UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE CUENCA

CARRERA DE INGENIERÍA DE SISTEMAS

Trabajo de titulación previo
a la obtención del título de
Ingeniera de Sistemas e
Ingeniero de Sistemas

PROYECTO TÉCNICO:

"DESPLIEGUE DE UNA NUBE HÍBRIDA PARA ENTORNOS DE DESARROLLO IMPLEMENTANDO KUBERNETES Y DOCKERS EN LA PLATAFORMA OPENSTACK E INFRAESTRUCTURA EN AMAZON"

AUTORES:

NUBE MARIELA MEJÍA RODRÍGUEZ

JUAN PABLO BARBECHO CHIMBO

TUTOR:

ING. ERWIN JAIRO SACOTO CABRERA, PhD

CUENCA - ECUADOR

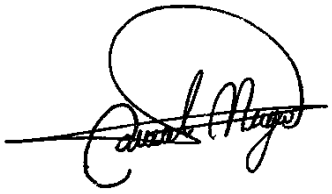
2021

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Nube Mariela Mejía Rodríguez con documento de identidad N° 0350216859 y Juan Pablo Barbecho Chimbo con documento de identificación N° 0105610059, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana, la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación: **"DESPLIEGUE DE UNA NUBE HÍBRIDA PARA ENTORNOS DE DESARROLLO IMPLEMENTANDO KUBERNETES Y DOCKERS EN LA PLATAFORMA OPENSTACK E INFRAESTRUCTURA EN AMAZON"**, mismo que ha sido desarrollado para optar por el título de: *Ingeniera de Sistemas e Ingeniero de Sistemas*, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores, nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, marzo del 2021



Nube Mariela Mejía Rodríguez

C.I. 0350216859

Juan Pablo Barbecho Chimbo

C.I. 0105610059

CERTIFICACIÓN

Yo, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **"DESPLIEGUE DE UNA NUBE HÍBRIDA PARA ENTORNOS DE DESARROLLO IMPLEMENTANDO KUBERNETES Y DOCKERS EN LA PLATAFORMA OPENSTACK E INFRAESTRUCTURA EN AMAZON"**, realizado por Nube Mariela Mejía Rodríguez y Juan Pablo Barbecho Chimbo, obteniendo el *Proyecto Técnico*, que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana.

Cuenca, marzo del 2021

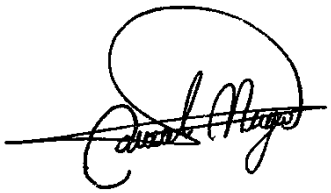
Ing. Erwin Jairo Sacoto Cabrera, PhD

C.I. 0301185229

DECLARATORIA DE RESPONSABILIDAD

Nosotros, Nube Mariela Mejía Rodríguez con documento de identidad N° 0350216859 y Juan Pablo Barbecho Chimbo con documento de identificación N° 0105610059, autores del trabajo de titulación: **"DESPLIEGUE DE UNA NUBE HÍBRIDA PARA ENTORNOS DE DESARROLLO IMPLEMENTANDO KUBERNETES Y DOCKERS EN LA PLATAFORMA OPENSTACK E INFRAESTRUCTURA EN AMAZON"**, certificamos que el total contenido de este *Proyecto Técnico*, es de nuestra exclusiva responsabilidad y autoría.

Cuenca, marzo del 2021



Nube Mariela Mejía Rodríguez

C.I. 0350216859

Juan Pablo Barbecho Chimbo

C.I. 0105610059

AGRADECIMIENTOS

Agradezco primeramente a Dios por guiarme y acompañarme en este camino, y de manera especial al PhD. Pablo Leonidas Gallegos, quien fue nuestro guía durante el desarrollo de este proyecto, así como al Grupo de investigación GIHP4C por brindarnos la oportunidad de realizar este trabajo de titulación. El más profundo agradecimiento a mis padres Medardo y Ayda por ser mi motor y mi motivación para seguir adelante en este camino, por darme su apoyo incondicional y sus grandes consejos que me han permitido guiarme en este camino. Finalmente quiero agradecer a mis tíos Laura y Luis quienes desde un principio me brindaron su apoyo para emprender este viaje en mi vida, aconsejándome, acompañándome, dándome fuerzas y siendo un pilar fundamental para emprender este logro.

Juan Pablo Barbecho Chimbo.

En primer lugar, quiero agradecer a Dios por brindarme la sabiduría y la fortaleza para superar obstáculos a lo largo de mi carrera y de toda mi vida. A la Universidad Politécnica Salesiana por haberme abierto sus puertas para educarme y convertirme en una profesional al servicio de los demás. Al PhD Pablo Leonidas Gallegos Segovia por permitirme realizar el proyecto de titulación en el grupo de investigación GIHP4C, por compartir sus conocimientos, por ser un guía en el desarrollo del proyecto de titulación, por el tiempo, la dedicación y la predisposición para ayudar en la elaboración de esta tesis a pesar de los duros momentos que estamos viviendo como consecuencia de la pandemia del COVID-19. También quiero expresar mi agradecimiento a toda mi familia de manera especial a mis padres Agustín Mejía y Susana Rodríguez por darme la vida, por convertirse en el pilar fundamental de mi vida, por el apoyo incondicional que me han brindado en cada instante, por los valores que me han inculcado los mismos que me han permitido ser una persona de bien, por el sacrificio que han hecho para brindarme una excelente educación durante toda mi vida. A mis hermanos y hermanas agradezco cada uno de sus consejos, felicidad, cariño, amor y apoyo incondicional. Por último, quiero agradecer profundamente a mis docentes por impartir sus valiosos conocimientos en las clases dictadas, a mis amigos, a mis compañeros y compañeras por cada momento de felicidad y experiencias que hemos vivido a lo largo de nuestra carrera universitaria, pero en especial a uno de ellos Juan Pablo por haber sido un excelente amigo y compañero de tesis, agradezco el esfuerzo, la dedicación y la perseverancia puesta para poder culminar este proyecto.

Nube Mariela Mejía Rodríguez

DEDICATORIAS

A mi madre

Por el esfuerzo que ha realizado, por el apoyo, por los consejos para que alcance este gran objetivo, hoy cosecha frutos de todo el sacrificio realizado para formarme como persona y como profesional.

A mi padre

Por todo sacrificio y esfuerzo de estar lejos y así darnos la oportunidad de ser una persona y profesional de bien, gracias por compartirme su carácter, su alegría y principalmente su disciplina y dedicación para lograr esta meta, gracias por ser mi mejor amigo, mi espejo, para siempre querer ser como el,

Hoy con la frente en alto me permito decirles que me siento orgulloso de María Ayda Chimbo y Paulino Medardo Barbecho Arias porque gracias a ustedes hoy concluye esta meta, gracias infinitas de corazón.

A mis hermanos

Por creer en mí, por acompañarme con risas y momentos de felicidad en cada paso de este duro camino, por darme ánimos y motivos para llegar a lograr esta meta tan añorada por toda nuestra familia, hoy me permito decirles a Estefania Barbecho y Michael Barbecho que, con esfuerzo, dedicación, y sacrificio si se puede lograr lo soñamos, me siento tan orgulloso de las personas que son gracias infinitas por brindarme su apoyo incondicional y estar siempre cuando los he necesitado

A mis amigos

que me han acompañado por todo este proceso y de manera muy especial a mi compañera y amiga de este proyecto Nube Mejia, gracias por la oportunidad por la confianza brindada en todo este trayecto, gracias por el esfuerzo y dedicación puesto para finalizar este proyecto de titulación, A Luis, Xavier, Vinicio, Elizabeth, Magna, Jorge por el grupo de estudio que conformamos durante todo este trayecto de vida.

Juan Pablo Barbecho Chimbo

A mis padres

Por todo el apoyo constante tanto moral como económico que me han brindado, por el amor incondicional, el trabajo, el esfuerzo y el sacrificio que han realizado para permitirme llegar a este momento tan especial en mi vida, ese sacrificio que valoro día tras día, gracias por inculcar en mí la valentía para superar dificultades y hacer posible que pueda culminar mi carrera profesional, gracias por ser mi ejemplo y hacerme sentir afortunada de tener unos padres como ustedes, sin ustedes no hubiese llegado a cumplir este sueño.

A mis hermanos y hermanas

Por los consejos, el cariño y apoyo incondicional que me han brindado, por estar siempre conmigo y animarme a alcanzar esta meta anhelada.

A mis amigos y a todas las personas que me han permitido ser parte de su vida

Por todos los momentos que hemos pasado juntos, por confiar en mí, por ser parte de este proceso y por ser una fuente de ayuda para alcanzar mis objetivos.

Nube Mariela Mejía Rodríguez

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO DE TÉRMINOS

AWS: Amazon Web Services

AWS CLI: Interfaz de Línea de Comandos de AWS

OpenStack: software libre y de código abierto, permite la gestión de clouds y además ofrece IaaS.

Kubernetes: es un orquestador de contenedores donde se puede desplegar, administrar y escalar aplicaciones.

Docker: software que permite construir, ejecutar e implementar aplicaciones de manera rápida.

SSH: Secure SHell protocolo que permite el acceso remoto a un servidor a través de un canal seguro.

Virtualización: software que simula características de hardware y permite crear sistemas informáticos virtuales.

testers: personas que se encargan de realizar pruebas de software en los ordenadores para verificar que funcionen de manera apropiada.

Dockerfile: archivo de texto que está constituido por comandos necesarios para crear una imagen de Docker.

Ubicuidad: capacidad de estar en diferentes sitios al mismo tiempo.

Big Data: son grandes cantidades de información que son recopilados y analizados para identificar ideas de negocios estratégicos.

Dispositivos PDA: Ayudante Personal Digital son dispositivos pequeños que combinan un ordenador, teléfono, Internet y conexiones de red, realizan funciones que se puede llevar a cabo en un ordenador de escritorio.

Hipervisores: software que posibilita la creación y ejecución de varias máquinas virtuales y separa su propio sistema operativo y recursos físicos de las máquinas virtuales que se crean.

NFS: Sistema de Archivos en Red, es un protocolo utilizado por Linux para compartir y permitir el acceso remoto a un sistema de archivos a través de la red.

HTTPS: Protocolo seguro de transferencia de hipertexto, es un protocolo de aplicación basado en el protocolo HTTP+SSL/TLS, mediante el cual se envían los datos de los usuarios entre sus ordenadores y el sitio web y además protege la integridad y confidencialidad de los mismos.

Alta disponibilidad: Protocolo que asegura la continuidad de un servicio durante un tiempo determinado.

Flexibilidad: son las posibilidades que tiene un software o hardware para agregar eliminar y modificar el sistema o recursos sin alterar su funcionamiento actual.

Token: Datos sensibles o críticos en programación se conoce como un lenguaje de programación.

DNS: servidor de nombre de dominio.

Docker Swarm: es un sistema establecido por desarrolladores de dockers que nos da la posibilidad de asociar un conjunto de host de docker en un clúster.

LDAP: Es un protocolo de la capa de aplicación que permite el acceso a un servicio de directorios.

Cloud computing: es un modelo que permite de forma ubicua y conveniente, el acceso a la red bajo demanda a un conjunto compartido de recursos informáticos configurables [1].

Gateway: es una puerta de enlace que permite la conexión entre diferentes redes [2].

PHP: Lenguaje de programación para de desarrollo web [3].

escalabilidad: posibilidad de incrementar recursos o servicios sin que afecte su funcionalidad.

IaaS: Infraestructura como Servicio

PaaS: Plataforma como Servicio

SaaS: Software como Servicio

Data center: Es un Centro de datos tanto físico como virtual.

Internet: Más conocido como la red de redes

EC2: Etiqueta comúnmente usada para mencionar a una máquina virtual.

Firewall: creada para bloquear bloquear el acceso no autorizado a un sistema así como también permitir el acceso a usuarios en específico [4].

Daemon: un software que se ejecuta en segundo plano.

Pods responsable de agrupar aquellos contenedores que necesitan trabajar juntos para ejecutar la aplicación [5].

RESUMEN

En la actualidad, las empresas que poseen enormes sistemas informáticos requieren una gran capacidad de recursos, procesamiento, almacenamiento, para soportar a cientos de usuarios y además generar entornos versátiles para incubar proyectos de desarrollo de software, de modo que, al desplegar el resultado de un proyecto o una actualización del mismo, el impacto sobre la infraestructura subyacente converja de una manera natural reduciendo de esta forma la heterogeneidad de hardware y software, así mismo que tareas como migración o replicación de un entorno privado hacia un entorno público no requieran una gran intervención y recursos informáticos que en consecuencia generan gastos capitales adicionales y pérdidas para la empresa. Como consecuencia a estas necesidades nosotros hemos propuesto utilizar tecnologías disruptivas de virtualización como son Dockers, Kubernetes sobre una infraestructura de Cloud híbrido y de esta manera crear escenarios apropiados para entornos de desarrollo, despliegue e implementación como un modelo de alta disponibilidad para los entornos de desarrollo de aplicaciones.

Por lo tanto, nuestra tesis plantea una alternativa para alcanzar una mayor eficiencia y rapidez al migrar o replicar una aplicación web, desplegando una infraestructura de Cloud híbrido, constituida por una infraestructura de nube privada basada en OpenStack y una infraestructura de nube pública basada en AWS (Amazon Web Services).

Con respecto a la infraestructura de OpenStack, se lanza la primera instancia que aloja a Docker con el fin de gestionar, brindar portabilidad, flexibilidad, ligereza, simplicidad y autosuficiencia a las aplicaciones desplegadas, para que se ejecuten de manera rápida y dispongan de un funcionamiento eficiente cuando son migrados de un entorno a otro. Así mismo, se procedió a crear una segunda instancia puesto que se requiere una integración completa con nuestra nube pública de AWS. Posteriormente se procede a generar Dockerfiles con la finalidad de Dockerizar nuestro aplicativo para migrar de un entorno privado a un público. Así, por ejemplo, se desarrolló una aplicación web de prueba y se demostró que en base al archivo de instrucciones Dockerfile, la herramienta Docker facilita la migración y despliegue de aplicaciones de manera rápida, óptima y segura.

Por otra parte, en la infraestructura de nube pública basada en AWS, se despliega un cluster de Kubernetes conformado por dos nodos y un maestro el cual nos va a permitir generar un entorno real de desarrollo basado en alta disponibilidad y réplicas, permitiéndonos en base al nodo maestro integrar nuestra infraestructura pública y privada mediante una instancia desplegada en OpenStack. Para esto, usamos la herramienta propia de AWS llamada AWS CLI, mismo que nos permite una comunicación y administración centralizada de ambos entornos en un mismo sitio. Además, nos facilita una estrategia de migración ágil y simplificada de un entorno a otro, de modo que, se beneficia a desarrolladores, administradores de sistemas, puesto que pueden proporcionar un entorno seguro a las aplicaciones y a los datos críticos de las empresas. Además, pueden implementar, ajustar, gestionar la escala de aplicaciones sin que se vean limitados por la infraestructura en la que se desarrolló el servicio.

Palabras clave: migración de aplicaciones, AWS, AWS CLI, Dockers, Nube Híbrida Kubernetes, OpenStack,

ABSTRACT

Nowadays, companies which have a huge variety of computing systems require a big resource capability, processing, storage, in order to support hundreds of users and generate versatile environments to incubate software development projects so that when displaying a project's result or an update, the impact on the underlying structure converges in such a natural manner, thus reducing the hardware and software heterogeneity, as well as tasks such as migration or replication from a private to a public environment do not require an important intervention and computing resources which consequently ends up in additional capital expenses and losses for the company. As a consequence, to these needs, we have proposed to use disruptive technologies of virtualization such as Dockers, Kubernetes on the hybrid Cloud infrastructure, thus creating appropriate settings for environments of development, display and implementation as a highly available model for the application development environment.

