursos posibles y que estos concuerden con los recursos de la máquina virtual donde está instalada, *OpenStack* mandará un mensaje de error a la hora de la creación de la instancia.

Todo esto dificulta la tarea, pero no todo son cosas malas, *OpenStack* ofrece en su instalación una imagen de prueba de aproximadamente 12Mb con la que podemos probar la creación de instancias y lanzamiento de estas sin tener que utilizar imágenes muy grandes.

En este caso, se han encontrado estos problemas a la hora de la implementación, pero gracias a la ayuda de la imagen que ofrece OpenStack, se puede saber que la migración del desarrollo software de este proyecto se puede implementar en el despliegue de OpenStack en un servidor común o un Data Center.

8.3 Despliegue

Como se ha comentado anteriormente, se ha utilizado una máquina virtual de *Linux CentOs 7* para el despliegue de la plataforma, en los siguientes apartados se comentan los aspectos más importantes del despliegue de la máquina virtual y la posterior implementación del software.

8.3.1 Máquina virtual

Para la creación de la máquina se ha utilizado *VirtualBox,* con este software se pueden instanciar los recursos necesarios fácilmente.

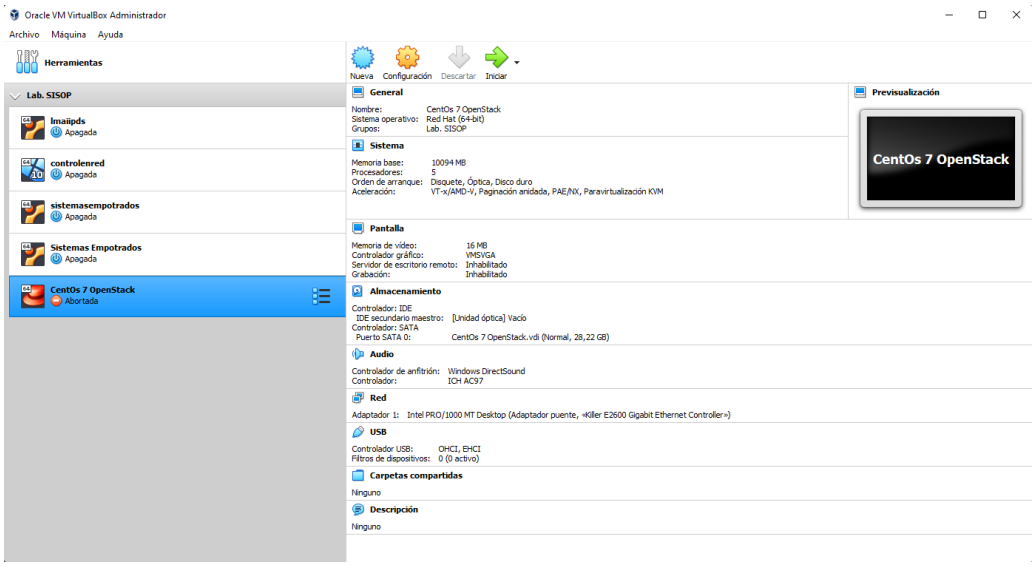

Ilustración 16 Entorno VirtualBox

Antes de nada, es necesario el archivo de imagen *.iso* de *CentOs 7* con el que se pone en marcha la máquina virtual, una vez se tiene el archivo se crea la máquina virtual.

Para ello, se han utilizado los requisitos que se han comentado anteriormente, ya que el ordenador que se ha utilizado no disponía de más recursos para ello, aunque, son suficientes para probar que el software funciona correctamente, y, por tanto, escalable a otro servidor con mejores recursos.

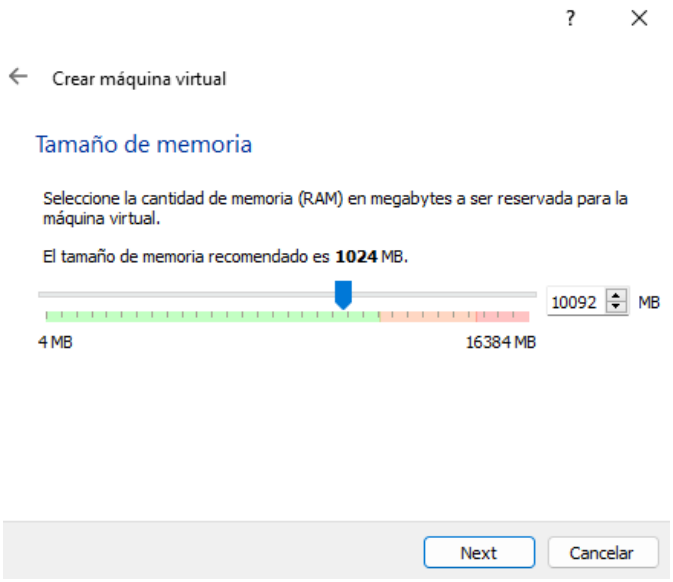

Ilustración 17 RAM de la máquina virtual

Una vez se tiene la máquina virtual con sus respectivos recursos, el paso más importante es la configuración red de la máquina. Esta debe estar dentro del rango del ordenador o servidor donde se instale, en este caso el ordenador utilizado tiene la siguiente IP:

192.168.100.4

Sabiendo esto, debemos poner el mismo identificador de red, en este caso la IP de la máquina es la siguiente:

192.168.100.163

Este es el paso más importante porque esta es la IP con la que vamos a acceder a *OpenStack* y vamos a realizar las diferentes peticiones en la plataforma que se ha desarrollado.

En cualquier otro caso, no se podrían realizar las peticiones a *OpenStack.*

Para ello configuramos la red de la máquina virtual con la opción de "Adaptador puente", teniendo así el mismo identificador red del computador principal.

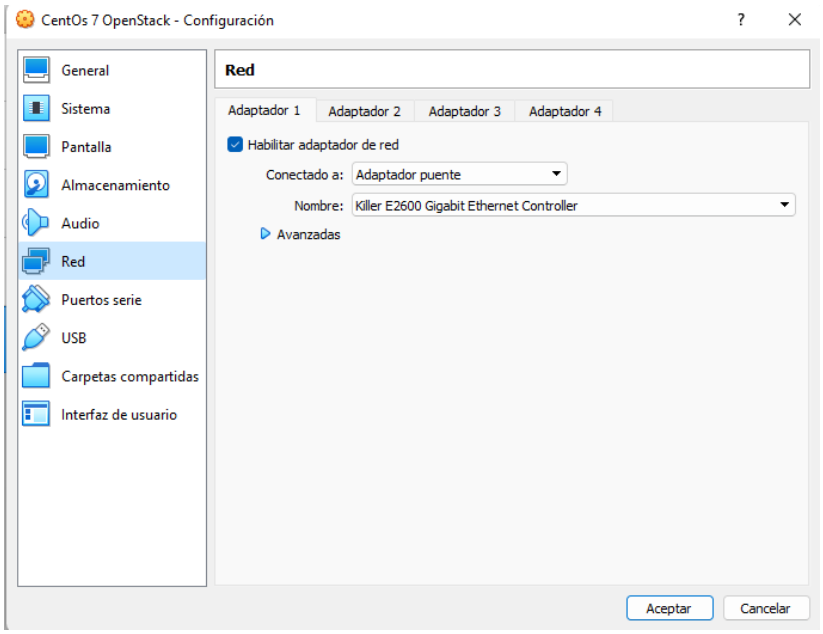

Ilustración 18 Opciones de Red de la máquina virtual.

Posteriormente, en la instalación del S.O, se fijará la dirección IP que se ha mencionado anteriormente.

Teniendo la máquina virtual prácticamente creada, las instancias que se creen y lancen en la nube, precisan de nombrar el número de CPU`s que se van a utilizar para mover las máquinas virtuales. En este caso se van a utilizar un máximo de 5 procesadores para no dañar el principal sistema operativo y el de la máquina virtual, así podrán seguir en funcionamiento todos los sistemas.

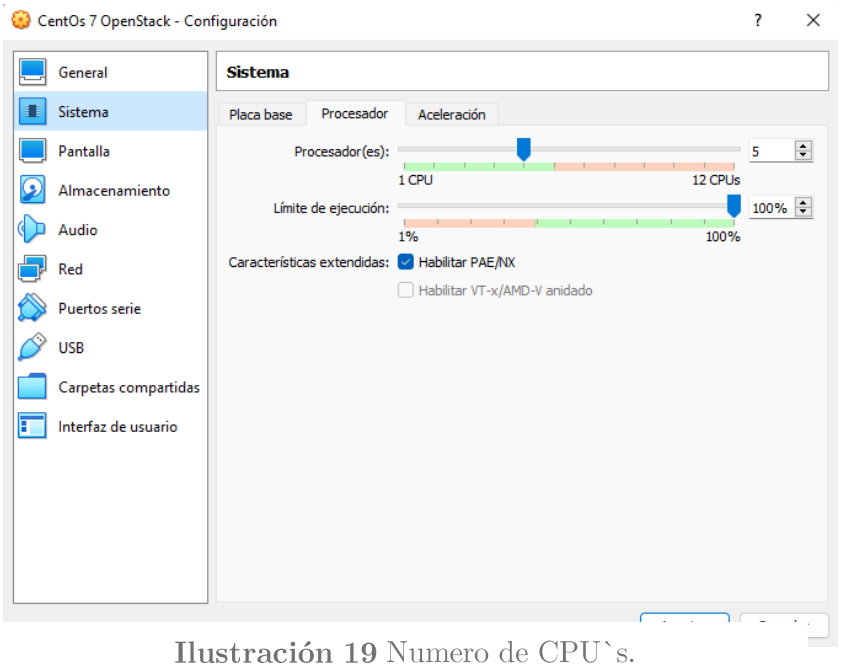

Ilustración 19 Numero de CPU`s.