Therefore, our thesis presents an alternative in order to reach a better efficiency and speed when migrating or replicating a web application by displaying a hybrid Cloud infrastructure made up of a private Cloud infrastructure based on OpenStack and a public Cloud infrastructure based on AWS (Amazon Web Services).

Regarding OpenStack infrastructure, the first stage is presented, which houses Docker with the purpose of manage, provide portability, flexibility, lightness, simplicity and auto-sufficiency for the applications displayed so that they perform quickly and own an efficient work whenever they are migrating from an environment to another. In the same way, a second stage was created since a complete integration with our public AWS Cloud is required. After that, Dockerfiles are generated in order to Dockerize our app to migrate from a private to a public environment. Thus, for instance, a test web app was developed and it was proven that based on the Dockerfile instruction file, the Docker tool facilitates migration and display of applications in a fast, optimal and safe manner.

On the other hand, on the public Cloud infrastructure based on AWS, a cluster of Kubernetes is displayed, made up of two nodes and a master, which will allow us to generate a developing real

environment based on a high availability and replications, allowing us, according to the master node, integrate our public and private infrastructure by means of an instance displayed on OpenStack. For this purpose, we use a tool from AWS called AWS CLI, which allows us centralized communication and administration of both environments on the same site. In addition, it enables an agile and simplified migration strategy from an environment to another, thus, developers, system administrators are benefited since they are able to provide a safe environment for the applications and critical data of companies. Besides that, they can implement, adjust, manage the scale of applications without being limited by the infrastructure in which the service was developed.

Key words: application migration, AWS, AWS CLI, Dockers, Hybrid Cloud Kubernetes, OpenStack,

INTRODUCCIÓN

Los centros de datos han evolucionado dejando las tecnologías tradicionales, de manera que las empresas han incursionado estrategias como la virtualización abriendo nuevas posibilidades de modelos de Computación en la Nube, ya que estas ofrecen soluciones para brindar infraestructura como servicio, gestión centralizada facilitando la administración de un centro de datos con características como: migración, alta disponibilidad, escalabilidad, optimización de recursos y administración simplificada de operaciones. Al mismo tiempo tanto la virtualización como la computación en la nube nos permiten desplegar soluciones híbridas, es decir, generar una plataforma de nube privada y pública e integrar contenedores tipo Dockers y ejecutarlas sin importar la arquitectura del que provengan, los recursos que dispongan o el tipo de Cloud, a su vez, nos habilita muchas posibilidades para la creación de escenarios apropiados para entornos de pruebas y ambientes de producción.

Teniendo en cuenta el estudio realizado por la firma citrix ‘Cloud Confision’. El 95% de la población que utiliza servicios de la nube cree que nunca los ha utilizado. Como consecuencia en el Ecuador, el tema recién está en consideración para ser implementado en el sector empresarial por lo que, estas tecnologías puesto que son nuevas en el país van a permitir solucionar una gran cantidad de problemas que se dan en las empresas al momento de migrar o replicar aplicativos que están situadas en diferentes infraestructuras, las mismas que son consideradas para este proyecto.

De ahí que, para implementar esta solución se optará por la comunicación entre una infraestructura de nube privada en OpenStack y una infraestructura de nube pública en Amazon. A su vez, esta nube híbrida plantea colocar a Kubernetes y Docker como una plataforma subyacente que permita gestionar la migración, réplicas, alta disponibilidad, escalabilidad de contenedores en máquinas virtuales, puesto que estas plataformas poseen instancias o controladores que desempeñan una función importante para la plataforma OpenStack y Amazon Web Services.

JUSTIFICACIÓN

Las tecnologías de la información y comunicación han forzado a las empresas a incorporar características de alta disponibilidad, escalamiento e ubicuidad de acceso a sus aplicaciones críticas migrándolas a la nube, por lo que los centros de datos tradicionales centrados en aplicaciones únicas deben migrar hacia los centros de nueva generación también conocidos como definidos por software, de igual manera los centros de datos definidos por software deben ampliarse e integrarse a ecosistemas alojados en la nube pública y privada. Sin embargo, dentro de los centros de datos en la nube existen micro ecosistemas que ayudan a la automatización y mejora de desarrollo de proyectos informáticos debido a su facilidad de despliegue, integración, migración y uso reducido de recursos, por lo que presentamos Dockers y Kubernetes.

Puesto que los Dockers nos van a permitir crear contenedores ligeros y portables para las aplicaciones de software distribuido, donde estos se puedan migrar de un ambiente de Cloud a otro, independientemente del SO que la VM tenga internamente, de modo que, se ve la necesidad de desplegar un cluster de Kubernetes dado que esta plataforma permite una gestión centralizada de Dockers logrando así una comunicación entre los nodos del Cloud híbrido desplegados tanto en la infraestructura privada de OpenStack y en la infraestructura pública de AWS.

Por lo tanto, mientras tengamos una arquitectura híbrida con Dockers y Kubernetes ejecutándose podremos migrar, replicar o respaldar un determinado servicio, reduciendo así el tiempo de despliegue, beneficiando a desarrolladores y administradores de sistemas.

OBJETIVOS

OBJETIVO GENERAL

- Desplegar una nube híbrida para entornos de desarrollo implementando Kubernetes y Dockers en la plataforma OpenStack e infraestructura en Amazon.

OBJETIVOS ESPECÍFICOS

- Desplegar una infraestructura de nube privada sobre OpenStack.
- Integrar Dockers y Kubernetes a la infraestructura de nube privada de OpenStack.
- Integrar la nube privada con la nube pública.
- Generar y documentar estrategias de migración de contenedores con Docker y Kubernetes.
- Crear y documentar escenarios y protocolos de pruebas basado en una simulación básica transaccional de un Core bancario.
- Analizar, cuantificar y documentar los resultados obtenidos.

CAPÍTULO 1: ESTADO DEL ARTE Y FUNDAMENTACIÓN TEÓRICA

En este apartado hacemos un acercamiento de las tecnologías que van a ser desplegadas dentro de nuestra tesis en la cual iremos dando diferentes prelaciones a cada uno de los componentes integrales de nuestro estudio.

1.1. CLOUD COMPUTING

1.1.1. DEFINICIÓN

"La nube (Cloud) es un conjunto infinito de servidores de información desplegados en centros de datos a lo largo de todo el mundo, donde se almacenan millones de aplicaciones web y enormes cantidades de datos (Big Data) a disposición de miles de organizaciones empresas y cientos de miles de usuarios que se descargan y ejecutan directamente los programas y aplicaciones del software almacenados en dichos servidores tales como Google, Amazon, IBM o Microsoft" [6].

"Cloud Computing es un modelo que permite de forma ubicua y conveniente, el acceso a la red bajo demanda a un conjunto compartido de recursos informáticos configurables (como redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser aprovisionados y liberados de manera rápida con un mínimo esfuerzo de administración o interacción del proveedor de servicios" [1].

1.1.2. CARACTERÍSTICAS

Según el NIST las características principales son:

- **Autoservicio bajo demanda:** brinda a los clientes un acceso automático y flexible a cada uno de los servicios o recursos disponibles en el Cloud Computing, según sus

necesidades, por tal motivo, no es necesario la interacción manual con uno o más de los proveedores de soluciones en la nube [7].

Entre los servicios disponibles en la nube se puede mencionar el tiempo CUP (Contents Under Pressure), almacenamiento, acceso a la red, tiempo de servidor, aplicaciones web, etc., [8].

- **Acceso amplio a la red:** esta característica incluye la posibilidad de que los usuarios accedan a los servicios contratados en la nube, mediante mecanismos estándares, sin importar el tiempo o el lugar donde se encuentren, siempre y cuando estén conectados a la red de Internet a través de diferentes tipos de dispositivos como, por ejemplo, estaciones de trabajo, teléfonos móviles, dispositivos PDA o computadoras portátiles que faciliten el uso de las plataformas [8].
- **Agrupación de recursos:** mediante un modelo de infraestructura compartida, los recursos del proveedor como almacenamiento, procesamiento, memoria, etc., son puestos al servicio de múltiples clientes, por esta razón, los proveedores deben tener un conjunto recursos informáticos físicos y virtuales para poder asignar y reasignar dinámicamente de acuerdo a la demanda de los consumidores, estos recursos no siempre son los mismos ya que tienen la posibilidad de ser modificados para satisfacer las necesidades de los usuarios. [9].
- **Elasticidad rápida:** esta característica permite aprovisionar y liberar los recursos informáticos de manera transparente para los usuarios y al mismo tiempo de forma rápida y flexible según las necesidades del consumidor, de modo que, se da a los clientes una perspectiva de disponer recursos ilimitados que pueden adquirir en cualquier instante [10].
- **Servicio medido:** los proveedores de soluciones en la nube (CSP) monitorean, gestionan y perfeccionan los recursos y servicios con la finalidad de mejorar las capacidades, para ello, emplean un patrón comercial de pago por el uso. Además, dado que los servicios son dinámicos el costo también es dinámico. Así pues, en base a las mediciones de las diferentes capacidades de los servicios utilizados se brinda información transparente tanto para el consumidor como para el proveedor [9].

1.1.3. ARQUITECTURA DE CLOUD COMPUTING

Una infraestructura de Cloud Computing se despliega en hardware, la misma base en la que se desarrolla la plataforma y las aplicaciones, para el desarrollo del mismo lo dividimos de la siguiente manera.

1.1.4. TIPOS DE SERVICIOS

El Cloud Computing en la actualidad alcanza una gran importancia en la vista actual de las tecnologías del mundo empresarial. Es por eso que se vuelve necesario distinguir los principales tipos de Cloud de acuerdo a las necesidades de la empresa, podemos agrupar estos servicios en tres tipos como: IaaS, PaaS y SaaS.

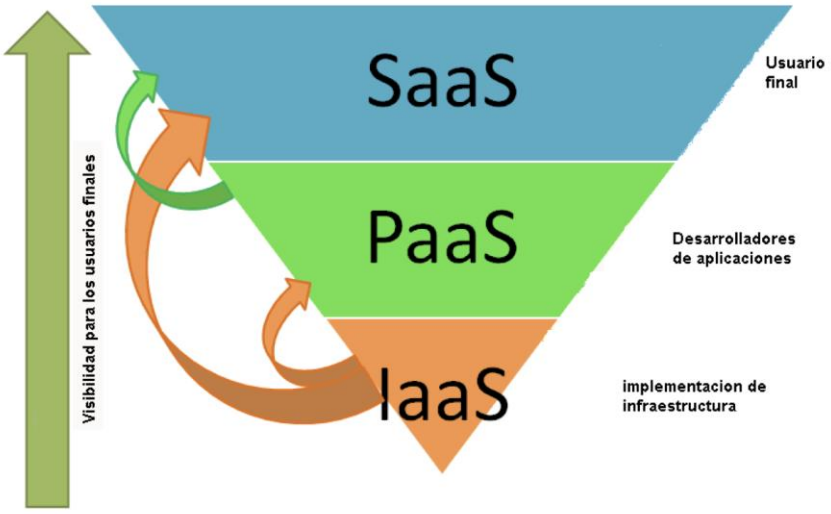


Figura 1. Servicios de Cloud. Fuente [11].

1.1.4.1. SOFTWARE COMO SERVICIO (SAAS)

En base a la figura [1] la encontramos en la capa superior del servicio en la nube, esta entrega la aplicación propiamente como servicio, es decir lo referente a su mantenimiento, disponibilidad, desarrollo y soporte se encuentra a cargo por el mismo desarrollador conocido como proveedor del servicio. Al mismo tiempo este se encarga de brindar el servicio a todos los usuarios conectados a la red, eliminado necesidad de que este tenga que instalar un software adicional. Las aplicaciones

en este servicio se basan en un modelo de uno a muchos, brindando la facilidad de crear un solo producto pero que sea usado por varios a la vez [11].

Como ejemplo claro de este modelo como servicio son los servicios de correo electrónico como: Gmail, Hotmail que tienen la característica de accederse desde un navegador web.

1.1.4.2. PLATAFORMA COMO SERVICIO (PAAS)

En base a la figura [1] la encontramos en la capa intermedia del servicio en la nube. En esta capa la entidad u organización contrata la infraestructura completa es decir el sistema operativo, las plataformas para alojar las aplicaciones, lenguajes de programación y, los servicios de internet, pero como punto a tomar en cuenta la organización no tiene el control de la infraestructura, pero si dispone de escalabilidad automática en caso de requerido. Con referente a las aplicaciones no se incluye en el servicio, este corre y debe ser ejecutada por las organizaciones, encargándose de esta sección el departamento de desarrollo [12].

1.1.4.3. INFRAESTRUCTURA COMO SERVICIO (IAAS)

En base a la figura [1] la encontramos en la capa inicial del servicio en la nube. En este primer nivel la infraestructura de computación dígase servidores, software, plataformas y componentes de red, son proporcionadas por un proveedor, es decir este servicio es administrada y gestionada por la misma entidad permitiendo así la escalabilidad y elasticidad según sea necesario, por lo que se realiza bajo demanda, el cual bajo dichas características podemos crear entornos apropiados para el desarrollo y así también ejecutar realizar pruebas las diferentes aplicaciones que se realicen en la organización [13].

Como uno de los proveedores virtuales más conocidos y usados a nivel mundial es Amazon bajo AWS que ofrece servicios como EC2, S3, SimplrDB, EKS, etc. Ofreciendo servidores virtuales, storage, y DB [11].

En base a los recursos IaaS se subdivide en tres grupos:

Computación como servicio (CaaS):

Brinda todos los recursos necesarios a las instancias al momento de requerir escalamiento y potencia de una manera autónoma si así lo requieren. Gracias a su interfaz de ayuda podemos gestionar y administrar nuestra instancia, es decir, iniciar, detener, eliminar y reiniciar cuando se lo requiera [14].

Almacenamiento como servicio:

Brinda almacenamiento bajo demanda basándose en proveedores de servicios y mediante estos llegar a las entidades finales, por lo general este servicio se oferta en GB [14].

Base de datos como servicio (DaaS):

Estandariza los procesos de administración y gestión muy conocidos en un entorno de base de datos como leer, escribir, actualizar [14].

1.1.5. MODELOS DE CLOUD COMPUTING

El Cloud Computing puede implementarse los bajo los siguientes modelos esto dependerá de las necesidades o los tipos de servicios a ofrecer:

1.1.5.1. CLOUD PÚBLICA

Es una infraestructura en la nube que está a disposición del público en general, esto quiere decir que la nube pública describe la computación en la nube en sentido tradicional, puede ser propiedad de organizaciones empresariales, académicas, gubernamentales o alguna combinación de ellas y además puede ser administrada y operada por proveedores de servicios en la infraestructura de Cloud [1].

1.1.5.2. CLOUD PRIVADA

Es una infraestructura en la nube exclusiva de una sola organización que comprende múltiples consumidores, puede existir dentro o fuera de las instalaciones, es necesario recalcar que es

propiedad de una sola organización y que puede ser gestionada por la misma entidad, un tercer involucrado o alguna composición de ellos [1].

1.1.5.3. CLOUD COMUNITARIA

Es una arquitectura compartida por diferentes entidades y es de uso exclusivo de una comunidad específica, puesto que es administrada por entidades o un involucrado y además puede encontrarse en las instalaciones de manera interna o externa [15].

1.1.5.4. CLOUD HÍBRIDA

Esta es una infraestructura en la nube que comprende dos o más nubes públicas y privadas que interoperan a través de la tecnología que admite la transferibilidad de datos y aplicaciones, así mismo, si la nube híbrida se queda sin capacidad en la nube privada puede acudir rápidamente a la nube pública y obtener recursos adicionales para seguir operando el negocio [15].

Esta última nube es la que brinda un mayor número de posibilidades como: conectividad de red segura y baja latencia, conectividad de ancho de banda alto, funcionalidad de aprovisionamiento, migración y como las más destacadas, el fácil transporte de datos y aplicaciones entre nubes privada y pública para obtener más flexibilidad y opciones de desarrollo. A su vez, ayuda a definir protocolos de ampliación de recursos tanto de la nube privada como de la pública ante una gran demanda cuando el sistema lo requiera, por ejemplo: un sistema de transacciones o aplicativo se ejecuta en la nube pública hasta que se produce un sobrecargo, en este punto los recursos pueden ampliarse de la nube privada hacia la nube pública para aprovechar más recursos informáticos y no perder calidad de servicio.

Las tecnologías antes detalladas han soportado el desarrollo de redes de Quinta Generación (5G) y el desarrollo de diferentes modelos de negocio como los planteados en [16] [17] [18] [19] [20], así como soportar servicios de Internet de las Cosas (IoT) como los descritos por los autores en [21] [22] [23] [24].

1.1.6. PLATAFORMAS DE CLOUD OPEN SOURCE

En la actualidad, las plataformas open source o de código abierto como OpenStack, CloudStack, Eucalyptus y OpenNebula son alternativas a las soluciones propietarias de empresas comerciales.

1.1.6.1. EUCALYPTUS

Es una plataforma propiamente utilizada para desplegar nubes híbridas y privadas siendo compatible con varias plataformas, pero una de las más conocidas es Amazon Web Service (AWS), haciendo posible una instalación sencilla en GNU/Linux. Eucaliptus brinda la compatibilidad de trabajar con grandes entornos de virtualización de hardware incluso con los hypervisores: VMware, Xen y KVM, donde su infraestructura se base en el uso del hardware y software físico sin modificar el equipo, y al mismo tiempo disponer y utilizar simultáneamente las interfaces de red cuando sea necesario [25].

Una de las características de esta plataforma es que dispone de capacidades de autogestión de suministro, seguridad, características de administración y configuración de usuarios finales [25].

1.1.6.2. OPENNEBULA

Es una plataforma flexible de código abierto con un modelo simple gestión, recursos compartidos, actualizaciones constantes, libertad de modificación, adaptación de productos a las necesidades actuales. Dispone de las últimas actualizaciones en el campo de la virtualización de data centers para un desarrollo básico, que permite desplegarlo en una infraestructura, volúmenes, capacidades preexistentes y además no depende de hypervisores [25].

Además, OpenNebula facilita la creación de cualquier tipo de nubes: públicos, privados e híbridos, que como componente adicional en una nube híbrida realiza una estructura propia de infraestructura pública y privada que este obtiene a partir de proveedores públicos como es AWS, google y ElasticHost, es decir que gracias a estas plataformas podemos tener escalabilidad y alta disponibilidad [26].

1.1.6.3. CLOUDSTACK

Como una plataforma open source con características esenciales que la mayoría de las empresas demandan al desplegar un Cloud como servicio donde su uso general es para desplegar infraestructuras (IaaS) como las más conocidas tenemos las Cloud privadas, públicas e híbridas basándose en conjuntos de recursos informáticos [27].

Es decir, esta plataforma permite gestionar su cómputo, red (NaaS), almacenamiento y gestión de cuentas aprovechando tanto el hardware como el software del equipo. Una característica común con plataformas similares es que utiliza hipervisores conocidos como KVM, vSphere y XenServer / XCP para desplegar su virtualización teniendo también una compatibilidad con las APIS de AWS EC2 y S3 [27].

1.1.6.4. OPENSTACK

Es una infraestructura de código abierto que hace uso de un conjunto de recursos virtuales para gestionar y diseñar los tres tipos de nubes públicas, privadas e híbridas. La plataforma OpenStack se compone por "proyectos" donde estas gestionan la computación, las redes, el almacenamiento, identidad e imagen.

“En la virtualización, los recursos, como el almacenamiento, la CPU y la RAM, se extraen de distintos programas específicos de los proveedores y se dividen con un hipervisor antes de distribuirlos según sea necesario. OpenStack utiliza un conjunto uniforme de interfaces de programación de aplicaciones (API) para extraer todavía más recursos virtuales, los cuales distribuye en conjuntos distintos que se utilizan para potenciar las herramientas del Cloud Computing estándares que utilizan los administradores y los usuarios” [28].

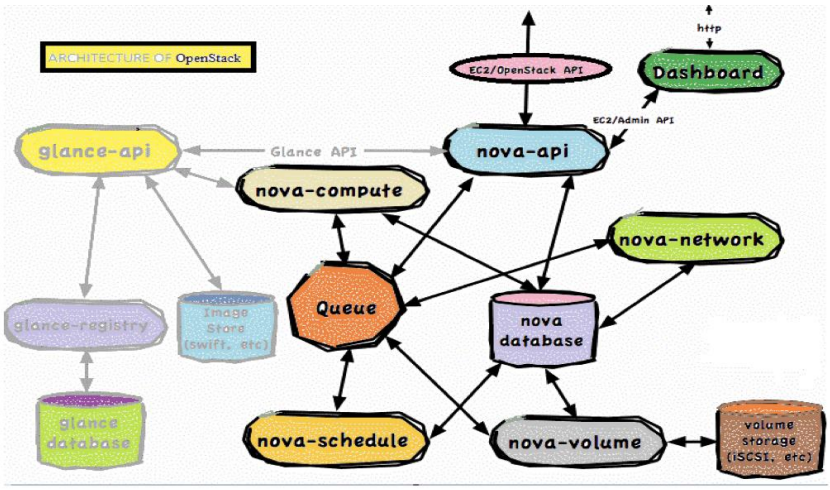


Figura 2. Arquitectura OpenStack. Fuente [29].

COMPONENTES GENERALES DE UNA INFRAESTRUCTURA OPENSTACK

Cómputo OpenStack (Nova): es un controlador propio de la plataforma utilizado para desplegar y administrar instancias, VM [29].

Almacenamiento de objetos (Swift): componente que permite tanto a objetos como archivos tener escalabilidad y redundancia. Swift por defecto escribe sus archivos y objetos en los discos que están distribuidos en la data center, ya que el único responsable de mantener una integridad del clúster es el software [29].

Almacenamiento en bloque (Cinder): como su nombre lo dice es un almacenamiento análogo en bloque, que permite a una instancia acceder a las ubicaciones exactas de una unidad de disco, además este componente nos permite tener un almacenamiento persistente [29].

Redes (Neutrón): componente dedicado a la gestión y administración de redes incluidas direcciones IP dentro de OpenStack [29]

OpenStack Dashboard (Horizon): es la interfaz gráfica de OpenStack que permite una gestión, automatización y aprovisionamiento más sencilla para los usuarios administradores de la nube [29].

Servicio de identidad (Keystone): es un método o proceso de autenticación para OpenStack, para que los usuarios puedan acceder a su administración y principalmente este permite integrase a varios de los directorios como LDAP [29].

Servicio de imágenes (Glance): controlador encargado de guardar, proporcionar, registrar y entregar las imágenes al entorno de OpenStack, como característica de glance es que permite usar dichas imágenes como plantillas y de esto lanzar un número ilimitado de instancias [29].

Telemetría (Ceilometer): es un controlador dedicado a la facturación a usuarios individuales dentro del Cloud manteniendo un registro del uso de recursos y componentes dentro del sistema [29].

Orquestación (Heat) y (Magnum): son controladores de OpenStack dedicados exclusivamente a desplegar contenedores, permitiendo una integración completa con la plataforma de OpenStack, además permite almacenar aplicaciones basadas en Dockers compartiendo los recursos [29].

1.1.7. PLATAFORMAS DE CLOUD PROPIETARIAS.

1.1.7.1. GOOGLE CLOUD

Es un medio a través del cual se puede acceder con gran facilidad a un conjunto de servicios de la plataforma de Google. La plataforma incluye una serie de servicios para computación, almacenamiento y desarrollo de app que se despliegan en el hardware de Google. Los desarrolladores de software, los administradores de la nube y otros profesionales de TI empresarial pueden acceder a los servicios de Google Cloud Platform (GCP) a través de Internet y usarlos según sus necesidades [30].

1.1.7.2. MICROSOFT AZURE

Es una arquitectura que se encuentra en constante expansión, permite acceder y administrar los recursos y servicios proporcionados por Microsoft. A su vez, estos

servicios y recursos incluyen almacenar y transformar datos según las necesidades de los usuarios. Además, para tener acceso a los mismos, se requiere conectividad a Internet. Entre los principales tipos de servicios proporcionados por esta plataforma están: SaaS, PaaS IaaS. A su vez, Microsoft Azure proporciona un entorno compatible con varios lenguajes de programación, herramientas y frameworks que son propiedad de Microsoft y de terceros, con el propósito de afrontar retos empresariales actuales y futuros [31].

1.1.7.3. IBM CLOUD

Es una arquitectura de estándares abiertos donde se puede crear, ejecutar y gestionar aplicaciones. Ofrece PaaS e IaaS. A su vez, proporciona opciones informáticas flexibles, herramientas DevOps y un potente conjunto de API y servicios de IBM y de terceros para que pueda centrarse en crear experiencias de usuario excelentes. Así pues, las organizaciones pueden implementar y acceder a recursos virtualizados, almacenamiento y redes, a través de Internet [32].

1.1.7.4. ALIBABA CLOUD

Es la nube pública más grande de China, que comprende una serie de servicios en la nube y expansión de disponibilidad global. Alibaba Cloud ofrece soluciones de computación, redes, almacenamiento, bases de datos, seguridad y nube híbrida, así como también, servicios de desarrollo de aplicaciones e infraestructura en la nube. Cabe destacar que sigue un patrón de precios estándar de pago por uso adoptado por la mayoría de los proveedores [33].

1.1.7.5. ORACLE CLOUD

“Es un conjunto complementario de servicios en la nube que permite crear y ejecutar varias aplicaciones y servicios en un entorno de alta disponibilidad. Oracle Cloud Infrastructure (OCI) proporciona funciones informáticas de alto rendimiento y capacidad de almacenamiento en una red virtual superpuesta flexible, a la que se puede acceder de forma segura desde la red local, también proporciona elasticidad en tiempo real para las aplicaciones comerciales al combinar servicios autónomos de Oracle, seguridad integrada y computación sin servidor” [34].

1.1.7.6. AMAZON WEB SERVICES

Es una arquitectura segura, más utilizada y más completa a nivel mundial, proporciona más de 175 servicios. Además, grandes empresas y agencias gubernamentales prefieren como opción para disminuir costos, ampliar la agilidad y acelerar la innovación. Por otro lado, la computación en la nube está acelerando su crecimiento gracias a las soluciones de AWS [35].

AWS provee servidores al instante, de igual modo, proporciona varias cargas de trabajo, mayores opciones de almacenamiento y mejores medidas de seguridad, permitiendo en efecto, que las empresas puedan crear aplicaciones sofisticadas con flexibilidad, escalabilidad y confiabilidad [36].

La plataforma de AWS está constituida por muchos servicios en la nube que pueden ser utilizados según las necesidades de las empresas. A continuación, se describen algunos de los servicios.

VPC: "permite aprovisionar partes aisladas lógicamente de la nube de AWS donde se puede lanzar recursos de AWS en una red virtual que se defina. Así mismo, permite controlar todos los aspectos del entorno de red virtual, incluso el rango de direcciones IP, la creación de subredes, la configuración de tablas de enrutamiento y gateways de red. Además, se puede utilizar direcciones IPv4 o IPv6 para acceder a recursos y aplicaciones de forma segura y sencilla. Los grupos de seguridad y las listas de control de acceso a la red, ayudan a controlar el acceso a las instancias de Amazon EC2 en cada subred" [37].

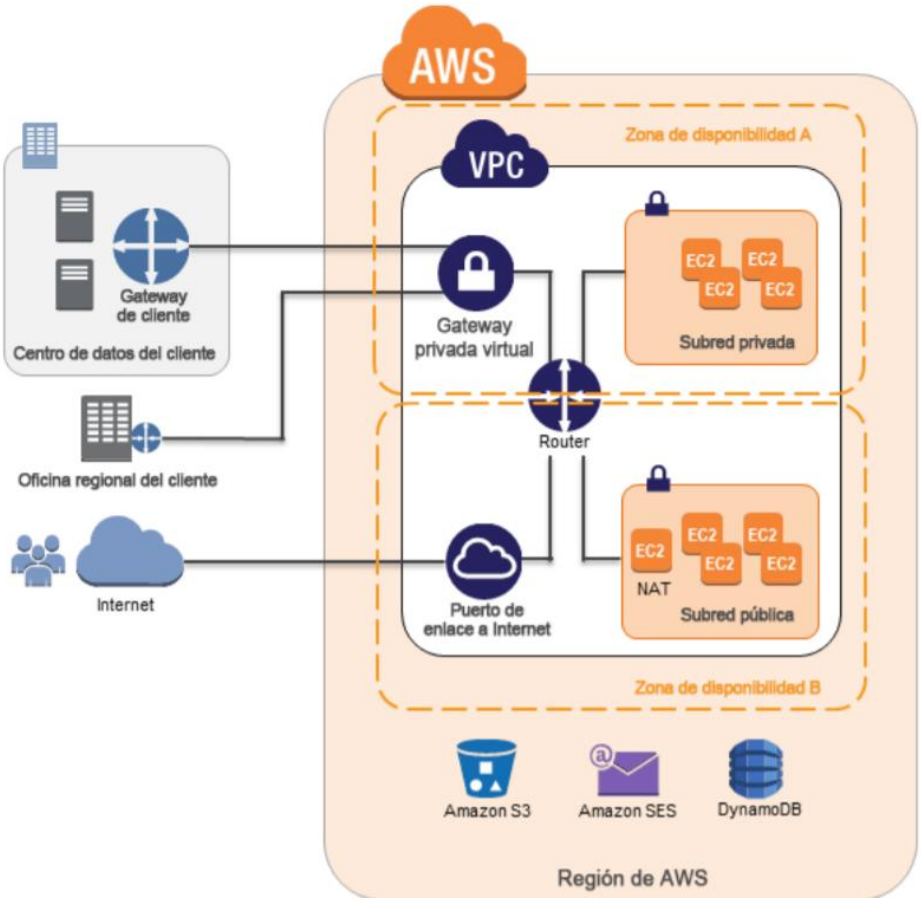


Figura 3. Arquitectura de Amazon VPC. Fuente [38].

Amazon Elastic Kubernetes Service: "proporciona flexibilidad para iniciar, ejecutar y escalar aplicaciones de Kubernetes en la nube de AWS o en entornos locales. Además, ayuda a proporcionar clústeres seguros y de alta disponibilidad y a automatizar tareas clave como parches, aprovisionamiento de nodos y actualizaciones. Intel, Snap, Intuit, GoDaddy y Autodesk usan EKS para ejecutar tareas críticas y aplicaciones sensibles con su seguridad, confiabilidad y escalabilidad. Cabe destacar que Amazon EKS puede migrar de manera fácil cualquier aplicación de Kubernetes estándar a EKS sin refactorizar su código" [39].

AWS CloudFormation: esta plantilla describe los recursos necesarios y sus dependencias para que pueda iniciados y configurarlos como una pila. Las plantillas permiten crear, actualizar y eliminar las pilas las veces que sean necesarias, sin administrar los recursos por separado [40].