Finalmente, solo sería necesaria la instalación de *CentOs* en la máquina virtual para luego desplegar toda la arquitectura de *OpenStack.*

8.3.2 Instalación CentOS

La instalación del sistema operativo es sencilla, ofrece una interfaz con diferentes opciones antes de la instalación completa del sistema, la parte más importante se encuentra en la configuración de la IP estática del sistema, en este caso se ha utilizado la IP nombrada en el apartado anterior.

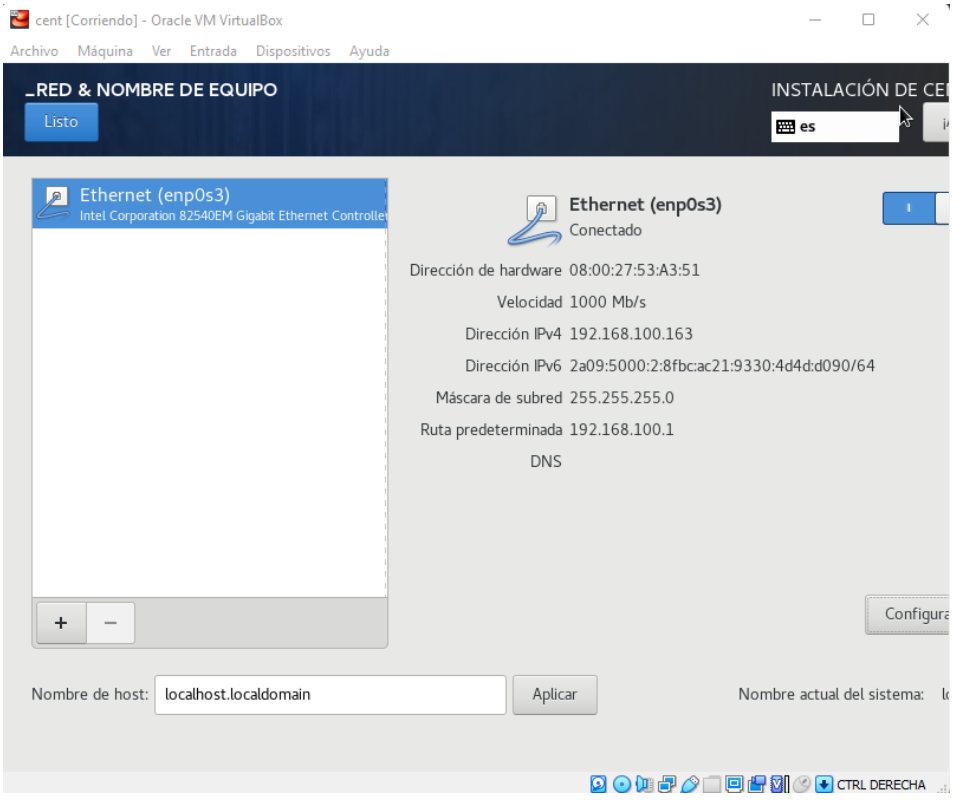

Ilustración 20 Configuración red máquina virtual

A partir de aquí, la instalación del S.O es sencilla, una vez terminada la instalación saldrá la terminal de Linux, desde aquí se realizará la instalación de *OpenStack Rocky.*

Por defecto el usuario y contraseña serian ***root***, esto es conveniente dejarlo tal y como esta, ya que para la instalación es

necesario los permisos *root* para ello, si creamos un usuario para acceder a la máquina virtual hay que poner el parámetro *sudo* en todas las llamadas o en consecuencia cambiar al usuario *root*.

8.3.3 Instalación OpenStack

Para que no haya problemas a medida que instalamos los servicios de *OpenStack*, una vez se tiene el *Linux* completamente instalado, hace falta una actualización del sistema, ya que este sistema operativo lleva un tiempo sin soporte, puede que la imagen que se ha utilizado esté desactualizada. Para ello utilizamos el siguiente parámetro y argumento.

```
$ yum update -y
```

Una vez instaladas las actualizaciones del sistema operativo, se deshabilita y se detiene el servicio de firewall por defecto.

```
$ systemcl disable firewalld
. . .
$ systemcl stop firewalld
```

Posteriormente se deshabilita el servicio de red por defecto, esto permite hacer todos los cambios de red necesarios para que la arquitectura de OpenStack cree una infraestructura de red estable y en el momento de crear cualquier red dentro de la plataforma desarrollada poder hacerlo sin errores. Primero se desactiva el configurador y luego se detiene el servicio.

```
$ systemcl disable NetworkManager
. . .
$ systemcl stop NetworkManager
```

Si no viene previamente instalado, se instala el paquete *network*.

```
$ yum install networks
. . .
```

Una vez instalado el paquete, habilitamos los servicios.

```
$ systemcl enable network
. . .
$ systemcl start network
```

A partir de este paso, ya se puede empezar la instalación sin problemas de *OpenStack*. En este paso se instalan las bases de datos y sistemas que ayudan a la lectura de información y guardado tales como las máquinas virtuales, imágenes de S.O, y estructura de almacenamiento. En este caso, como se ha ido comentando a lo largo del proyecto, se ha instalado la versión *rocky*, que es la mejor opción para este sistema operativo.

```
$ yum install -y centos-release-openstack-rocky
. . .
```

Para comprobar que todo ha ido bien, los más conveniente es actualizar otra vez el sistema.

```
$ yum update -y
```

Seguidamente, teniendo ya la arquitectura de *OpenStack* asentada en la máquina virtual, lo siguiente que se instala son los servicios de *OpenStack*.

Son los que dan vida a la nube, sin estos no se puede realizar la plataforma *cloud*. Para ello se va a utilizar el llamado *packstack*, este instala todos los servicios de una sola vez. Aunque luego no vayamos a utilizar alguno de los servicios, es conveniente instalarlos todos conjuntamente, ya que como se ha comentado anteriormente, cada uno de los servicios se alimenta de otro para su funcionamiento, si se instalan los servicios de forma separada, es posible que encontremos algunos errores en el desarrollo de la plataforma. De todas formas, es posible realizar la instalación por separado, pero habría que meterse en la infraestructura de OpenStack para comprobar que todos los nodos están bien hilados para que no haya problemas futuros.

En cualquier otro caso, si estuviéramos delante de un proyecto a gran escala, disponiendo de varios *Data Center*, lo mas conveniente es instalar todo por separado, en cualquier otro caso, lo más sencillo y que puede evitar cualquier problema, lo mejor es instalarlo todo de una sola vez.

```
$ yum install -y openstack-packstack
. . .
$ packstack --allinone
```

Una vez hechos todos los pasos, ya se ha terminado con la instalación de *OpenStack.*

Mientras esté la máquina virtual encendida, todos los servicios de *OpenStack* están activos. La primera vez que se apague la máquina, lo más seguro es que cuando se vuelva a encender no se activen automáticamente todos los servicios, para ello, se instala un paquete que los activa automáticamente cada vez que se encienda la máquina virtual.

```
$ yum install -y openstack-service
. . .
$ openstack-service start
```

También cada cierto tiempo, es conveniente lanzar un *status* de la plataforma para ver que todo está correcto.

```
$ openstack-service status
```

Con esto comprobamos que todo está activo y funciona correctamente. Este comando es muy utilizado si algún servicio deja de funcionar, nos indica que todo está activo, si de lo contrario algo ha dejado de funcionar, indica que el servicio esta inactivo. Si en algún momento encontramos que algún servicio ha dejado de funcionar por el motivo que sea, solo es necesario reiniciar los servicios con el paquete anteriormente nombrado.

```
$ openstack-service stop
```

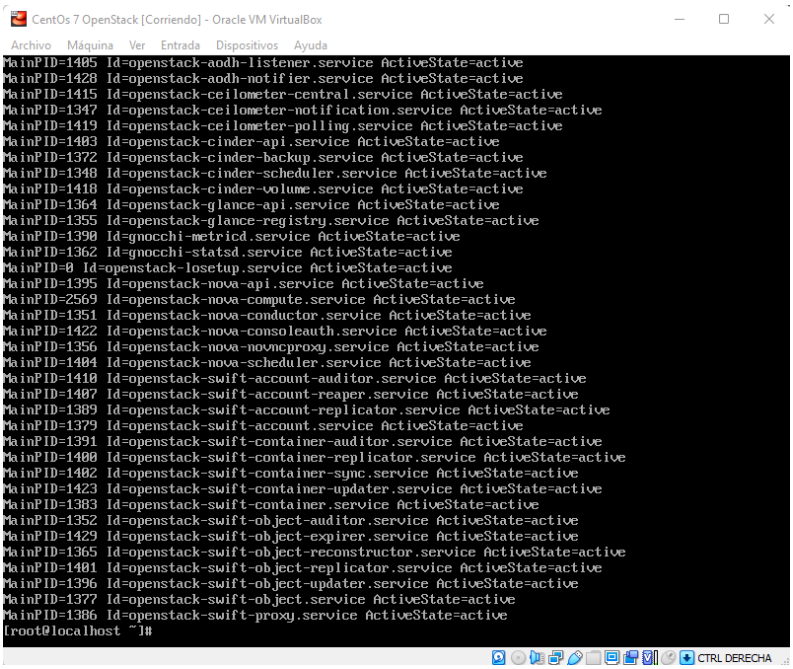

Ilustración 21 Comprobación de los servicios activos

Finalmente, ya se ha visto que todos los servicios están activos y listos para realizar el desarrollo del cloud. Teniendo todos los servicios activos podemos comprobar también que *OpenStack* funciona correctamente, podemos acceder al servicio *horizon* desde un navegador web con la IP configurada de la máquina virtual y el número de puerto donde está alojado.