Amazon EC2: permite la escalabilidad en la nube de AWS. Su objetivo es simplificar tareas de desarrollo y despliegue de apps. La sencilla interfaz de Amazon EC2 permite lanzar servidores virtuales, configurar la seguridad, las redes y administrar el almacenamiento según sea necesario [41].

AWS CLI: permite administrar y controlar los servicios de AWS mediante comandos en su shell de la línea de comandos y automatizarlos mediante scripts [42].

INFRAESTRUCTURA GLOBAL DE AWS

"AWS ofrece 175 servicios completos a partir de centros de datos distribuidos en todo el mundo. AWS posee el ecosistema más grande y dinámico con millones de clientes activos y miles de socios a nivel mundial" [43].

La infraestructura AWS está construida a base de regiones y zonas de disponibilidad conectadas de manera redundante. La región de AWS físicamente posee zonas de disponibilidad que están constituidas por uno o más centros de datos separados [44].

MAPA DE LA INFRAESTRUCTURA DE AWS

"AWS incluye 77 zonas de disponibilidad en 24 regiones geográficas de todo el mundo. Además, se anunciaron planes para incorporar 15 zonas de disponibilidad y 5 regiones de AWS adicionales en India, Indonesia, Japón, España y Suiza" [43].

Figura 4. Zonas y regiones de AWS. Fuente [43].

1.2. DOCKER

1.2.1. DEFINICIÓN

"Es un proyecto de código abierto que empezó a ganar popularidad dado que permite empaquetar y aislar aplicaciones con todo lo necesario para que los procesos funcionen correctamente, es decir, permite mover la aplicación que se encuentra en el contenedor a diferentes entornos, garantizando la ejecución de los servicios independientemente del entorno en el cual sean desplegados estos contenedores de Linux y al mismo tiempo a un muy bajo costo computacional" [45].

En la plataforma interna de Docker existen cuatro componentes principales a este le agregamos cliente servidor.

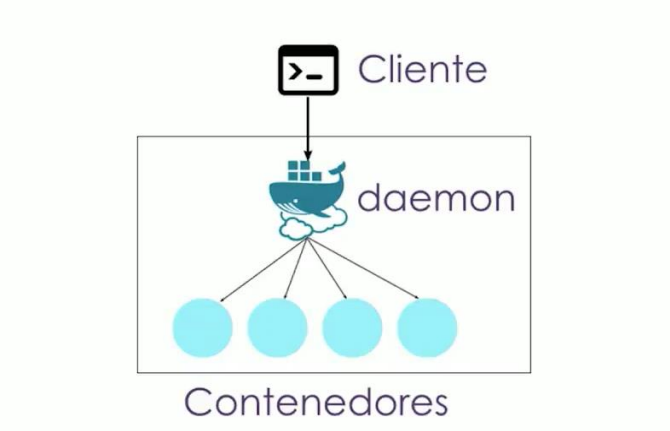


Figura 5. Arquitectura Docker. Fuente [46].

CLIENTE Y SERVIDOR (DOCKER)

En este complemento el servidor recibe una solicitud del cliente y procesa dicha información, Docker envía la transferencia. En esta sección el daemon se puede procesar en la misma instancia o máquina virtual, pero también se lo puede realizar de manera remota [46].

IMÁGENES (DOCKER)

Para desplegar una imagen existen dos procesos.

1) Se crea una imagen a partir de una plantilla de lectura, donde las imágenes que se usan como sistema operativo son las mismas imágenes base es decir por lo general cualquier distribución de linux donde estos son capaces de ejecutar un sistema completo mientras empaquetan un aplicativo. También se puede crear una imagen desde cero donde después de instalar todas las dependencias necesarias se pueden agregar las aplicaciones, pero como proceso adicional es que se deben generar o crearse una imagen nueva llamando a este proceso "comprometer un cambio" [46].

2) Se puede crear un archivo acoplable, esto parte de una serie de instrucciones del comando "Docker build" creando así una imagen de manera autónoma [46].

1.2.2. REGISTROS DE DOCKER

Todas las imágenes desplegadas en Docker se pueden guardar en registros propios de la plataforma, esto funciona como un repositorio de código fuente con la facilidad de enviar y descargar desde un mismo lugar. Existen dos tipos de repositorios: públicos y privados [46].

1.2.3. DOCKER HUB

Es un repositorio o registro en el cual cualquier usuario puede enviar una imagen de su autoría y también descargarse todas las imágenes disponibles dentro de esta plataforma. Todas las imágenes de este repositorio se pueden compartir en un área de preferencia como puede ser pública o privada a partir de la función Docker Hub [46].

1.2.4. Contenedores Docker

Partiendo de una imagen se crea un contenedor Docker. Contenedor es un conjunto de instrucciones preparado para correr un aplicativo, es decir la app de puede correr de manera independiente o aislada [46].

1.3. KUBERNETES

1.3.1. DEFINICIÓN

Kubernetes o k8s es un gestionador creado en el año 2014 y donado a la comunidad en 2015. Se define como una infraestructura de código libre, que no sólo permite gestionar despliegues, escalar y administrar aplicaciones de contenedores, sino también, permite automatizar de servicios y balanceo de carga [47].

K8s permite agrupar contenedores en un solo nodo llamado Pod. En otras palabras, agrega una capa de abstracción a varios contenedores agrupados e interconectados entre sí, proporcionando redes y almacenamiento. De modo que, K8s puede mover y expandir contenedores a través del

clúster con mucha facilidad, además, permite la portabilidad entre proveedores de infraestructura [47].

Los contenedores son desplegados en varios hosts a fin de proporcionar alta disponibilidad y cargas de trabajo para satisfacer necesidades requeridas por los usuarios. Así mismo, los componentes y herramientas de Kubernetes facilita el despliegue, escalado y administración de apps [48].

1.3.2. CARACTERÍSTICAS

Kubernetes posee varias características, entre ellas están:

- **Service Discovery y load balancing:** "provee un sistema de autodescubrimiento entre contenedores y enrutamiento mediante la asignación de direcciones IP y también provee un nombre único a los servicios mediante el cual balancea la carga de tráfico" [47].
- **Orquestación del almacenamiento:** permite que el sistema de almacenamiento necesario se acople automáticamente. Así, por ejemplo: almacenamiento local, de un CSP o almacenamiento de red [47].
- **Despliegues y rollbacks automáticos:** esta característica permite definir políticas de despliegue para no parar los servicios al momento de actualizar las aplicaciones o realizar algún cambio en ellas [47].
- **Ejecución Batch:** "K8s puede gestionar cargas de trabajo batch y CI, reemplazando los contenedores que fallan" [47].
- **Planificación:** "los contenedores son colocados en los mejores nodos según los requisitos y restricciones, sin afectar la disponibilidad" [47].
- **Autorreparación:** "reinicia contenedores que fallan, reemplaza y reprograma los contenedores cuando los nodos mueren y no los presenta a los usuarios hasta que estén listos" [47].
- **Gestión de la configuración y secrets:** "se puede almacenar los datos sensibles como tokens, claves a una API, claves SSH, contraseñas, etc., en los objetos llamados secrets que

se guardan en un sitio seguro y no dentro de una imagen o almacenamiento. Esto permite desplegar y actualizar la configuración sin reconstruir las imágenes" [47].

- **Escalado y auto-escalado:** "se puede escalar aplicaciones manualmente o definir reglas para escalar aplicaciones en función del uso del CPU" [47].

1.3.3. BENEFICIOS

- **Ágil creación y despliegue de aplicaciones:** esto simplifica las operaciones y mejora la eficacia al generar imágenes de contenedor[48].
- **Observabilidad:** no solo muestra información e indicadores del sistema operativo, sino que también, muestra el estado de la app [48].
- **Consistencia entre los entornos de desarrollo, pruebas y producción:** la app se ejecuta de la misma manera tanto en una computadora portátil como en el Cloud [48].
- **Portabilidad entre nubes y distribuciones:** disponible en Ubuntu, RHEL, CoreOS, datacenter físico, Google Kubernetes Engine y más [48].
- **Microservicios distribuidos, elásticos, liberados y débilmente acoplados:** "las aplicaciones se dividen en partes pequeñas e independientes que son desplegadas y administradas de forma dinámica, es decir, no opera en una sola máquina de gran capacidad" [48].

1.3.4. ARQUITECTURA

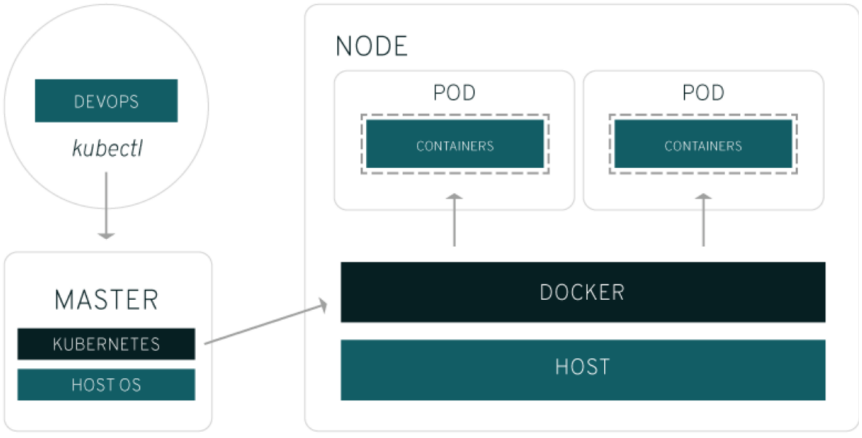


Figura 6. Arquitectura de Kubernetes. Fuente [49].

"Kubernetes se ejecuta en un sistema operativo e interactúa con pods de contenedores que se ejecutan en los nodos. El master de Kubernetes posee comandos de administrador y de esta manera los a los nodos subordinados. Esta transmisión se puede utilizar con una gran cantidad de servicios para determinar automáticamente qué nodo es el que mejor se adapta a la tarea. Luego asigna los recursos y los pods en ese nodo para completar la tarea requerida. Los pods abstraen la red y el almacenamiento del contenedor subyacente. Esto facilita mover los contenedores en el clúster" [5].

Los elementos de la arquitectura de Kubernetes son los siguientes:

- **Contenedor:** contiene apps y entornos de software [5].
- **Pod:** es responsable de agrupar contenedores que requieren operar juntos para ejecutar la aplicación [5].
- **Nodo:** uno o más pods corren en un nodo. Estos nodos son administrados y controlados por el maestro de k8s. Está constituido por kubelet el cual se encarga de establecer la comunicación con el maestro y por kube-proxy, que se encarga del balanceo de carga y permite conectividad de red a través de TCP u otros protocolos [5].
- **Master:** es un servidor que garantiza la administración y gestión de los nodos, es el encargado de coordinar el clúster, decidir en qué nodo se va a ejecutar cada contenedor y actualizar aplicaciones de forma coordinando cuando se despliegan nuevas versiones [50].
- **Clúster:** los nodos se asocian en clústeres [5].

1.4. SERVICIOS WEB

1.4.1. DEFINICIÓN

Los servidores web conocidos también como servidores HTTP permiten guardar información de Internet y de esta manera proveer una disponibilidad constante y segura. Por lo general las empresas tienen sus propios servidores web para que su contenido se pueda utilizar en la Intranet e Internet. Así mismo, los servidores web como parte de una red de ordenadores, transmiten documentos a los clientes, páginas web a un navegador [51].

1.4.2. CARACTERÍSTICAS

Un servidor web posee las siguientes características.

- **A nivel de Software**
  - ✓ **Sistema operativo (Linux, Unix o Windows**): garantiza que el software opere correctamente y consiga comunicarse con servicios que se ejecutan en él [52].
  - ✓ **Sistemas de archivos:** permite al sistema localizar, ordenar y filtrar datos en el disco duro para poder leer, modificar o borrar los mismos [52].
  - ✓ **Software servidor HTTP:** son servidores web como Apache, Nginx, IIS, Caddy, etc., dedicados a transferir contenido vía web [52].
  - ✓ **Virtual Hosting:** permite alojarse en sitios web diferentes bajo el mismo servidor web e IP [52].
  - ✓ **Despacho de ficheros estáticos y dinámicos:** proporcionan soporte para alojar y despachar archivos [52].
  - ✓ **Monitoreo de red y límites:** monitorea el tráfico de red, los paquetes de datos entrantes y salientes, los servicios del sistema y el uso de hardware [52].
  - ✓ **Sistema de seguridad**: impone restricciones de acceso a cada dirección IP, permite o niega la entrada a determinados registros o URL, solicita la autenticación HTTP, filtra solicitudes inseguras y soporta distribución de información encriptada con certificados de seguridad SSL a través de HTTPS [52].
- **A nivel de hardware:**
  - ✓ **Rack y gabinete**: rack se refiere a la ubicación donde se coloca físicamente el servidor y gabinete es el marco que soporta componentes de hardware [52].
  - ✓ **CPU:** centro de procesamiento de datos del servidor [52].
  - ✓ **Memoria RAM:** para almacenar de manera temporal datos e información y datos según la necesidad de los usuarios a través del SO [52].
  - ✓ **Unidades de almacenamiento:** “el almacenamiento de servidores web se hace en discos duros” [52].

- ✓ **Puerto de red:** el ancho de banda brinda suficiente capacidad para transferir información desde y hacia el servidor web [52].

1.4.3. FUNCIONAMIENTO

Para que el servidor web funcione correctamente, necesitamos un servidor para almacenar información y un cliente web que envíe solicitudes http o https a través de un navegador [53].

PROCESO

- Luego de que el usuario consulte el sitio web se establece una conexión entre el servidor DNS y el ordenador que emitió la consulta y de esta manera el servidor DNS responde con la dirección IP correcta del servidor web que aloja la información solicitada [53].
- Solicitar contenido del servidor web mediante el protocolo HTTP o HTTPS [53].
- Cuando el servidor web recibe la solicitud enviada por el cliente web, se procesa la solicitud hasta que se encuentre el contenido solicitado en el dominio correspondiente [53].
- Se envía el contenido solicitado al cliente web que lo solicitó [53].

1.4.4. TIPOS DE SERVIDORES WEB

Entre los tipos de servidores más utilizados podemos encontrar servidores como Apache, Nginx, LiteSpeed e IIS. Los tres primeros se utilizan en SO Linux mientras que IIS se usa en entornos Windows [54].

Apache: es un servidor de código abierto más usado en el mundo, aunque ha perdido popularidad frente a competidores como Nginx o IIS. Sin embargo, sigue siendo un servidor robusto, seguro, eficiente y fácil de configurar [54].

Apache admite PHP como lenguaje de programación

“Una de las ventajas de PHP es su potente API, que puede conectarse a servidores de bases de datos como PostgreSQL y MySQL” [55].

1.5. SERVICIO DE BASE DE DATOS

1.5.1. MYSQL

Es una plataforma que gestiona las DB principalmente desplegado por su rendimiento y fácil gestión. Como dato adicional este sistema no posee características en otros gestores SGBD, pero es una alternativa a considerar tanto para apps comerciales, como para proyectos beta, esto gracias a su facilidad de implementación al poner en producción dicha aplicación, cabe recalcar que su distribución se lo realiza bajo la licencia GPL [56].

1.5.2. CARACTERÍSTICAS

- Desarrollado en el lenguaje de programación C/C++ [56].
- Tienen ejecutables diecinueve plataformas diferentes [56].
- Su api está disponible en C, C++, Eiffel, Java, Perl, PHP, Python, Ruby y TCL [56].
- Apto para equipos con múltiples procesadores [56].
- Tiene un alto rendimiento, desempeño y estabilidad [56].
- Funciona como cliente-servidor o desplegado en aplicaciones [56].
- Gran variedad de tipos de datos [56].
- Con una administración basada en usuarios y privilegios [56].
- Puede gestionar hasta 50 millones de registros, 70 mil tablas y 5 millones de columnas [56].
- Conectividad TCP/IP, sockets UNIX y sockets NT [56].

CAPÍTULO 2: TOPOLOGÍA PROPUESTA

En el siguiente capítulo se detalla la topología propuesta, así como también los recursos necesarios tanto de hardware como de software.

2.1. FACTORES FUNCIONALES

REDES VIRTUALES

Las redes virtuales tienen como función principal conectar diferentes dispositivos o máquinas virtuales independientemente de su ubicación, para que estos funcionen con los mismos recursos como si estuvieran conectados a través de redes físicas.

Esto como consecuencia brinda la posibilidad de que los data center estén ubicados en diferentes lugares, así mismo haciendo su administración de redes más eficiente, es decir el administrador de red tiene la capacidad de crear, actualizar o modificar la red sin ninguna limitación de hardware, el aprovisionamiento de la red se vuelve más sencillo, dependiendo netamente de la demanda, con aplicaciones, cargas de trabajo en la red, sin que este afecte la alta disponibilidad, seguridad y servicio.

Dicho esto, en este proyecto se ha desplegado una red virtual interna e IPs flotantes para el servicio de aplicaciones, cuyo objetivo es interconectar todos los servicios alojados dentro de OpenStack con la red local y principalmente con la Internet.

SERVIDOR DOCKER

Este servidor se desplegará en la red interna creada en OpenStack. A partir de este servidor automatizamos el despliegue de nuestras aplicaciones dentro de contenedores, para luego gestionarlos con la misma plataforma, o con la más conocida Kubernetes.

DOCKER HUB

Este repositorio de igual manera será desplegado conjuntamente con los servicios de Docker y Kubernetes, es decir dentro de la red interna de OpenStack.

Este servicio nos ayudará con las pruebas de migración de nuestra aplicación entre diferentes plataformas de nube que para este proyecto son OpenStack y AWS, en este caso la aplicación será creada dentro de OpenStack.

SERVIDOR DE KUBERNETES

Este servidor permitirá la gestión de contenedores dentro de nuestra nube pública, por lo que para una administración, integración y comunicación centralizada se instalará tanto en la nube privada OpenStack y nuble pública AWS.

2.2. TOPOLOGÍA

En esta sección detallaremos la topología propuesta para el despliegue de una nube híbrida para entornos de desarrollo implementando Kubernetes y Dockers en la plataforma OpenStack e infraestructura en Amazon.

Para el despliegue de la nube híbrida utilizaremos dos herramientas como infraestructura, tanto la plataforma abierta de OpenStack, como la plataforma propietaria de Amazon (AWS), así mismo dentro de estas plataformas se instalarán los servicios de Docker y Kubernetes.

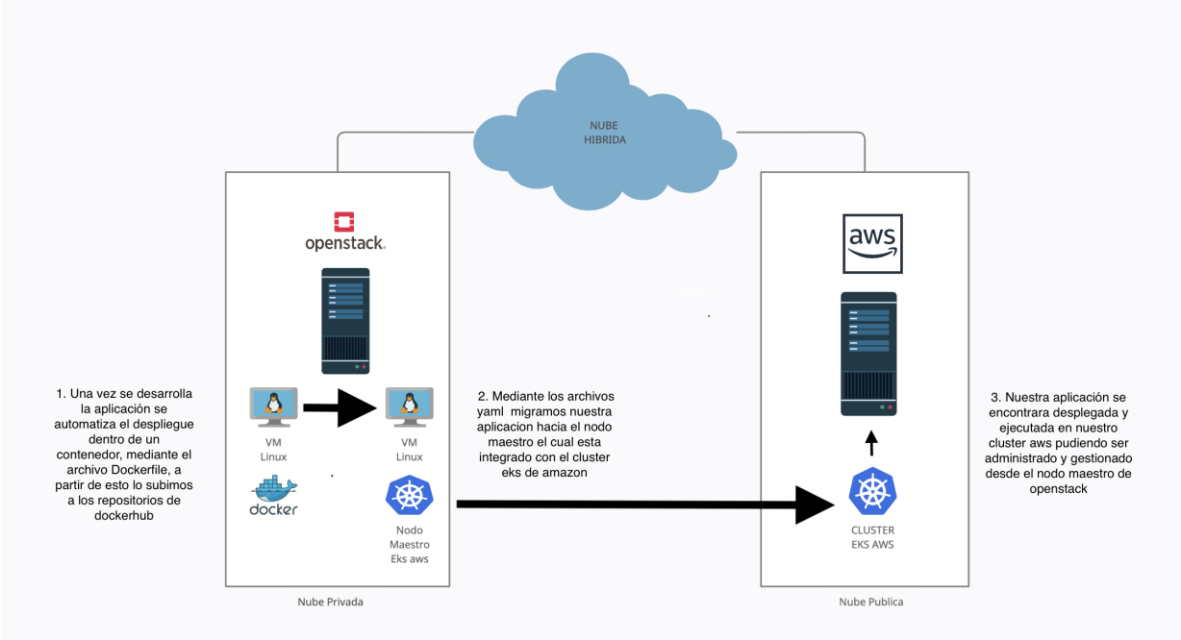


Figura 7. Topología de la infraestructura. Fuente Autor

Esta topología demuestra cómo interactúa físicamente toda la infraestructura para la integración de la nube privada con la nube pública mediante Kubernetes, para la plataforma de OpenStack hemos desplegado una máquina todo en uno, donde el cómputo, la red, y el almacenamiento serán administrados por la misma máquina, dentro de OpenStack tenemos desplegado dos instancias (VM), en la primera instancia tenemos corriendo el servidor de Docker y en la segunda instancia tenemos corriendo el servidor maestro de Kubernetes el cual se integrará con el clúster EKS que se encuentra en la nube de AWS, así mismo, con la instancia del servidor Docker, se creará unos archivos yaml y Dockerfile, mismos que se podrán ejecutar con la ayuda de los servidores mencionados anteriormente, en base a la configuración de estos script nos permitirá una migración completa del aplicativo de un ambiente de nube privado a un público.

A continuación, se muestra la topología de la red en OpenStack , para esto utilizamos una red interna, una red externa e IPs flotantes, donde en la red externa nos permitirá tener una conexión a la red local física, así mismo, la red interna mediante un router nos permitirá conectarnos hacia la red externa creada en OpenStack y a partir de esta tener una salida a la Internet, pero para comunicarnos desde y hacia afuera de una instancia de OpenStack necesitamos IPs flotantes, la

cual es asignada de forma dinámica, exclusivamente para nuestras instancias las mismas en las que desplegaron los servidores.

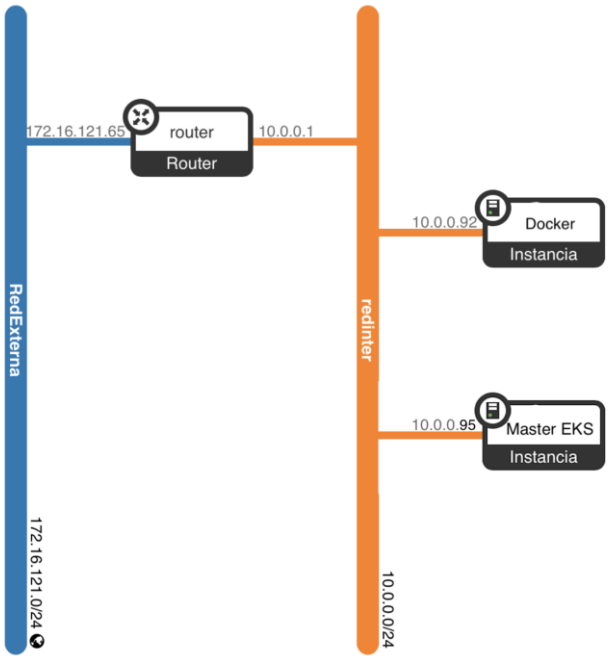


Figura 8. Topología de red OpenStack. Fuente Autor

A partir de esto, detallamos los recursos necesarios tanto en hardware y software que se utilizó para la arquitectura antes mencionada.

2.3. RECURSOS DE HARDWARE Y SOFTWARE UTILIZADOS

Para el despliegue se utilizó un ordenador con los siguientes recursos de hardware.

Marca	Modelo	Hipervisor	RAM	Disco HHD	Tipo de Procesador	Procesadores
Acer	Aspire 5 A515-51-89UP	VMware Workstation	20 GB	1 TB	Core i7	4

Tabla 1. Características del equipo físico. Fuente Autor

2.3.1. RECURSOS DE HARDWARE UTILIZADOS EN EL DESPLIEGUE DE OPENSTACK

Los recursos utilizados para la plataforma de OpenStack son los siguientes:

Sistema Operativo	RAM	Procesador	Disco duro
CentOS 7	14 GB	6 procesadores	100 GB

Tabla 2. Características físicas de OpenStack. Fuente Autor

2.3.2. RECURSOS DE AMAZON WEB SERVICES (AWS)

Las características utilizadas en AWS son relativamente altas puesto que los requerimientos mínimos para que un clúster de Kubernetes sea funcional por máquina es de: 2,5 de RAM 20 GB de disco y 2 procesadores, cabe mencionar que esta plataforma al ser propietaria es de pago, pero los costos no varían por los recursos utilizados sino más bien por el tiempo que se los tenga en funcionamiento.

Sistema Operativo	RAM	Procesador	Disco duro
Ubuntu 16.04 2	9 GB	6 procesadores	60 GB

Tabla 3. Características de AWS. Fuente Autor

CAPÍTULO 3: MARCO METODOLÓGICO

Esta sección describe la metodología implementada con la finalidad de cumplir con los objetivos establecidos.

3.1. METODOLOGÍA

La metodología SCRUM será la metodología empleada en este proyecto, dado que es una metodología ágil que permite realizar interacciones entre los miembros del equipo y avanzar de manera exitosa en el proyecto para obtener mejores resultados.

Las etapas que permitirán cumplir con los objetivos propuestos son tres, en las mismas se detalla el proceso para el desarrollo del proyecto.

3.2. ETAPA 1: REQUERIMIENTOS Y DISEÑO DE LA ARQUITECTURA DEL SISTEMA DE CLOUD

Se realiza un análisis exhaustivo para la elaboración del documento de la especificación de requerimientos y el diseño de la arquitectura del sistema de Cloud.

Los requerimientos para la implementación del sistema de Cloud híbrido se encuentran en las tablas 1, 2, 3 situadas en el capítulo 2.

De igual modo, la arquitectura desplegada para implementar el diseño de Cloud híbrido se muestra en la Figura 7.

3.3. ETAPA 2: DESARROLLO

En esta etapa se dará a conocer el proceso de configuración de la plataforma de OpenStack y Amazon Web Services, con los protocolos necesarios para una integración completa entre las dos plataformas de Cloud, basado en Dockers y Kubernetes.

3.3.1. CONFIGURACIÓN DE OPENSTACK

Para instalar y configurar la plataforma de Cloud privado en la infraestructura de OpenStack, es necesario ejecutar los siguientes comandos.

Los resultados de la instalación y configuración se encuentran en el Anexo 1.

Una vez que se haya desplegado OpenStack , verificamos que los ficheros de configuración se encuentren en el directorio root, estos nos permitirán el acceso a OpenStack , entre la lista de archivos debe estar un archivo llamado keystonerc_admin, el cual nos permite el acceso a OpenStack , este archivo contiene las variables que se pasan al sistema para poder acceder a OpenStack , es decir, podemos conocer tanto el usuario como la contraseña para acceder a OpenStack mediante Dashboard, esto es, haciendo uso de un navegador, mismo que se conecta a un servidor Apache y ejecuta el componente de Horizon, este a su vez permite administrar los elementos de OpenStack de manera más sencilla debido a que se pueden visualizar. Así mismo, se puede acceder a OpenStack de una forma común a través del CLI (Interfaz de Línea de Comandos), para ello , se debe pasar al sistema todas las variables que contiene el archivo keystonerc_admin con el comando `source keystonerc_admin` y a continuación se podrá ejecutar cualquier comando que empiece con OpenStack, sin embargo, existen otros comandos disponibles, por ejemplo, para trabajar con componentes de keystone utilizamos el comando keystone y para trabajar con la red usamos el comando neutrón.

El módulo Neutrón controlará todas las redes, ya que este servicio nos permite crear grupos de seguridad para controlar los puertos de acceso a una instancia lanzada dentro de OpenStack, del mismo modo, permite crear las redes internas y externas con sus respectivas subredes. Además, se puede crear routers y conectarlos a las redes externas e internas. Este módulo puede ser gestionado tanto en Dashboard como en CLI.

Con el módulo Heat, que es el módulo de orquestación de OpenStack, se lanza las instancias en base a un código escrito. Por otra parte, el servicio Magnum, administra la orquestación de

contenedores Kubernetes, Docker Swarm y Apache Mesos, es decir, Magnum utiliza el módulo Heat para orquestar una imagen del sistema operativo que contiene Docker y Kubenetes.

3.3.2. CONFIGURACIÓN EN AMAZON WEB SERVICES (AWS)

En AWS vamos a desplegar el servicio EKS que es equivalente a un clúster de Kubernetes, para ello se requiere realizar algunas configuraciones, es importante recalcar que montar el clúster en AWS requiere un costo puesto que estas configuraciones y funcionalidades son propietarias.

Lo primero que hacemos es crear y configurar toda la red dentro de AWS, para esto vamos a crear una Virtual Private Cloud (VPC) y las subredes, para hacerlo de manera sencilla utilizamos el servicio de CloudFormation, el cual permite gestionar y trabajar con la infraestructura a través de plantillas para crear todo el modelo de red con las configuraciones correctas que se necesita para trabajar con el clúster.

El resultado de la configuración se encuentra en el Anexo 2.

3.3.3. CONFIGURACIÓN DE DOCKER

Se debe instalar la infraestructura de Docker en la instancia lanzada en OpenStack, para posteriormente usarlo en el proceso de migración de la aplicación.

El manual completo de la instalación de Docker se encuentra en el Anexo 3.

3.3.4. INSTALACIÓN Y CONFIGURACIÓN DE KUBERNETES

El manual de la instalación y configuración de Kubernetes en OpenStack se encuentra en el Anexo 4.

Dado que en la instancia lanzada en OpenStack no fue posible crear el clúster de Kubernetes debido a que al ejecutar el comando `kubeadm init --pod-network-cidr=192.168.0.0/16`, se presentó el error que se muestra en la figura 9, de modo que, se

procedió a desplegar la máquina virtual del master del clúster de Kubernetes fuera de OpenStack y los nodos dentro de OpenStack donde la configuración se realizó de manera satisfactoria.