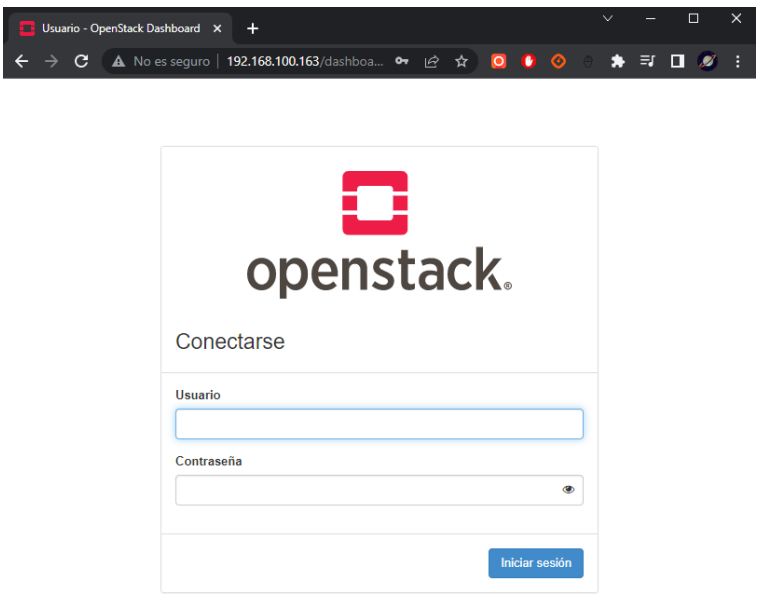

Ilustración 22 Servicio horizon

Para acceder a este servicio será necesario un usuario y contraseña. Para ver estas credenciales accedemos a un archivo de texto alojado en la raíz de root de la máquina virtual.

```
$ ls -l
. . .
$ cat keystone_admin
```

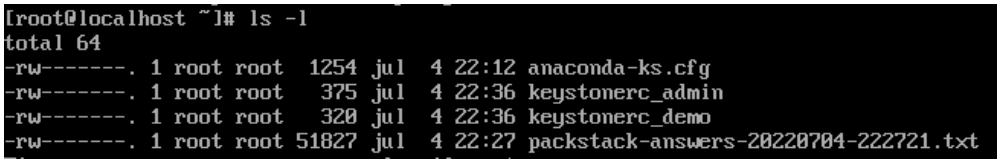

Ilustración 23 Archivos OpenStack

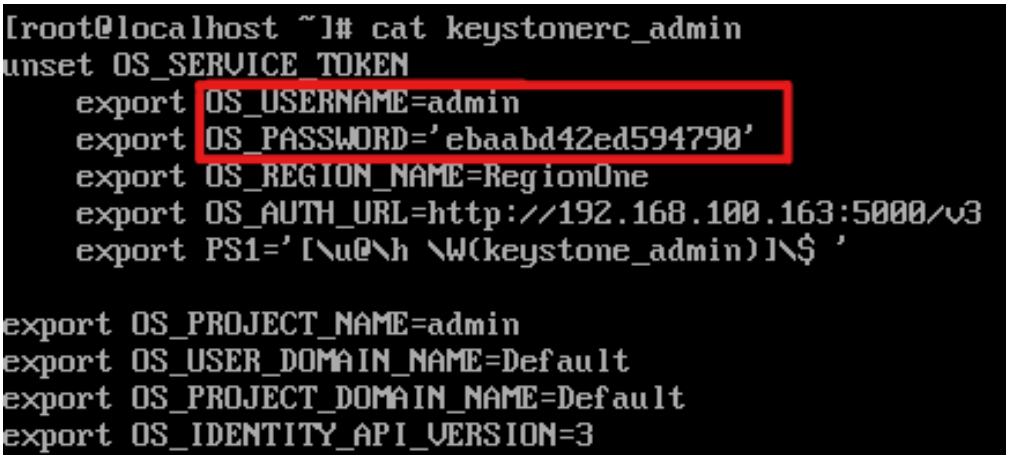

Ilustración 24 Credenciales para el acceso a OpenStack

Con estas credenciales podemos acceder a *horizon* y crear cuentas y proyectos para los usuarios que vayan a acceder a la plataforma *cloud*. En este caso, no se han creado otros usuarios ya que el *cloud* se va a utilizar de forma privada.

En el caso del servicio *horizon*, no se va a utilizar ya que necesita de muchos recursos de la máquina virtual para su funcionamiento, esta realiza todas las peticiones a *OpenStack* a la vez una vez se entra, es por eso se ha desarrollado un *cloud* a parte para poder optimizar todos los recursos que se disponen.

En el siguiente apartado se explica detenidamente el desarrollo de la plataforma cloud.

9. Desarrollo Cloud

Actualmente, nos encontramos en un momento incierto en cuanto a recursos de hardware se refiere.

La gran demanda en cuanto al hardware, ocasionado por la escasez de semiconductores y por la inflación, muchas empresas han apostado por plataformas *cloud* donde guardar información de valor y poder acceder a ella de manera sencilla.

El objetivo final de este proyecto es desarrollar una plataforma que pueda satisfacer esas necesidades de muchas empresas en el sector industrial.

Se ha desarrollado mediante *Django* una plataforma relativamente básica que satisfaga esas necesidades y que lo haga de una manera intuitiva y sencilla, con la que se puede acceder desde cualquier dispositivo conectado a internet desde cualquier parte.

En los siguientes apartados se podrán ver los pasos seguidos para el desarrollo de esta plataforma basada en *OpenStack*, los lenguajes que se han utilizado y el resultado final de este.

Finalmente habrá un apartado de mejoras que se podrían realizar en la plataforma, si esta llegara a venderse a gran escala.

9.1 Repositorio GitLab

Una de las partes más importantes para el desarrollo de cualquier plataforma o software es la creación de un repositorio donde ir almacenando el código creado y evitar perderlo por cualquier circunstancia. También sirve si nos encontramos en un entorno de trabajo con varios compañeros que necesitan el código para desarrollar la parte asignada.

El repositorio GitLab es de los más importantes y famosos que existen, se podría considerar una *cloud* donde almacenar código.

Según se especifique en el proyecto creado, podemos ponerlo de manera pública o privada. Un proyecto público puede contribuir cualquier persona que tenga una cuenta en Git y ayudar al desarrollo de ese proyecto, en este

caso se ha creado un proyecto privado, en el cual solo el propietario puede acceder al código, aunque se le pueden dar permisos a otros usuarios, pero no es este caso.

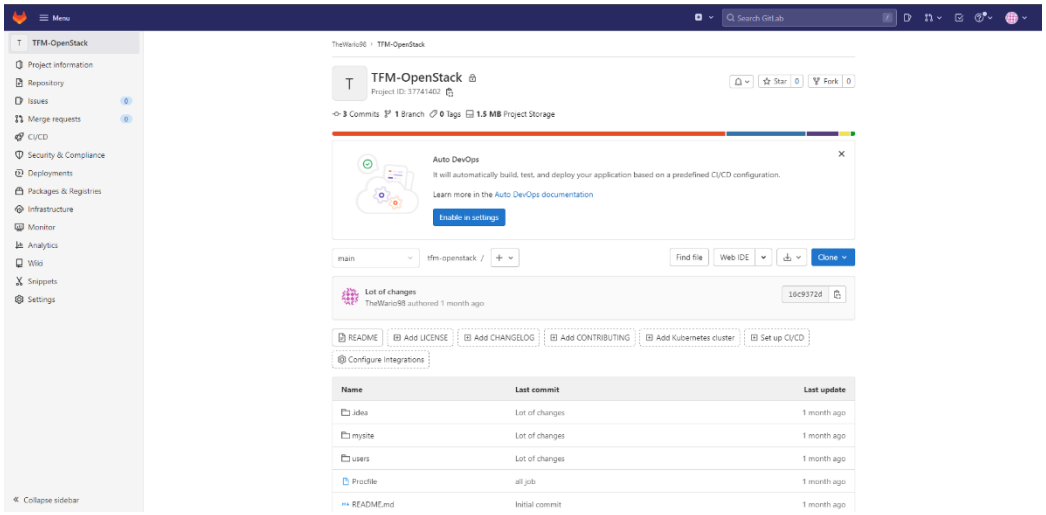

Ilustración 25 Repositorio GitLab

Primeramente, se instalan los paquetes necesarios de *Git* para clonar el repositorio.

Una vez se ha creado el proyecto e instalado *Git*, se ha colocado una carpeta en local en el ordenador donde se ha clonado el repositorio de *Git*, presionando el botón derecho del ratón en la carpeta donde clonaremos el proyecto, podemos encontrar el terminal de *Git*.

Teniendo todo preparado, en primer lugar, se ha creado una clave SSH con la que mantener la seguridad en el proyecto, para ello, en la terminal de Git, se escribe lo siguiente:

> $ ssh-keygen -t rsa -b 2048 -C “EMAIL”
> . . . (Se escribe la contraseña deseada)

Si todo ha ido bien, saldrá por la terminal una especie de clave encriptada.