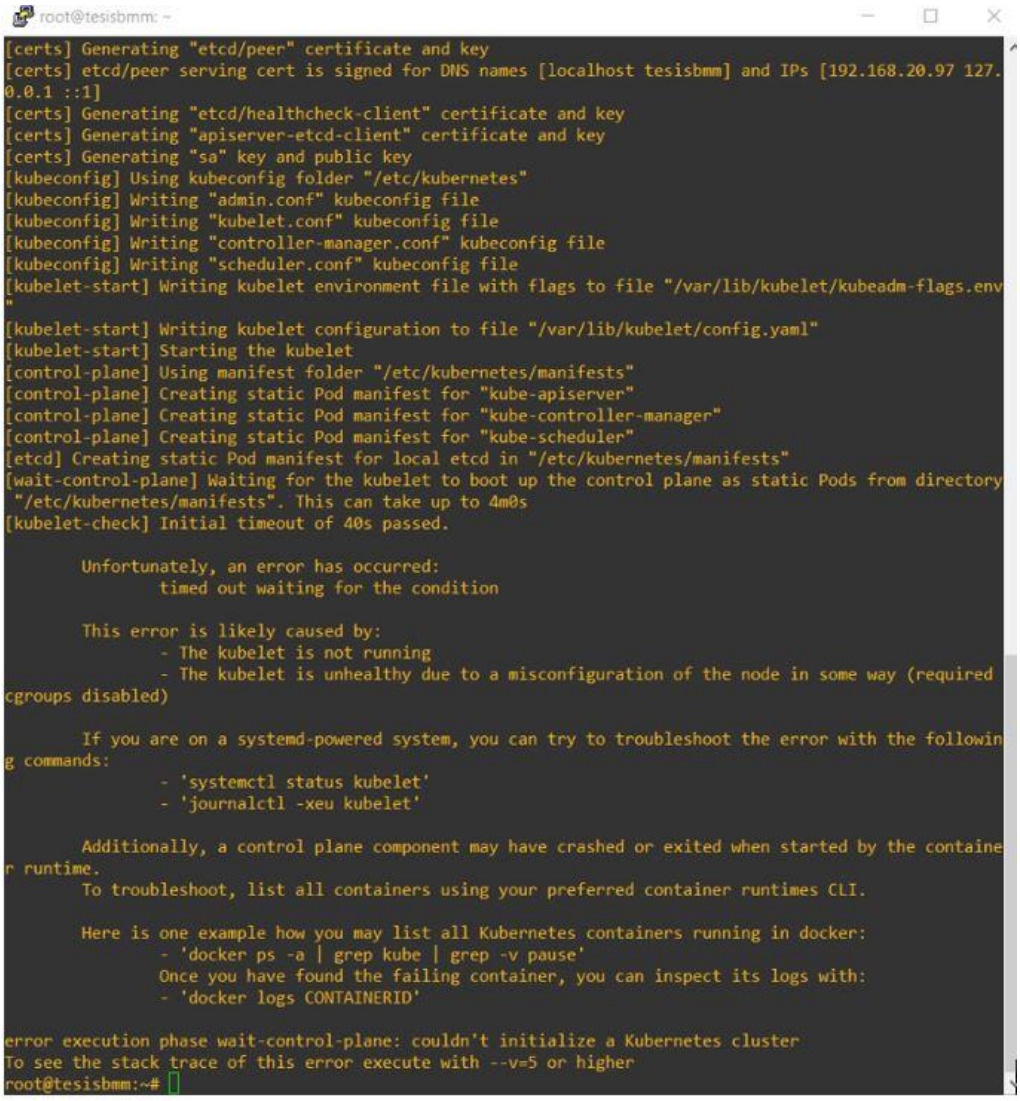


Figura 9. Error al crear el clúster de Kubenetes en OpenStack. Fuente Autor

Entonces, al momento de añadir estos nodos al máster se vincularon, pero no de manera exitosa, ya que una vez creados los pods no arrancan de manera exitosa como se puede observar en la figura 10, donde tenemos una respuesta 0/1 respecto al estado de los pods, puesto que, al ejecutar el comando `kubectl describe pod calico-node-9fx6n,` se visualiza el error que se muestra en la figura 11.

```
root@tesisbmm:~# kubectl get pods -n kube-system -o wide
NAME                                       READY   STATUS    RESTARTS   AGE   IP               NODE        NOMINATED NODE   READINESS GATES
calico-kube-controllers-6b8f6f78dc-x47p2   1/1     Running   0          20m   192.168.151.195  tesisbmm    <none>           <none>
calico-node-9fx6n                          0/1     Running   0          18m   192.168.20.241   tesisbmn1   <none>           <none>
calico-node-9v5rl                          0/1     Running   0          20m   192.168.16.240   tesisbmm    <none>           <none>
coredns-f9fd979d6-26qmn                    1/1     Running   0          21m   192.168.151.194  tesisbmm    <none>           <none>
coredns-f9fd979d6-csvzw                    1/1     Running   0          21m   192.168.151.193  tesisbmm    <none>           <none>
etcd-tesisbmm                              1/1     Running   0          22m   192.168.16.240   tesisbmm    <none>           <none>
kube-apiserver-tesisbmm                    1/1     Running   0          22m   192.168.16.240   tesisbmm    <none>           <none>
kube-controller-manager-tesisbmm           1/1     Running   0          22m   192.168.16.240   tesisbmm    <none>           <none>
kube-proxy-ctk8c                           1/1     Running   0          21m   192.168.16.240   tesisbmm    <none>           <none>
kube-proxy-l921p                           1/1     Running   0          18m   192.168.20.241   tesisbmn1   <none>           <none>
kube-scheduler-tesisbmm                    1/1     Running   0          22m   192.168.16.240   tesisbmm    <none>           <none>
```

Figura 10. Error del estado de preparación de los pods. Fuente Autor

```
  Optional:    false
QoS Class:     Burstable
Node-Selectors:  beta.kubernetes.io/os=linux
Tolerations:     :NoScheduleop=Exists
                 :NoExecuteop=Exists
                 CriticalAddonsOnly op=Exists
                 node.kubernetes.io/disk-pressure:NoSchedule op=Exists
                 node.kubernetes.io/memory-pressure:NoSchedule op=Exists
                 node.kubernetes.io/network-unavailable:NoSchedule op=Exists
                 node.kubernetes.io/not-ready:NoExecute op=Exists
                 node.kubernetes.io/pid-pressure:NoSchedule op=Exists
                 node.kubernetes.io/unreachable:NoExecute op=Exists
                 node.kubernetes.io/unschedulable:NoSchedule op=Exists
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  18m                default-scheduler  Successfully assigned kube-system/calico-node-9fx6n to tesisbmn1
  Normal   Pulled     18m                kubelet            Container image "calico/cni:v3.11.3" already present on machine
  Normal   Created    18m                kubelet            Created container upgrade-ipam
  Normal   Started    18m                kubelet            Started container upgrade-ipam
  Normal   Pulled     17m                kubelet            Container image "calico/cni:v3.11.3" already present on machine
  Normal   Created    17m                kubelet            Created container install-cni
  Normal   Started    17m                kubelet            Started container install-cni
  Normal   Pulled     17m                kubelet            Container image "calico/pod2daemon-flexvol:v3.11.3" already present on machine
  Normal   Created    17m                kubelet            Created container flexvol-driver
  Normal   Started    17m                kubelet            Started container flexvol-driver
  Normal   Pulled     17m                kubelet            Container image "calico/node:v3.11.3" already present on machine
  Normal   Created    17m                kubelet            Created container calico-node
  Normal   Started    17m                kubelet            Started container calico-node
  Warning  Unhealthy  16m                kubelet            Readiness probe failed: calico/node is not ready: BIRD is not ready: BGP not established with 192.168.16.2402020-10-13 02:50:17.530 [INFO][179] hea
th.go 156: Number of node(s) with BGP peering established = 0
  Warning  Unhealthy  16m                kubelet            Readiness probe failed: calico/node is not ready: BIRD is not ready: BGP not established with 192.168.16.2402020-10-13 02:50:28.498 [INFO][214] hea
th.go 156: Number of node(s) with BGP peering established = 0
  Warning  Unhealthy  16m                kubelet            Readiness probe failed: calico/node is not ready: BIRD is not ready: BGP not established with 192.168.16.2402020-10-13 02:50:39.697 [INFO][247] hea
th.go 156: Number of node(s) with BGP peering established = 0
  Warning  Unhealthy  15m                kubelet            Readiness probe failed: calico/node is not ready: BIRD is not ready: BGP not established with 192.168.16.2402020-10-13 02:50:50.665 [INFO][272] hea
th.go 156: Number of node(s) with BGP peering established = 0
  Warning  Unhealthy  15m                kubelet            Readiness probe failed: calico/node is not ready: BIRD is not ready: BGP not established with 192.168.16.2402020-10-13 02:51:02.039 [INFO][298] hea
th.go 156: Number of node(s) with BGP peering established = 0
  Warning  Unhealthy  15m                kubelet            Readiness probe failed: calico/node is not ready: BIRD is not ready: BGP not established with 192.168.16.2402020-10-13 02:51:14.321 [INFO][331] hea
th.go 156: Number of node(s) with BGP peering established = 0
  Warning  Unhealthy  15m                kubelet            Readiness probe failed: calico/node is not ready: BIRD is not ready: BGP not established with 192.168.16.2402020-10-13 02:51:23.926 [INFO][358] hea
th.go 156: Number of node(s) with BGP peering established = 0
  Warning  Unhealthy  15m                kubelet            Readiness probe failed: calico/node is not ready: BIRD is not ready: BGP not established with 192.168.16.2402020-10-13 02:51:39.667 [INFO][383] hea
th.go 156: Number of node(s) with BGP peering established = 0
  Warning  Unhealthy  14m                kubelet            Readiness probe failed: calico/node is not ready: BIRD is not ready: BGP not established with 192.168.16.2402020-10-13 02:51:47.817 [INFO][410] hea
th.go 156: Number of node(s) with BGP peering established = 0
  Warning  Unhealthy  3m13s (x59 over 14m)  kubelet         (combined from similar events): Readiness probe failed: calico/node is not ready: BIRD is not ready: BGP not established with 192.168.16.2402020-10-
13 03:03:31.644 [INFO][1999] health.go 156: Number of node(s) with BGP peering established = 0
```

Figura 11. Detalle del error del estado de preparación de los pods. Fuente Autor

Por todo esto, para realizar la integración y cumplir con los objetivos establecidos en nuestra infraestructura híbrida, se procede a configurar el clúster de Kubernetes en AWS y se crea la instancia para el nodo maestro en OpenStack, esta instancia permite conectamos a nuestro clúster EKS de AWS, de manera que se logra la integración de estos entornos mediante Kubernetes.

El proceso de configuración de Kubernetes en AWS se encuentra en el Anexo 5.

Es necesario recalcar que, se instala 3 componentes básicos de cualquier clúster de Kubernetes que son kubelet kubeadm kubectl, con la finalidad de instalar herramientas que permitan gestionar el clúster.

Ahora bien, una vez creado el clúster en AWS, deberá estar en modo activo, si esto es así, se procede a crear los esclavos o nodos.

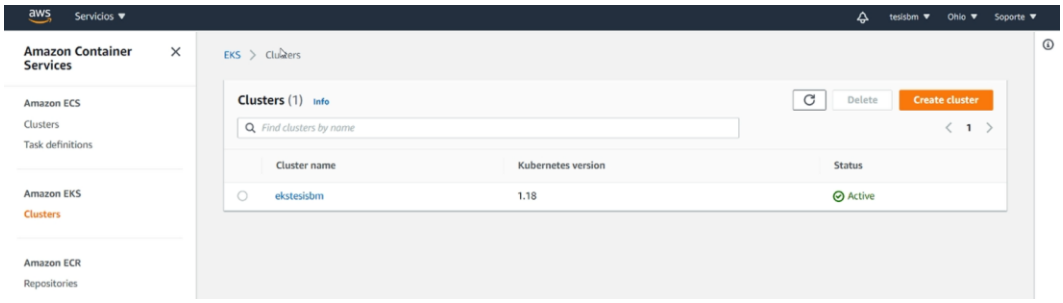


Figura 12. Clúster creado en estado Activo. Fuente Autor.

Luego se procede a crear los nodos, una vez creado correctamente el grupo de nodos, verificamos que esté activo.

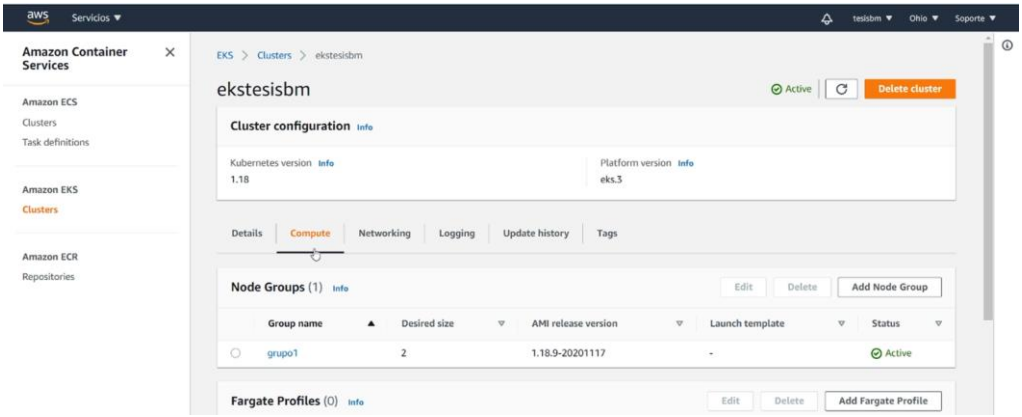


Figura 13. Grupo de nodos creado y en estado Activo. Fuente Autor.

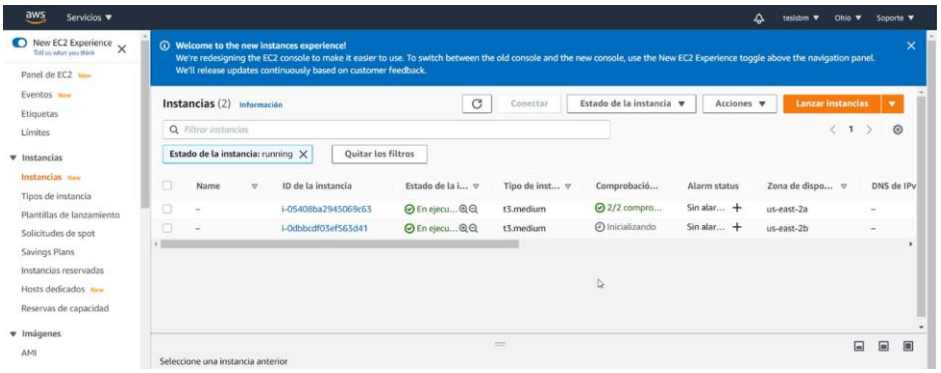


Figura 14. Nodos creados en AWS. Fuente Autor.

3.3.5. INTEGRACIÓN DE AWS Y OPENSTACK MEDIANTE KUBERNETES

Para realizar la integración, desde el nodo maestro de Linux lanzado en la infraestructura de OpenStack nos conectamos a nuestro clúster EKS, para ello, debemos descargar AWS CLI, para configurar la conexión contra el clúster creado en AWS.

Vamos a instalar AWS CLI versión 2 en la instancia de Linux x86 (64-bit) lanzada en OpenStack, esta información podemos encontrarla en la documentación de Amazon.

Como usuario root, ejecutamos los siguientes pasos:

1. Para descargar el paquete de AWS CLI ejecutamos:
   ```
   curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
   ```
2. Descomprimimos
   ```
   unzip awscliv2.zip
   ```
3. Procedemos a instalar
   ```
   sudo ./aws/install
   ```
4. Para configurar el acceso ejecutamos el siguiente comando, de esta manera, vamos a acceder de manera remota al clúster para ello vamos a proporcionar unas credenciales.
   ```
   aws configure
   ```
5. En vista de que, solicita una AWS Access Key ID, nos dirigimos a la cuenta de AWS y seleccionamos Mis credenciales de seguridad, en Claves de acceso que son las que se usan para acceder desde AWS CLI, creamos una clave de acceso la misma que contiene el ID de clave de acceso y la clave de acceso secreta, se debe copiar el ID y la clave o también se puede descargar en la PC.