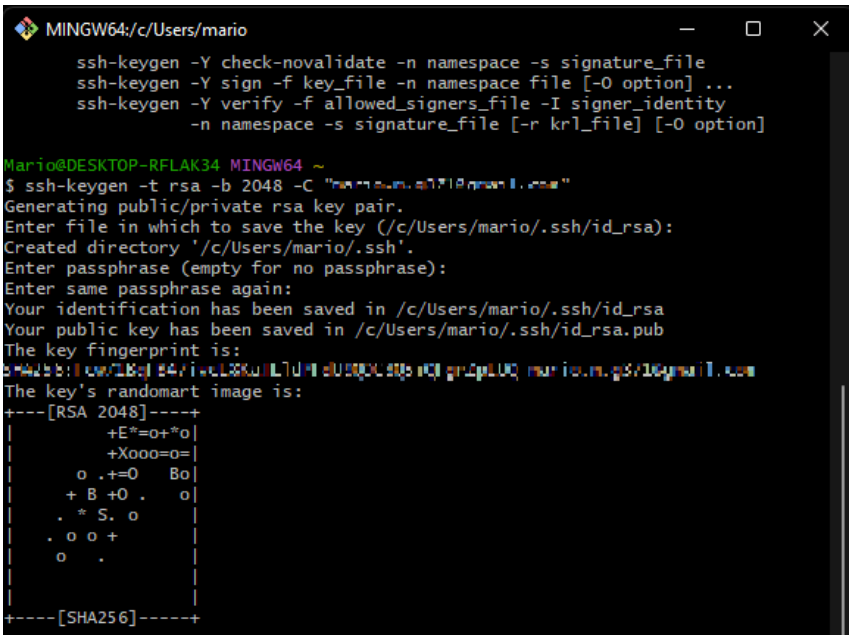

Ilustración 26 Clave SSH Git

Esta clave aparece en la carpeta que especifica la terminal, en este caso llamado *id_rsa,* se abre con Notepad y se copia para luego pegarla en la plataforma de *GitLab.*

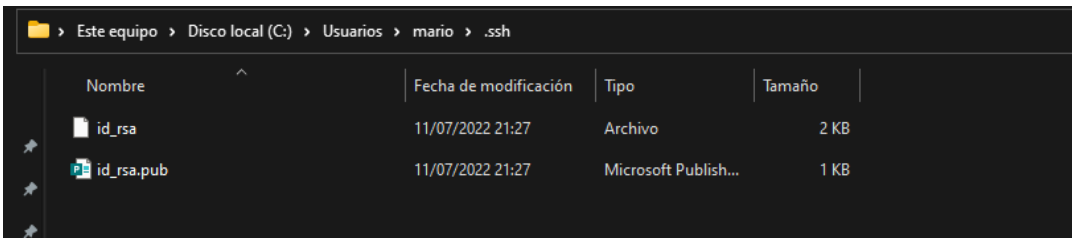

Ilustración 27 Clave rsa

Una vez hechos todos los pasos, solo queda clonar el repositorio mediante la siguiente combinación en el *bash* de *Git.*

Finalmente, dentro de esta carpeta, se almacena el proyecto que se ha desarrollado.

Mientras se ha ido desarrollando el proyecto, se han realizado subidas a la plataforma de *GitLab* para evitar la pérdida de datos en local, aunque no se haya creado una rama en el proyecto, esta se ha ido almacenando en la rama principal mediante los siguientes comandos:

```
$ git commit -m "TEXTO"
. . . (Se resuelven conflictos si los hay)
$ git push origin main
```

Así se han ido subiendo el código al repositorio para tenerlo a buen recaudo.

9.2 Plataforma Cloud

La plataforma se ha desarrollado mediante *Django,* ha permitido realizar una plataforma web en dos planos, mediante el *Front-end*, que es lo que el cliente sencillamente ve, y escrito mediante *JavaScript, HTML* y *CSS*, y el *Back-end*, que es la parte de código donde se realizan las peticiones a *OpenStack,* todo escrito en lenguaje *Python.*

Con esto se ha conseguido, aparte de no ser una web estática, poder convertir una simple página web en un cloud funcional.

La plataforma se ha dividido en varias pestañas dependiendo de los diferentes servicios de *OpenStack*, llevando así, una estructura modular.

El cloud se ha dividido en los siguientes módulos:

- **Log in**: autenticación con el servicio *Keystone*
- **Dashboard:** pestaña para visualización de recursos utilizados
- **Instancias:** servicio de máquinas virtuales.
- **Redes:** visualización de redes mediante el servicio *Neutron*
- **Routers:** visualización de routers mediante el servicio *Neutron*
- **Imágenes:** visualización de imágenes de disco en la plataforma mediante *Glance*
- **Sabores:** visualización de recursos de máquina utilizados

9.2.1 Login

Como toda plataforma segura, es necesario una autenticación, por ello la primera pestaña es el login al *cloud* privado.

Anteriormente se ha comentado que el servicio que ofrece esta autenticación es *keystone*, por lo que se ha realizado un *front-end* acorde con las credenciales principales para obtener el token de autenticación. Este token se obtiene llevando las credenciales al back y hacer la petición a *OpenStack*.

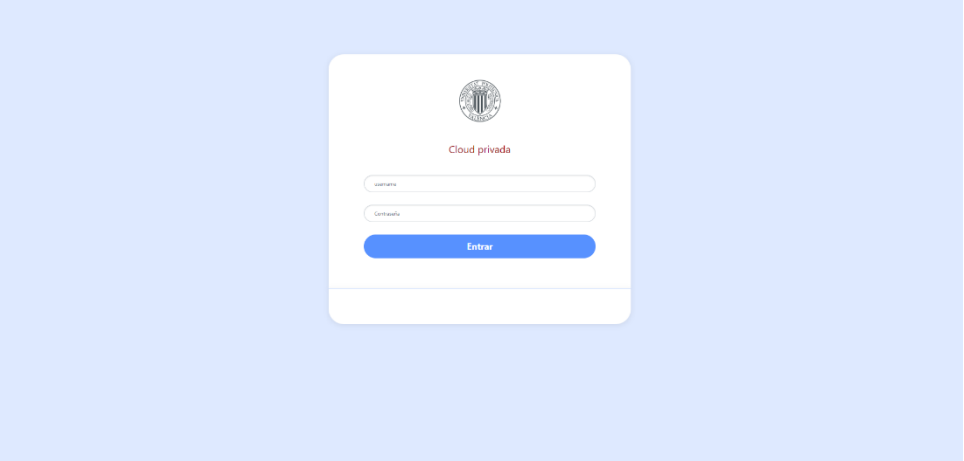

Ilustración 28 Login cloud privada

En el caso de tener varios proyectos, se añadiría un campo input adicional para el nombre del proyecto al que pertenece el usuario, pero para estos casos la plataforma seria robusta y podría aguantar muchas máquinas virtuales encendidas simultáneamente.

Como cualquier login presenta un mensaje de error si las credenciales son erróneas y devuelve a la misma pestaña al usuario para volver a escribirlas.

A parte, como cualquier software que se precie, es muy importante que tenga la función para verse en dispositivos móviles, ya que esto nos permitiría ver la plataforma desde cualquier parte.

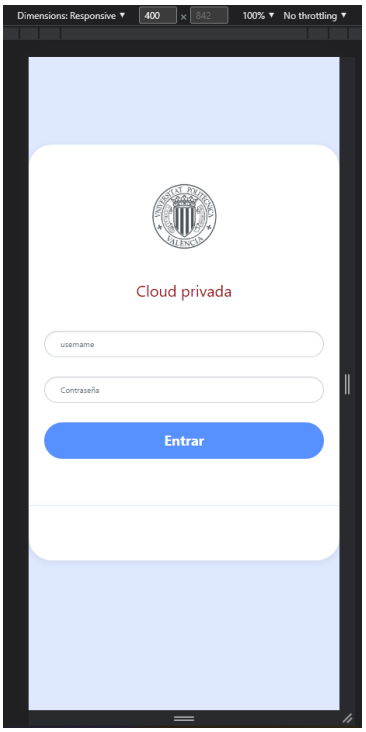

Ilustración 29 Login responsive

9.2.2 Dashboard

Una vez se entra a la plataforma, lo primero que se observa es el *dashboard*. En esta pestaña podemos visualizar los recursos totales que se están utilizando al momento comparados con los totales que dispone la máquina virtual donde está instalada el *cloud*, así se evitan problemas de recursos y que la máquina colapse.

Se pueden ver una serie de gráficos, en los cuales se visualizan los siguientes recursos:

- **CPU`s**: número de CPU`s utilizadas
- **RAM**: cantidad de memoria utilizada en MB
- **Discos**: número de discos de volumen utilizados
- **Instancias**: número de máquinas en marcha
- **Volumen**: total de GB ocupados

A parte, se visualiza un diagrama donde se puede ver la topología red del proyecto en cuestión proveniente del servicio *horizon*.