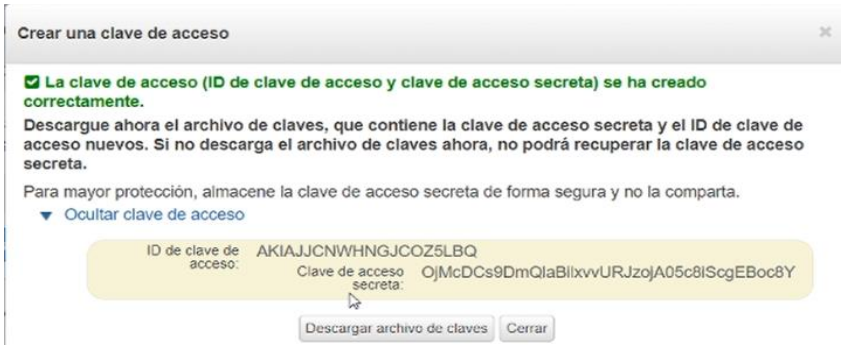


Figura 15. Crear clave de acceso. Fuente Autor

6. Solicita también el código de la región en la que se está trabajando con Amazon.

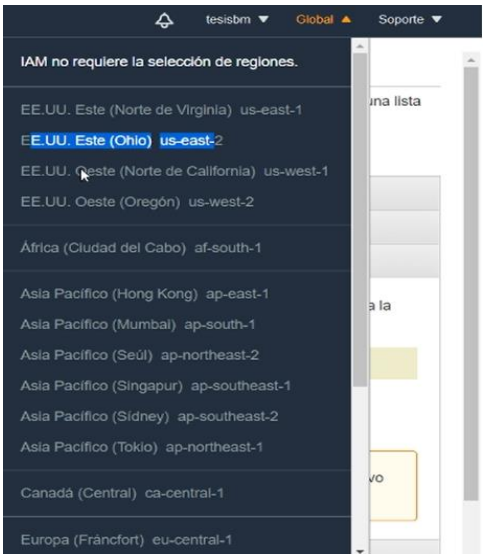


Figura 16. Código de la región. Fuente Autor.

7. Dentro de los comandos que existen en AWS, hay un comando que permite actualizar o generar el fichero CONFIG para kubectl.

   `aws eks --region us-east-2 update-kubeconfig --name ekstesisbm`

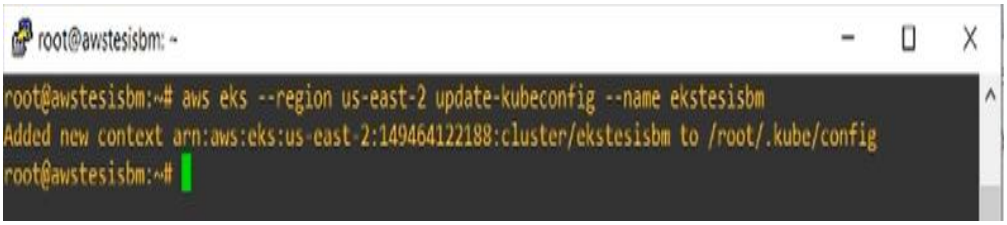


Figura 17. Comando para actualizar o generar el fichero config de kubectl. Fuente Autor.

8. En el directorio cd .kube ubicamos el fichero config mismo que contiene el certificado del clúster, además sitúa el server que debe coincidir con el API server endpoint y de la misma forma el clúster ARN.

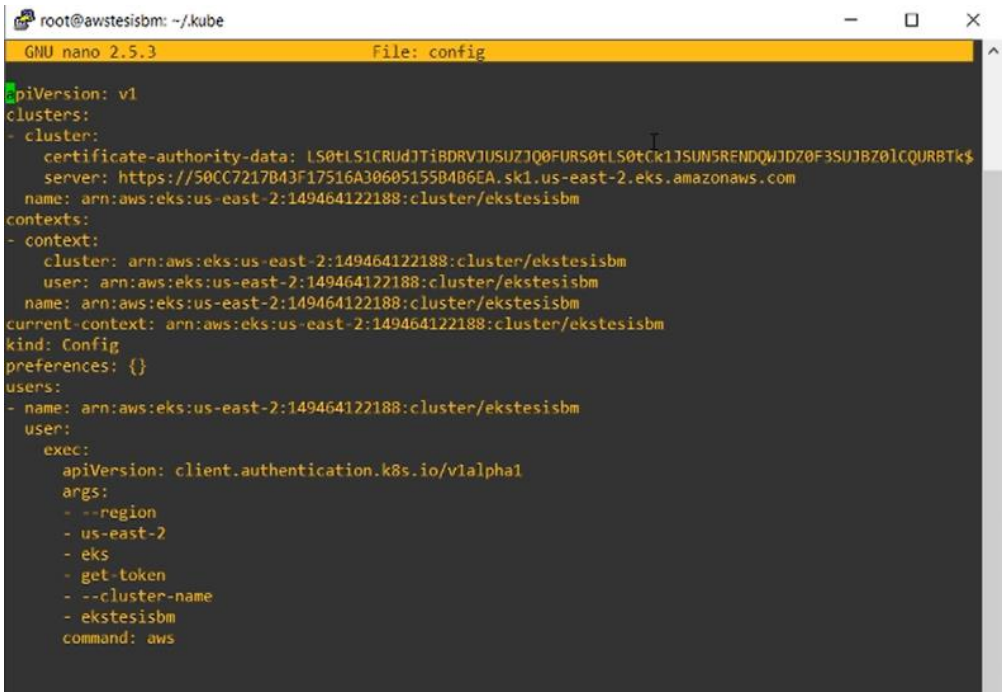


Figura 18. Archivo config. Fuente Autor.

3.3.6. CONFIGURACIÓN DEL ARCHIVO DOCKERFILE

El siguiente punto trata sobre los servicios, es decir, componentes que permiten que las aplicaciones que se encuentran en diferentes contenedores puedan interconectarse entre sí o ser expuestos al mundo exterior. En este caso, creamos un servicio con una aplicación basada en Apache, colocamos los ficheros Dockerfile para posteriormente crear la imagen y un directorio del proyecto web para desplegarla en el contenedor.

La estructura del archivo Dockerfile se encuentra en el Anexo 6.

En el archivo Dockerfile se indica que inicie el servicio de apache, que se copie todo el contenido del directorio prueba que contiene la aplicación web a la ruta por defecto de apache en el sistema operativo Ubuntu que es /var/www/html y finalmente exponer el puerto 80.

Con el Dockerfile vamos a construir la imagen con el comando

```
Docker build -t dhtesisbm/prueba1 .
```

Si todo está correcto, la imagen se construirá sin inconvenientes y para verificar que la imagen se haya creado ejecutamos el comando

```
Docker images
```

3.3.7. CONFIGURACIÓN DE LOS ARCHIVOS YAML

Los archivos yaml permiten construir recursos de Kubernetes, estos archivos son una forma de construir ficheros de properties que trabajan con socialización de datos, es decir, en base a estos archivos se puede construir ficheros de configuración legibles que se pueden aplicar en diferentes entornos, lenguajes y herramientas.

El fichero yaml, básicamente está constituido por componentes de tipo clave valor, donde se tiene la clave seguida de dos puntos y luego de un texto, esto posibilita la configuración de recursos dentro de Kubernetes.

Para establecer una cadena multilínea se realiza una identación con espacios en blanco respecto al principal, para crear secuencias respecto al principal, se establece una lista que empieza por un guión, también se pueden crear objetos complejos como modo árbol o modo anidado donde una propertie está formado por más properties.

Dicho lo anterior, se procede a crear el deployment que es uno de los componentes básicos de Kubernetes ya que mediante este objeto desplegamos proyectos. Los contenedores son los objetos más básicos de Kubernetes que para lanzarlos hay que colocarlos dentro de un pod que es un objeto mínimo que funciona dentro de Kubernetes, el pod puede tener uno o más contenedores, sin embargo, no son escalables, no se recuperan ante caídas y tampoco se puede realizar updates ni rollbacks, para eso se coloca los pods dentro de los deployments. Cuando se crea un deployment se crea un replicaset que es un componente encargado de hacer réplicas y de gestionar la recuperación ante caídas.

La estructura de nuestro archivo php-deployment.yaml se encuentra en el Anexo 7, además es de tipo LoadBalancer, debido a que este recurso normalmente se usa en Cloud.

3.4. ETAPA 3: PRUEBAS DE VALIDACIÓN

En esta etapa, se llevará a cabo la simulación básica transaccional de un Core bancario con la creación de escenarios reales y protocolos de pruebas para la validación de su correcto funcionamiento satisfaciendo los objetivos.

3.4.1. PRUEBA DE COMUNICACIÓN ENTRE OPENSTACK Y AWS

Para comprobar la integración entre ambas infraestructuras ejecutamos el comando

```
aws ec2 describe-instances
```

Como resultado devuelve un json con las instancias lanzadas en el cluster de Kubernetes en AWS.

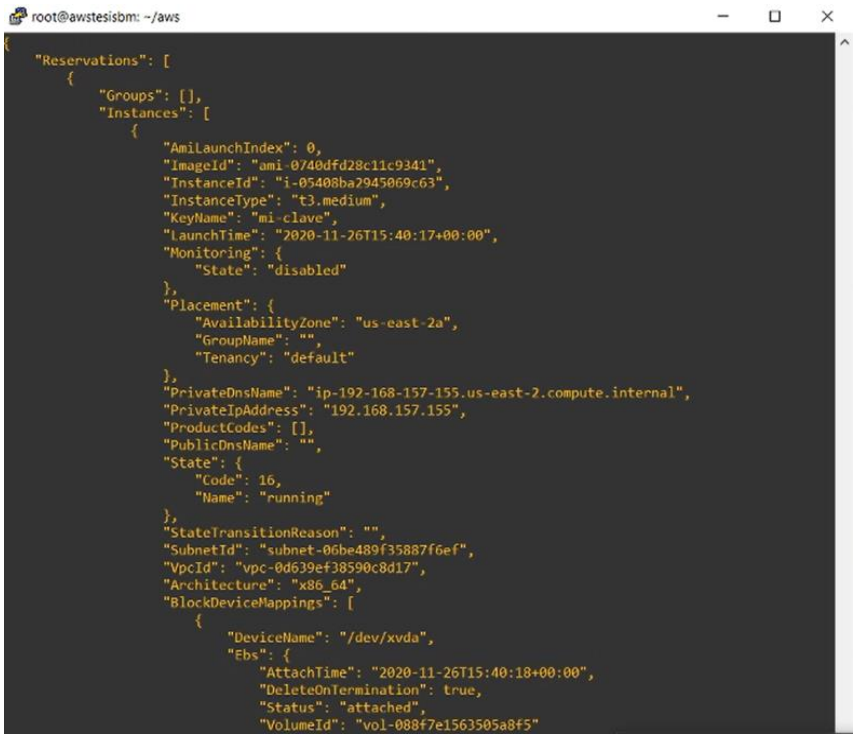


Figura 19. Comprobar la conexión con el clúster de AWS. Fuente Autor.

Comprobar el funcionamiento del *clúster* con el comando

```
kubectl cluster-info
```

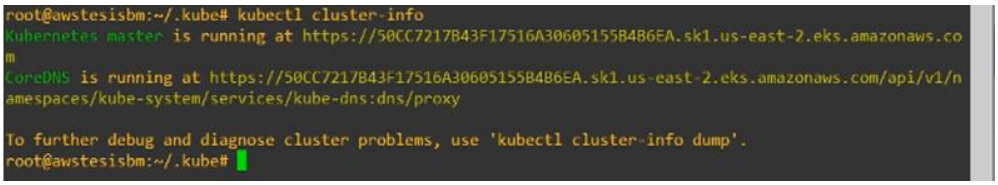


Figura 20. Comprobar funcionamiento del clúster desde kubectl. Fuente Autor.

Si todo está correctamente configurado, con el siguiente comando deberá mostrar los nodos lanzados en AWS, los mismos que deben estar ejecutándose.

```
kubectl get nodes
```

Figura 21. Nodos ejecutándose en AWS. Fuente Autor.

3.4.2. PRUEBA DE FUNCIONAMIENTO DEL CORE BANCARIO EN OPENSTACK

Descargamos la imagen de MySQL

```
docker pull mysql:5.6
```

Listamos las imágenes

```
docker images
```

Ejecutamos el contenedor de MySQL y asignamos una contraseña para el usuario root de MySQL con el siguiente comando:

```
docker run -p 3306:3306 --name mysql -v /home/tesisbmn/mysql:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=bdpasstesisbm -d mysql:5.6
```

Ejecutamos el contenedor de Apache

```
docker run -d -p 80:80 --name prueba --link mysql dhtesisbm/prueba1:latest
```

Para visualizar los Dockers creados ejecutamos

```
docker ps
```

Ingresamos al contenedor de MySQL

```
docker exec -it mysql bash
```

Accedemos a MySQL proporcionando la password configurada.

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

Creamos una base de datos

```
create database db_banco;
```

Listamos las bases de datos

```
show databases;
```

Usamos la base de datos creada

```
use db_banco;
```

Ahora vamos a probar que funcione la imagen creada en base al archivo Dockerfile, para ello ejecutamos el siguiente comando:

```
docker run -d -p 80:80 --name prueba --link mysql dhtesisbm/prueba1:latest
```

Verificamos con el comando:

```
docker ps
```

En el navegador apuntando a la IP, probamos que el contenedor esté funcionando correctamente, es decir, que accedemos a la página web de la aplicación.

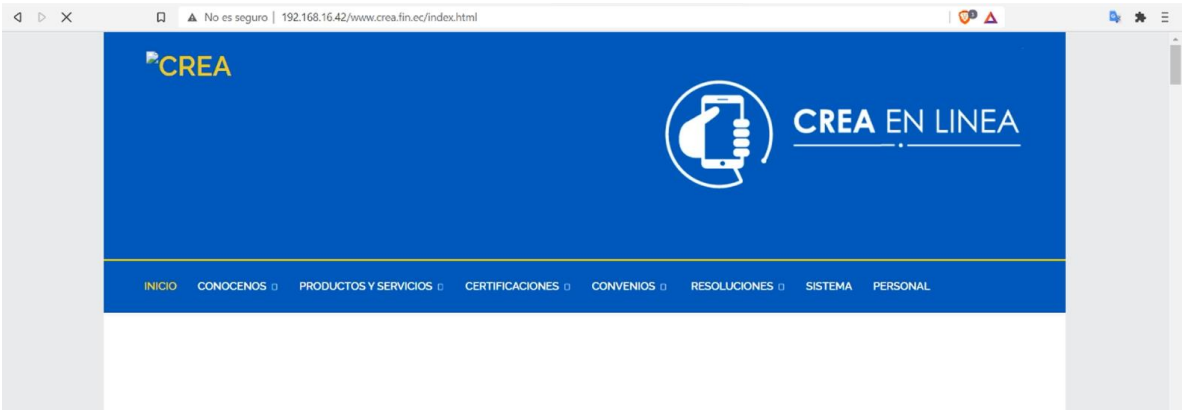


Figura 22. Contenedor web en OpenStack. Fuente Autor.

3.4.3. PRUEBA DE MIGRACIÓN DEL CORE BANCARIO DESDE OPENSTACK HACIA AWS

Como estrategia de migración del Core bancario de prueba desde OpenStack hacia AWS, vamos a subir nuestra imagen a Docker Hub, para luego poder usarla en el clúster de Kubernetes de AWS.