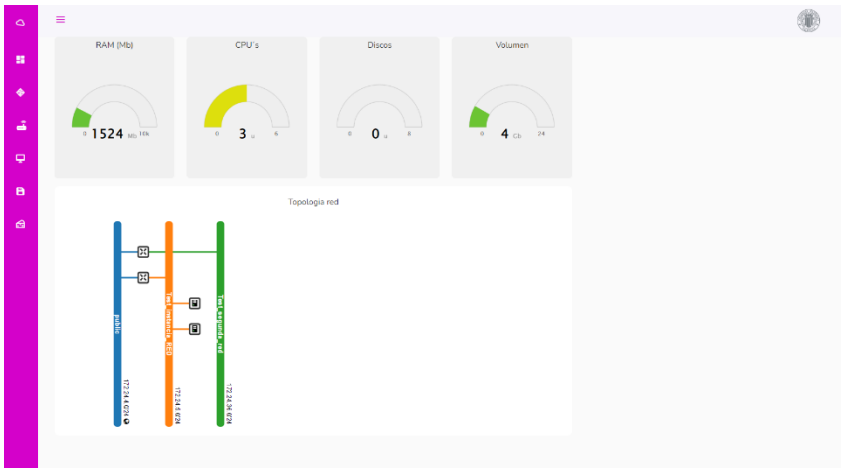

Ilustración 30 Dashboard

Como el apartado anterior, se ha hecho totalmente *responsive* para que se adapte a cualquier dispositivo en el que se lance la plataforma.

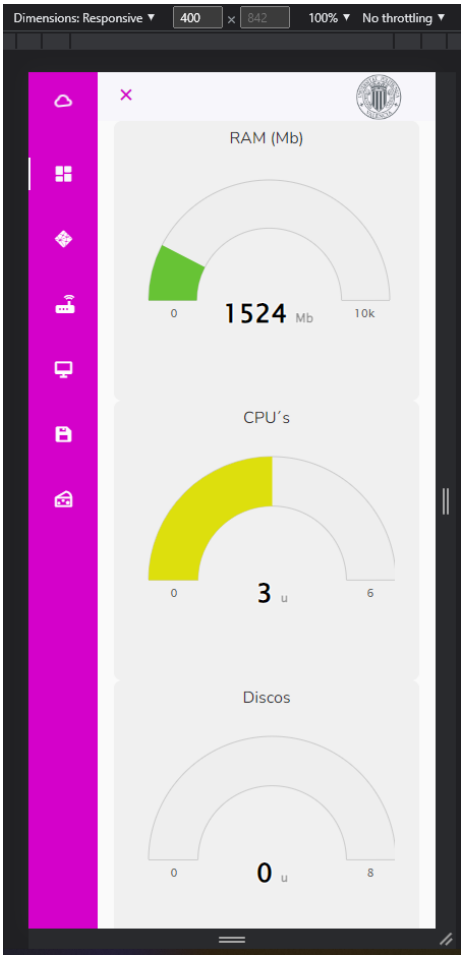

Ilustración 31 Dashboard responsive

9.2.3 Redes

Una parte importante para la creación de máquinas virtuales son las **redes.** Son indispensables a la hora de crear instancias ya que estas necesitan de una ip en específico.

Al crear una instancia esta se le asigna una subred, y esta automáticamente le añade una ip a la máquina.

En esta pantalla se pueden visualizar la lista de redes que se han creado con un buscador en la esquina superior derecha y paginación en el caso de tener muchas redes. Como no se suelen tener muchas redes, se ha optado por poner un botón de borrado en cada red.

En esta tabla se visualizan los siguientes parámetros de cada red, son los más importantes a la hora de crear otras cosas en la plataforma:

- **Id** de las redes.
- **Estado de la red**: Activado o desactivado.
- **Fecha de creación**.

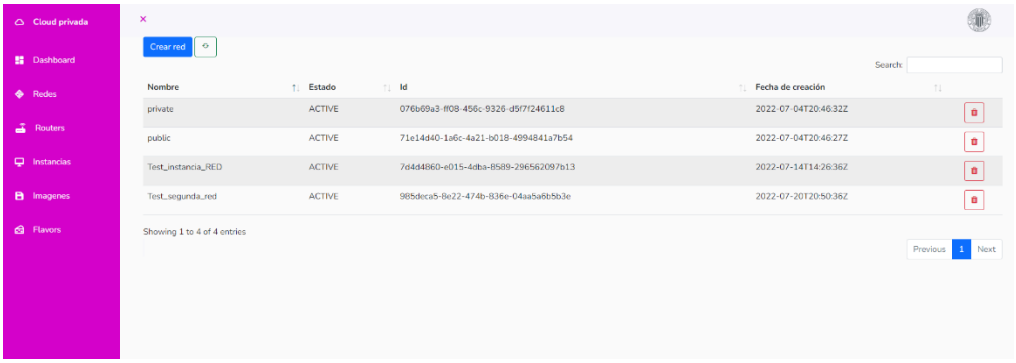

Ilustración 32 Pestaña redes

En la parte izquierda están los botones de creación de red y de actualización de la tabla.

Si se presiona el botón de crear red este abre una ventana de diálogo con los parámetros más importantes a la hora de creación de una red. Estos son los parámetros mínimos que necesita *OpenStack* para crear una red.

En esta ventana hay que incluir en los campos input el nombre de la red, el nombre de la subred, la ip de la subred con la máscara correspondiente (suele ir de 21 hasta 31) y la puerta de enlace.

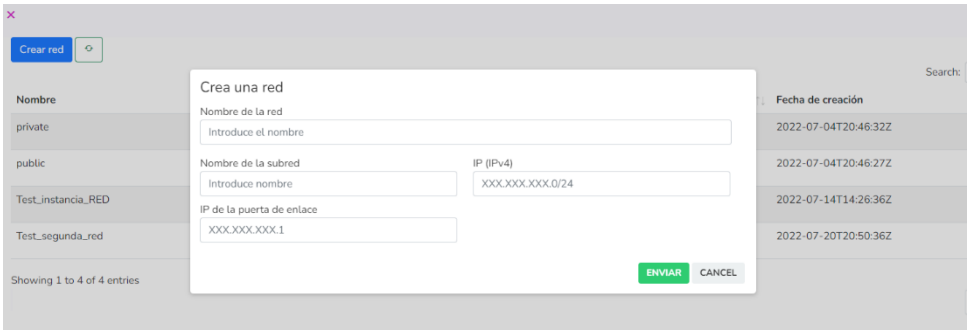

Ilustración 33 Ventana creación red

Y como cualquier pantalla está adaptada a cualquier dispositivo.

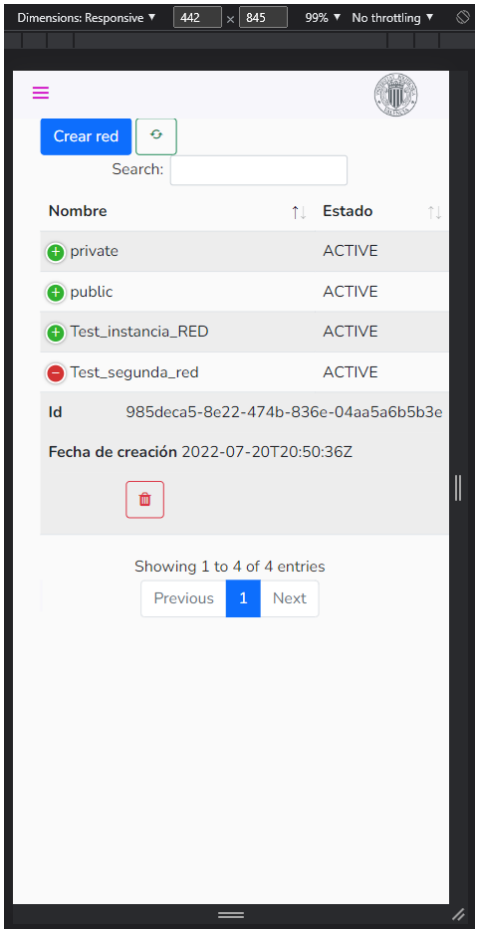

Ilustración 34 Pestaña redes responsive

9.2.4 Routers

Los routers son los elementos que conectan la red externa por defecto, que es la red donde está conectada la máquina madre, con las redes que se crean en la pestaña anterior.

Esta pantalla es parecida al de redes, se listan todos los routers creados con sus respectivos parámetros más importantes:

- **Nombre** del router
- **Estado**: activo o desactivado
- **Id** del router
- **Ip** del router
- **Id** de la subred conectada
- **Fecha de creación**.

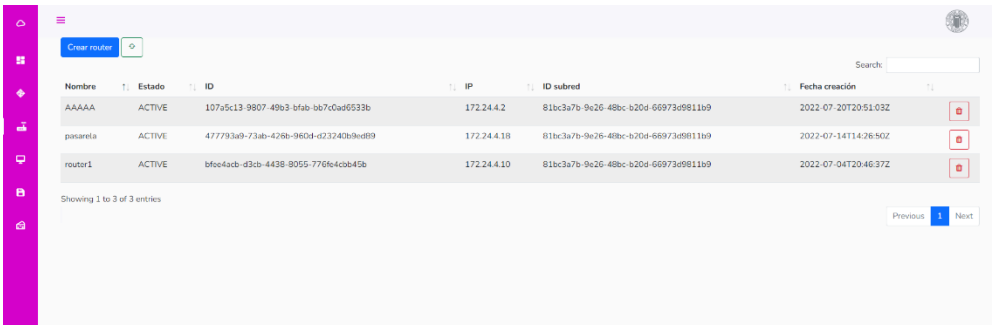

Ilustración 35 Pestaña routers

Para la creación de los routers solo es necesario el nombre de este, la id de la red externa (es necesaria indicarla ya que se pueden crear varias externas), y la subred de cualquiera de las redes creadas.

Estos datos aparecen automáticamente al abrir la ventana de diálogo haciendo una llamada a la base de datos por defecto de *OpenStack*, haciendo así una forma más visual la creación del router. De otra manera necesitaríamos la Id de las cosas que se han comentado, pero así, con unos simples selectores tendríamos la información.

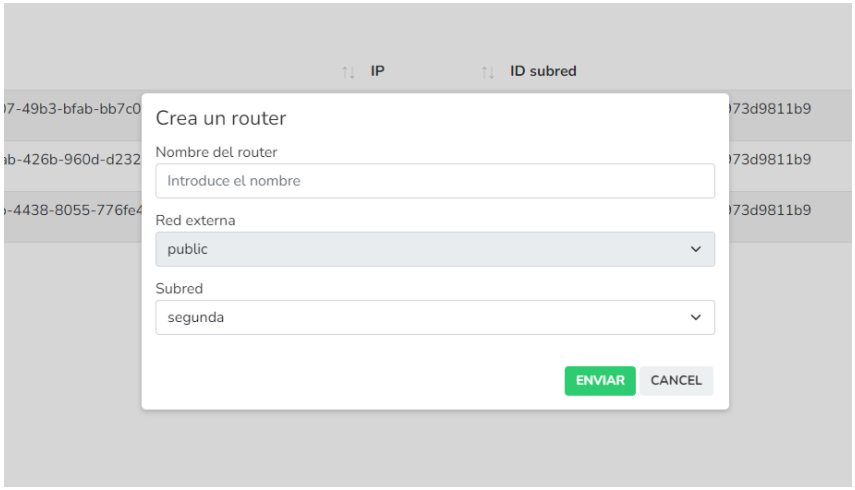

Ilustración 36 Ventana diálogo creación router

Y como se ha visto anteriormente, igual que las demás pantallas esta ofrece el estilo responsive para otros dispositivos.

9.2.5 Imágenes

Como cualquier máquina virtual, esta necesita de una imagen de S.O para arrancar, por lo que, *OpenStack* ofrece el servicio *glance* donde poder almacenar mediante base de datos los archivos de disco para arrancar estas máquinas.

Para ello, como las demás pantallas creadas, la pantalla de imágenes visualiza todas las imágenes almacenadas en la nube mediante una tabla, donde se visualizan los atributos más importantes de ellas:

- **Nombre** de la imagen
- **RAM mínima** que necesita la máquina para utilizar la imagen
- **Estado**: activo o en cola
- **Id** de la imagen
- **Visibilidad** de las imágenes
- **Formato** de las imágenes

El campo de estado de la imagen indica si la imagen ha sido aceptada por la plataforma o no, si esta ha sido aceptada esta aparecerá con el atributo “Activa”, si por el contrario la imagen se ha subido, pero no se ha aceptado manualmente esta aparecerá en el estado “En cola” a la espera de aceptarla.

Glance permite compartir imágenes entre proyectos y usuarios dentro del *cloud,* para ello la *API* devuelve un atributo en específico que indica la visibilidad que la imagen tiene sobre los usuarios que entran a la plataforma:

- **Public**: La imagen la pueden visualizar todos los usuarios dentro del *cloud.*
- **Shared**: Indica que la imagen es privada, pero ha sido compartida entre usuarios.
- **Private**: La imagen es privada, únicamente la puede utilizar el usuario que la creó.

Esta pantalla es muy parecida a las demás ya que, como todas, visualiza como una base de datos las imágenes que hay almacenadas en ella.

Ilustración 37 Pantalla de imágenes

Los botones situados en la parte superior izquierda permiten crear y subir la imagen de archivo, aceptar las imágenes entrantes compartidas por otros usuarios y activarlas, y el último, actualizar la tabla. El último botón es necesario ya que la tabla no se puede actualizar automáticamente, la petición de aceptar imágenes tarda unos segundos en realizarse ya que esta petición necesita de varios servicios de *OpenStack* para realizarse.

El cuadro de diálogo para crear y subir la imagen es la siguiente pide para realizar la petición a OpenStack los siguientes parámetros:

- **Nombre** de la imagen
- **RAM** mínima
- **Almacenamiento** mínimo que necesita la imagen para ser implantada
- **Formato** de la imagen
- **Archivo imagen**

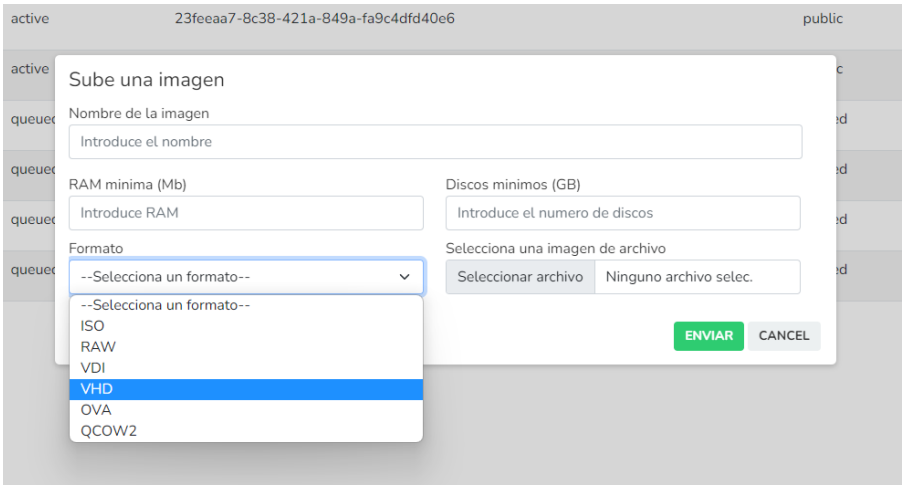

Ilustración 38 Cuadro de diálogo creación de imagen

Para el botón de aceptar imagen solo es necesario un campo input donde introducir la Id de la imagen entrante:

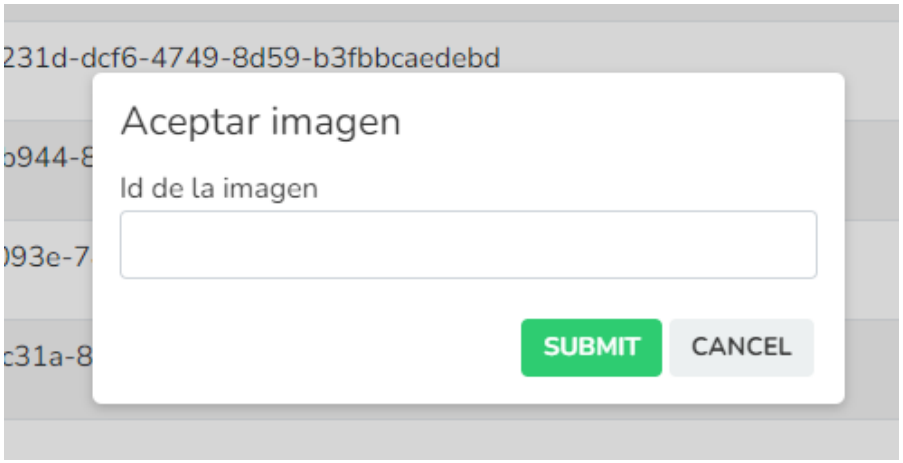

Ilustración 39 Botón aceptar imágenes

Como cualquiera de las demás pantallas, es adaptable a otros dispositivos.

9.2.6 Sabores

Uno de los parámetros más importantes para la creación de instancias son los ***flavors*** o sabores. Estos se utilizan para indicar los recursos mínimos y máximos que va a disponer la máquina virtual.

Siguiendo la dinámica de pantallas anteriores, si visualizan los datos más importantes de todos los sabores en una tabla:

- **Nombre** del sabor
- **RAM**
- **CPU`s**
- **Discos**
- **Id** del sabor

En el caso de esta pestaña, como los sabores solo son parámetros numéricos no necesitan de mucha memoria dentro de *OpenStack*, pudiendo así crear muchos sabores y almacenándose en la base de datos. Sabiendo esto, se han colocado *checkbox* en cada sabor para poder realizar un borrado múltiple, y también, botones de edición de cada sabor y borrado único.

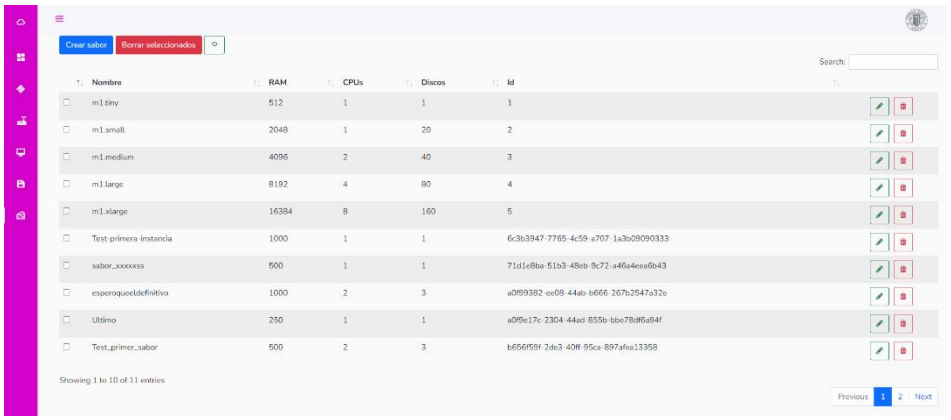

Ilustración 40 Pantalla de sabores

El cuadro de diálogo solo pide los parámetros importantes del sabor a crear. El botón de editar situado en cada fila de la tabla realiza la misma función que el botón de crear.

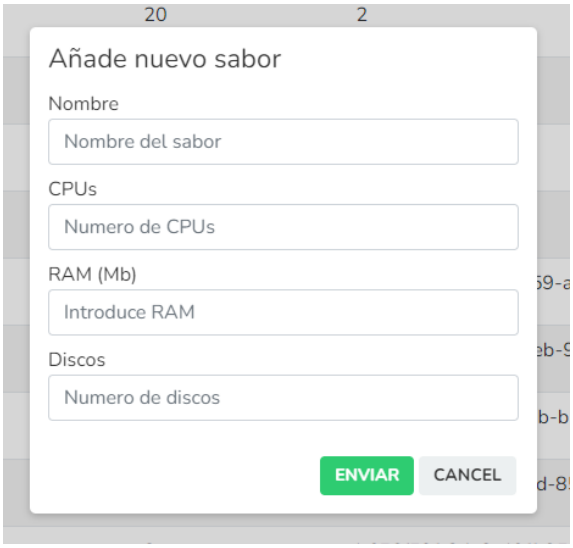

Ilustración 41 Cuadro de diálogo crear sabor

9.2.7 Instancias

Las instancias son el objetivo principal en la programación de este proyecto, sin las anteriores pantallas no se podrían crear máquinas virtuales.

Al poder crear varias instancias, estas se pueden visualizar en una tabla como los demás parámetros anteriormente vistos. En esta tabla se visualizan los siguientes datos:

- **Estado** de la máquina
- **Id** de la máquina
- **Fecha de creación**

Según el estado de la máquina podemos saber si está encendida o apagada, por lo que, si está apagada, se ha colocado un botón de encendido en la parte derecha de cada instancia junto con el botón de borrado.

Una vez la máquina esté en estado activo, el botón de encendido cambiará a visualización de máquina, para visualizar la terminal de esta bastará con pulsar este botón.

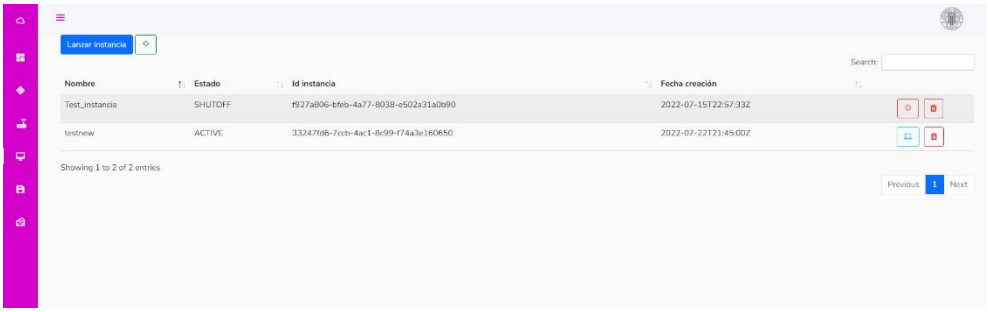

Ilustración 42 Pantalla de instancias

Según la problemática comentada en apartados anteriores solo se han podido crear instancias con la imagen llamada "cirros", esta imagen la trae por defecto *OpenStack*. Esta imagen al ocupar solamente alrededor de 10Mb, no implica una gran capacidad de procesamiento en la máquina virtual donde se ha instalado *OpenStack,* por eso es buena para hacer pruebas.

Para lanzar una máquina virtual se ha programado un cuadro de diálogo parecido a los anteriores, en este se visualizan los parámetros mínimos necesarios para la creación de estas. Se han colocado varios selectores donde se visualizan todas las imágenes almacenadas en la nube, todos los sabores, las redes creadas y la Ip que se le va a asignar. Todos estos selectores hacen una llamada a la base de datos del *cloud* para visualizarlos.

Aparte, según la imagen seleccionada en el selector, se calcula automáticamente el mínimo de almacenamiento necesario para la creación de la máquina.

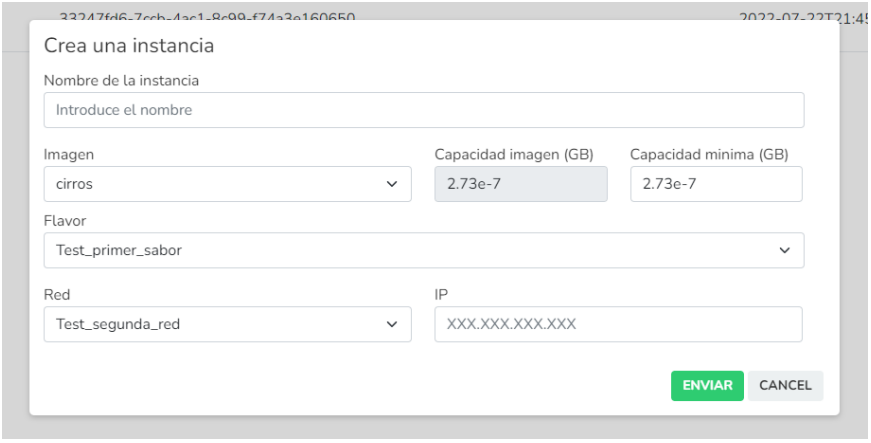

Ilustración 43 Cuadro de diálogo creación de máquina

Una vez se crea la máquina virtual, como se ha explicado anteriormente, solo queda encenderla. Una vez encendida, esta abre una pantalla donde se visualiza la terminal del ordenador. En este caso utilizando la imagen "cirros" no se puede hacer gran cosa, pero migrando la plataforma a un servidor con mejores recursos, podrá mover cualquier sistema operativo.

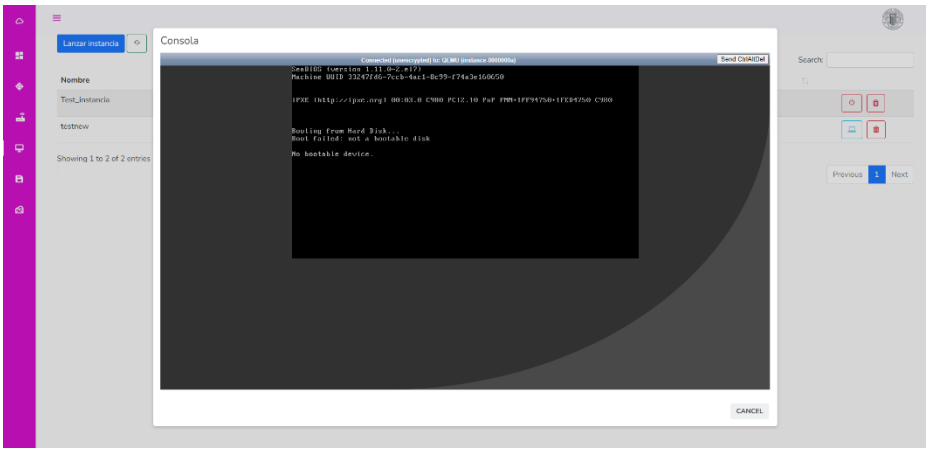

Ilustración 44 Consola máquina virtual

Cabe recalcar que para visualizar la consola en cualquier dispositivo habría que tenerlo de forma horizontal para ver la terminal completamente.

9.3 Mejoras futuras

La plataforma desarrollada realiza las funciones más fundamentales para la creación y utilización de máquinas virtuales en la nube, pero esto puede ir más allá.

Existen miles de funciones dentro del ecosistema de *OpenStack* a parte de las ya implementadas en esta plataforma que pueden ser de utilidad según las funciones que se le quieran dar a la plataforma:

Funciones	Descripción
Ip´s flotantes	*OpenStack* ofrece la opción de utilizar una dirección IP flotante creada a partir de la subred de la red externa. Una "IP flotante" es una dirección IP que puede añadirse dinámicamente a una instancia virtual en ejecución.
Grupos de seguridad	Es una colección nombrada de reglas de acceso a la red que se utilizan para limitar los tipos de tráfico que tienen acceso a las instancias.
Aplicaciones	Este servicio ofrece la creación de instancias con aplicaciones en específico para cualquier ámbito. Si se necesita un software de programación de PLC´s por ejemplo, este servicio crea la máquina virtual con el software ya instalado.
Secretos	Los secretos son básicamente elementos singulares o llaves utilizadas para almacenar cualquier tipo de cosa, tales como:

	• Llaves privadas • Certificados • Contraseñas • Llaves SSH
Clusters	Un clúster es un grupo de objetos lógicos, puede contener cero o más nodos. Un clúster tiene una propiedad "profile_id" que especifica qué perfil se debe utilizar cuando se crean nuevos nodos como miembros del clúster.
Firewalls	Proporcionan protecciones adicionales a los routers creados anteriormente.
Par de claves	Estas descargan un par de claves *OpenSSH* para usarlas como acceso a los servidores ya creados.

Tabla 7 Funciones adicionales

Como estas funciones, existen muchas más para poder implementar en la plataforma.

Dependiendo de las necesidades del cliente se pueden usar más o menos, ya que, cuantas más funcionalidades se implementen, más recursos consumirá. Si el cliente no va a utilizar alguna de esas funcionalidades, no merece la pena añadirlas ya que quitan capacidad de procesamiento que puede ser utilizado para cosas más importantes.

Por último, una gran mejora sería realizar un diagrama "*drag and drop*" con el que crear las instancias, redes, routers y demás elementos de una forma más gráfica utilizando el diagrama de topología red de una forma no estática. *OpenStack* permite la realización de estas funciones, pero hay que tener conocimientos amplios del lenguaje *Angular*.

10. Conclusiones

Por lo general, gracias al desarrollo de este proyecto se han adquirido conocimientos amplios sobre temas de redes y *cloud,* también habilidades sobre programación con *API REST* y distintos lenguajes de programación tanto para Front-end como Back-end.

Se han podido cumplir todos los objetivos propuestos satisfactoriamente, la implantación de OpenStack en un ordenador personal, creación de máquinas virtuales y el desarrollo de un cloud donde realizar estas tareas de una manera sencilla para empresas.

Entre estos objetivos cumplidos, uno de ellos, es de los más importantes para la sociedad, y es el impacto positivo que puede tener esta plataforma en el **ámbito medioambiental**, el cual se pueden reciclar dispositivos con pocos recursos electrónicos para frenar el cambio climático. Cualquier dispositivo con acceso a internet se puede utilizar esta plataforma y evitar la compra de nuevos dispositivos.

Se han encontrado algunos problemas a medida que se ha desarrollado el proyecto, pero se han ido solventando a lo largo de este. Uno de los problemas más importantes ha sido la falta de recursos en la implementación de *OpenStack*, pero finalmente se han logrado todos los objetivos y se ha conseguido que la plataforma funcione correctamente.

11. Presupuesto

Dependiendo de las necesidades de cada cliente, se debe de disponer de mayores recursos o menos, en este caso, se ha tomado en cuenta el supuesto anteriormente presentado, la plataforma se desarrollaría para una empresa tecnológica con todos los servicios necesarios. Se va a suponer que no se dispone de un *Data Center* donde alojar todos los datos y correr las máquinas virtuales, se va a construir mediante un servidor único, ya que en un ordenador personal el proyecto es complicado como ya se ha visto.

A continuación, se presenta la tabla donde se puede observar el presupuesto **anual** contando con un servidor dedicado con los recursos suficientes y un desarrollo completo de la plataforma añadiendo todos los módulos posibles.

	Días
Procesamiento Servidor	365
Mano de obra	200

Item	Precio unidad (€)	Unidades diarias (U)	Unidades totales (U)	Total (€)
Horas de trabajo de ingeniero	50	8	1.600	80.000€
Servidor (Grafica RTX 2060, RAM 64GB, Almacenamiento 6Tb)	1.500	-	1	1.500€
Horas procesamiento servidor	0,20	24	8.760	1.752€
			TOTAL	83.252€

Tabla 8 Presupuesto total

12. Bibliografía

[1] «OpenStack installation», jul. 29, 2022. https://docs.openstack.org/install-guide/common/conventions.html

[2] «Architecture Guide», 2022. https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/9/html-single/architecture_guide/index#doc-wrapper

[3] «Security OpenStack». https://subscription.packtpub.com/book/virtualization-and-cloud/9781782170983/6/ch06lvl1sec55/securing-openstack-swift

[4] «Networking API v2.0», ago. 08, 2022. https://docs.openstack.org/api-ref/network/v2/index.html?expanded=show-port-details-detail

[5] «Compute API», nov. 09, 2022. https://docs.openstack.org/api-ref/compute/?expanded=remove-flavor-access-from-tenant-removetenantaccess-action-detail#flavors

[6] «OpenStack API Documentation - Images», ago. 29, 2022. https://docs.openstack.org/api-ref/image/v2/?expanded=create-image-detail

[7] María José Martín, «Servicios Cloud: ¿Qué es IaaS, SaaS y PaaS?», mar. 2018, [En línea]. Available: https://profile.es/blog/servicios-cloud-que-es-iaas-saas-y-paas/

[8] «Los servicios de Cloud Computing más comunes en empresas». https://ayudaleyprotecciondatos.es/cloud-computing/servicios-empresas/

[9] «OpenStack API Documentation», jul. 25, 2022. https://docs.openstack.org/api-ref/identity/v3/?expanded=password-authentication-with-unscoped-authorization-detail,validate-and-show-information-for-token-detail

[10] «Install OpenStack Victoria on CentOS 8 With Packstack», jul. 02, 2021. https://computingforgeeks.com/install-openstack-victoria-on-centos/

[11] «Servicio de identificación en OpenStack», dic. 01, 2014. https://pilas.guru/20141201/servicio-de-identificacion-en-openstack/

[12] Frankier Flores, «Cloud Computing: Tipos de nubes, servicios y proveedores», mar. 2022, [En línea]. Available: https://openwebinars.net/blog/tipos-de-cloud-computing/

[13] AWS, «¿Qué es la informática en la nube?», 2022. https://aws.amazon.com/es/what-is-cloud-computing/?pg=TOCC

[14] «Cloud Computing: Conceptos Básicos», abr. 10, 2014. https://openwebinars.net/blog/cloud-computing-tutorial-conceptos-basicos/

Diseño e implementación de una nube computacional basada en OpenStack

ANEXOS

Anexo I – Framework

DJANGO

Django es un framework de aplicaciones web gratuito y de código abierto (open source) escrito en Python. Un framework web es un conjunto de componentes que te ayudan a desarrollar sitios web más fácil y rápidamente.

Cuando se construye un sitio web, siempre se necesitan un conjunto de componentes similares: una manera de manejar la autenticación de usuarios (registrarse, iniciar sesión, cerrar sesión), un panel de administración para el sitio web, formularios, una forma de subir archivos, etc.

Por suerte, hace tiempo que otros desarrolladores se dieron cuenta de que siempre se enfrentaban a los mismos problemas cuando construían sitios web, y por eso se unieron y crearon los frameworks (Django es uno de ellos) con componentes listos para usarse.

Los frameworks sirven para que no tener que desarrollar toda la infraestructura de cero cada vez y que podamos avanzar más rápido al construir un nuevo sitio.

Este framework ha permitido crear la plataforma cloud de una forma relativamente sencilla, ya que esta permite la conexión más o menos directa del front-end al back-end, haciendo así que los cambios que realice el usuario en la plataforma se puedan realizar rápidamente sin necesidad de recargar cualquier página.

INSTALACIÓN

Este framework utiliza principalmente Python como se ha comentado anteriormente, para la instalación de este es necesario un entorno virtual para trabajar con él. Este aisla cada proyecto que se haga en este framework y no haya conflictos.

Primeramente, se instala Python y se comprueba la versión instalada para que no haya problemas en la instalación de django y futuros proyectos:

```
$ Python3 --version
```

Para crear un nuevo proyecto basta con crear una carpeta donde se quiera realizar el proyecto, situarnos en esa carpeta mediante la terminal de Windows, preferiblemente powershell, ya que este nos permitirá realizar peticiones mediante comandos parecidos a los de Linux, se escribe lo siguiente:

```
python -m venv myvenv
```

Donde myvenv es el nombre del entorno virtual. Se instala Django

```
python -m pip install Django
```

A partir de aquí, se crea el proyecto en Django y se introducen los archivos necesarios para el desarrollo de este

```
(myvenv) C:\Users\Name\tfm>
django-admin.exe startproject mysite .
```

Posteriormente, dentro de la carpeta creada para el proyecto se disponen los archivos de la siguiente manera (paso importante para que django pueda compilar los scripts correctamente).

```
├tfm
├───manage.py
├───mysite
├   views.py
├   urls.py
├───users
├       ├───static
├       ├───templates
├                ├────users
├                    login.html
├                     Platform.html
├                     Script.js
└───requirements.txt
```

Solo se utilizan los archivos views para las funciones de Python y las urls para las llamadas a estas funciones.

Para el front-end los archivos login, platform y script para los lenguajes HTML, CSS y JavaScript.

Anexo II – Lenguajes

HTML y CSS

En cuanto a la programación básica de la plataforma, se ha utilizado HTML.

Para tener una disposición de página y hacer la plataforma de forma responsiva, se ha utilizado una biblioteca muy conocida para desarrollo web, "Bootstrap 5"

Esta librería permite darle estilos a la página fácilmente, iconos y poner la disposición de la página de forma responsiva.

JavaScript

Este lenguaje permite la conexión del *front* con el *back*, convirtiendo así la web en un software bidireccional, evitando así la estaticidad, y poder realizar las peticiones a *OpenStack*.

Para ello se han utilizado las siguientes librerías de JavaScript:

Librería	Función
JQuery	Es biblioteca multiplataforma de JavaScript, que permite interactuar con los documentos HTML
JConfirm	Un plugin de jQuery que proporciona un gran conjunto de características como, Auto-cierre, Ajax-carga, Temas, Animaciones y más. Ha permitido crear las ventanas de dialogo dentro de la plataforma
DataTables	DataTables es un plug-in para la librería jQuery Javascript. Es una herramienta muy flexible, que añade todas estas características avanzadas a cualquier tabla HTML.
JValidate	Librería secundaria de JQuery que permite la validación de formularios tales como el login de la plataforma y las ventanas de dialogos

Highcharts	Librería de JavaScript que permite crear todo tipo de gráficos usando Ajax como pasarela para recepción de datos
Bootstrap 5	Bootstrap 5 es la versión más reciente de Bootstrap, que es el framework más popular de HTML, CSS y JavaScript para crear sitios web con capacidad de respuesta y orientados a los dispositivos móviles.

Tabla 9 Plug-ins utilizados

Python

Django está construido con Python, gracias a este lenguaje, permite el acceso a base de datos por defecto y realizar las peticiones pertinentes a *OpenStack.*

Todas las funciones que se han creado se han realizado dentro de una clase en el archivo views.py y direccionada en el archivo urls.py.

Mediante llamadas en JavaScript mediante **Ajax** se llaman a las diferentes funciones declaradas de Python del archivo "urls", consiguiendo así la conexión bidireccional en la plataforma.