Vamos a loguearnos con el comando:

```
docker login
```

Debemos proporcionar las credenciales correctas para loguearnos.

Con el comando se prepara el repositorio y se sube a Docker hub.

```
docker push
```

Ahora nos dirigimos a Docker Hub y comprobamos que efectivamente se haya subido la imagen.

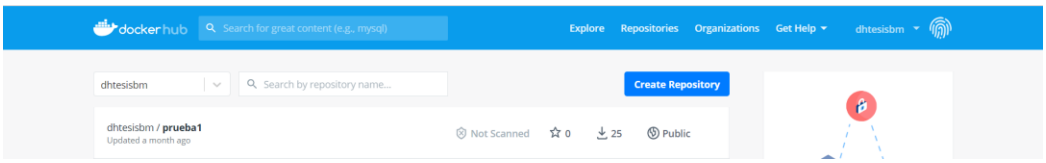


Figura 23. Imagen subida a Docker Hub. Fuente Autor.

Ejecutamos tres archivos yaml, para configurar la base de datos MySQL, para descargar y ejecutar la imagen de Docker Hub en el clúster de Kubernetes en AWS.

La configuración del archivo mysql-deployment.yaml se encuentra en el Anexo 8.

La configuración del archivo mysql-pv.yaml se encuentra en el Anexo 9.

Ejecutamos el archivo php-deployment.yaml con el comando:

```
kubectl apply -f php-deployment.yaml
```

Figura 24. Ejecución del archivo php-deployment.yaml. Fuente Autor

Ejecutamos el archivo mysql-pv.yaml

```
kubectl apply -f php-deployment.yaml
```

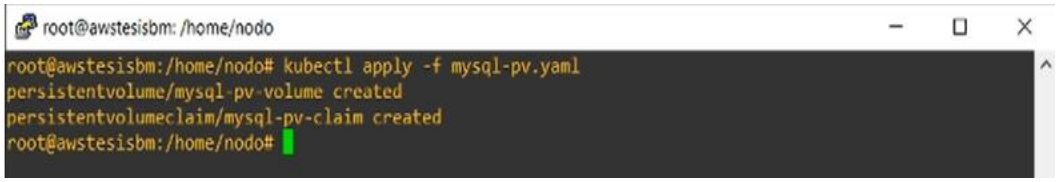


Figura 25. Ejecución del archivo mysql-pv.yaml. Fuente Autor.

Ejecutamos el archivo mysql-deployment.yaml

```
kubectl apply -f mysql-deployment.yaml
```

```
root@awstesisbm:/home/nodo# kubectl apply -f mysql-deployment.yaml
service/mysql created
deployment.apps/sqldb created
root@awstesisbm:/home/nodo#
```

Figura 26. Ejecución del archivo mysql-deployment.yaml. Fuente Autor.

3.4.4. PRUEBA DE FUNCIONAMIENTO DEL CORE MIGRADO

Verificamos que se estén ejecutando los pods

```
kubectl get pods
```

```
root@awstesisbm:/home/nodo# kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
sqldb-7548cc7fd5-b9sjf    1/1     Running   0          51s
webapp1-86d4887dfc-x4vkl  1/1     Running   0          3m7s
root@awstesisbm:/home/nodo#
```

Figura 27. Lista de pods. Fuente Autor.

Con el siguiente comando se puede ver que se asigna una EXTERNAL-IP que es una URL para exponer la aplicación web.

```
kubectl get svc
```

```
root@awstesisbm:/home/nodo# kubectl get svc
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP
             PORT(S)        AGE
kubernetes   ClusterIP      10.100.0.1       <none>
             443/TCP        97m
mysql        ClusterIP      None             <none>
             3306/TCP       114s
webapp-sql   LoadBalancer   10.100.211.246   af095fbd9b3d54bcaadbbab64b7a4ed5-29367856.us-east-2.elb.am
azonaws.com  80:30677/TCP   4m10s
```

Figura 28. URL para exponer la aplicación web. Fuente Autor.

En el navegador ingresamos la URL y deberá mostrar la aplicación web desplegada.

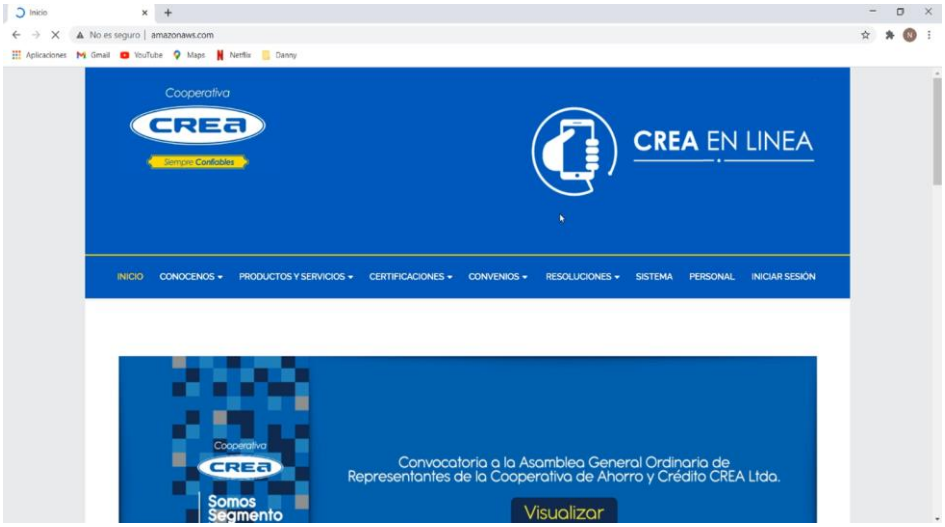


Figura 29. Aplicación web desplegada. Fuente Autor.

Ingresamos al pod de la base de datos con el comando

```
kubectl exec -it NOMBRE_DEL_POD bash
```

```
kubectl exec -it sqldb-7548cc7fd5-b9cjf bash
```

Luego ingresamos a MySQL proporcionando la password

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

Listamos las bases de datos

```
show databases;
```

Usamos la base creada

```
use db_banco;
```

CAPÍTULO 4: ANÁLISIS Y RESULTADOS

En base a las pruebas realizadas en el capítulo anterior, analizamos las características tanto en el desempeño general de la infraestructura como las condiciones que tienen que cumplirse para que la migración sea posible.

4.1. ANÁLISIS 1 DE PRUEBAS DEL ESCENARIO

Al realizar las pruebas de migración del Core bancario se analizó que los recursos disponibles en nuestro entorno se vieron limitados. En cuanto a OpenStack el rendimiento del CPU llega casi al límite de capacidad, debido a que los recursos asignados para toda la infraestructura son relativamente estándar, siendo su pico más alto de rendimiento un 74%.

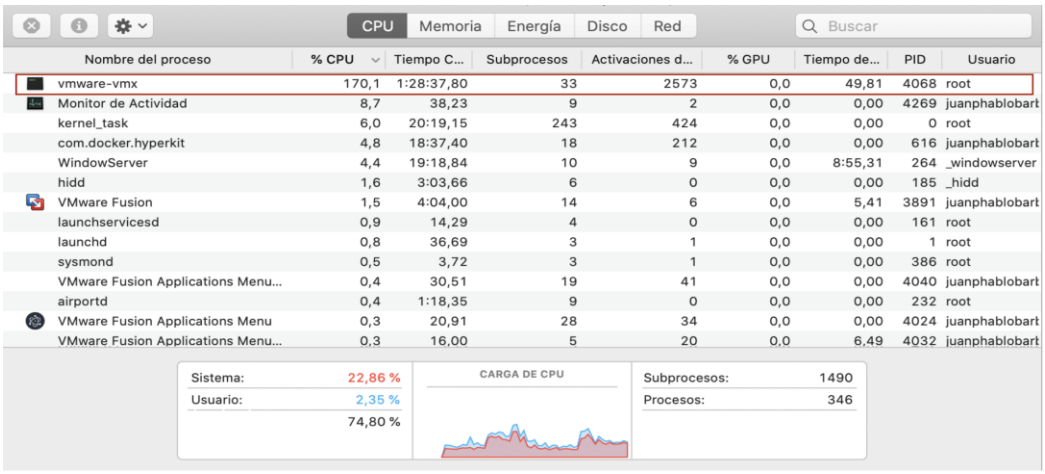


Figura 30. Demanda en recursos en CPU de la infraestructura de OpenStack montado en VMware. Fuente Autor.

En cuanto a la memoria RAM tenemos un uso de un 85% de usabilidad.

Figura 31. Demanda en recursos en RAM de la infraestructura de OpenStack montado en VMware. Fuente Autor.

En lo referente a la usabilidad del disco tenemos un 75% de uso y de la red tenemos un 15.2 MB consumido. A partir de este resultado vemos claramente que la plataforma de openstack no reporta problemas ante una saturación de trabajo, pero si nos vemos limitados por los recursos asignados, cabe mencionar que el sistema a migrar si es relativamente pesado, Por otra parte, al iniciar con el proceso de migración vemos que también que influye el ancho de banda disponible ya que si este es bajo el tiempo para completar dicha operación será relativamente alto.

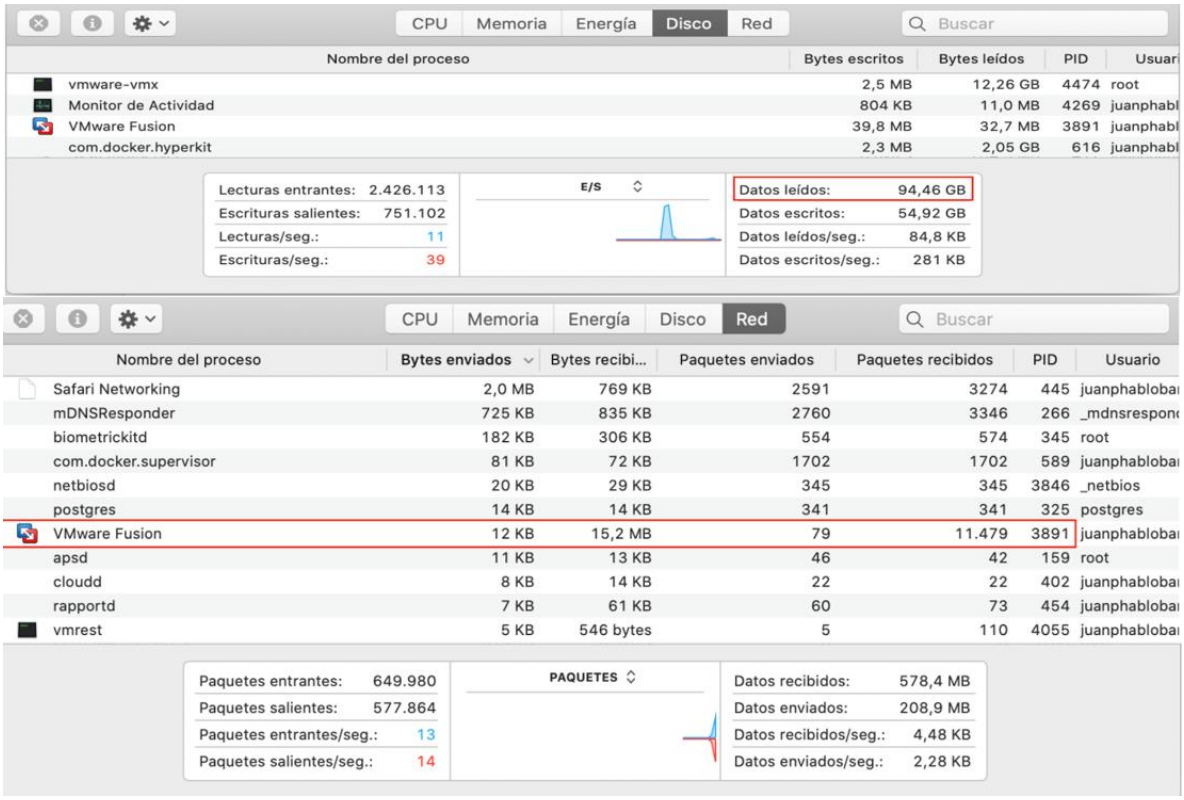


Figura 32. Demanda en recursos en disco y red de la infraestructura de OpenStack montado en VMware. Fuente Autor.

4.2. ANÁLISIS 2 DE PRUEBAS DEL ESCENARIO

Al momento de realizar las pruebas de migración del Core bancario de una entorno a otro el proceso se realiza con un 98% de efectividad ya que este puede variar por el ancho de banda y recursos disponibles en la infraestructura, pero las condiciones para que este proceso se realice son que tanto los comandos en Docker y los archivos yaml de Kubernetes, obtengan las mismas

versiones de complementos necesarios para que el contenedor arranque tanto en el ambiente de OpenStack y AWS con Kubernetes ejecutándose como se indica a continuación.

Nodo Openstack	Nodo EK-AWS
docker run -p 3306:3306 --name mysql -v /home/tesisbmn/mysql:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=bdpasstesisbm -d mysql:5.6	mysql-deployment.yaml spec: containers: - image: mysql:5.6 name: mysql

Tabla 4. Condiciones de Migración. Fuente Autor.

Migración Completa hacia AWS
root@awstesisbm:/home/nodo# kubectl get pods NAME READY STATUS RESTARTS AGE sqldb-7548cc7fd5-b9sjf 1/1 Running 0 51s webapp1-86d4887dfc-x4vkl 1/1 Running 0 3m7s root@awstesisbm:/home/nodo#

Tabla 5. Migración Satisfactoria. Fuente Autor.

CONCLUSIONES Y RECOMENDACIONES

En esta última sección terminaremos con las conclusiones y recomendaciones tomadas de las pruebas migración y replicación entre los entornos de nube privada y pública realizadas en los capítulos anteriores.

CONCLUSIONES

- Hemos logrado determinar que la integración de un Cloud público situado en AWS y un Cloud privado en OpenStack es posible, puesto que al usar la herramienta de Kubernetes y basándonos en los servicios disponibles de AWS como EKS y en sistemas Linux con Docker y Kubernetes como tal, permite una administración y gestión completa de ambos entornos, entonces en base a esta integración es posible la migración de contenedores de un Cloud a otro. Sin embargo, la literatura existente hasta el momento no es funcional en una arquitectura híbrida para lograr una integración de Kubernetes en OpenStack, ya que se ha realizado una búsqueda en todos los manuales, sugerencias y foros que se encuentran en las diferentes páginas en la web, pero no se encontró una solución para este problema, por lo que no se pudo realizar este tipo de integración.
- Se logró implementar una arquitectura de administración centralizada, el cual ofrece servicios auto-gestionables para los usuarios de la nube híbrida. Esta arquitectura nos permitió gestionar los recursos de ambos entornos, implementar equipos virtuales como servidores, equipos de red y equipos de almacenamiento y sobre todo lograr una comunicación eficiente
- El análisis amplio de la plataforma de Kubernetes, fue de vital importancia ya que con sus principios básicos de compatibilidad en un ambiente híbrido se logró definir a Docker Hub como un método para un proceso de migración ya que, al ser un repositorio de contenedores, este nos la accesibilidad a nuestros proyectos independientemente de la infraestructura.
- De igual manera, se logró reducir drásticamente el tiempo promedio de migración de una aplicación, dado que la tecnología Docker nos permite unificar tanto la virtualización de

cómputo, como el sistema, almacenamiento y redes en un mismo contenedor. Es decir, se redujo el tiempo de implementación de un sistema en un 90%, debido a la portabilidad y fácil despliegue.

- En base al análisis de resultados se determinó que una migración correcta, es posible solo si este se realiza en las mismas versiones de complementos disponibles en los repositorios de Docker Hub, esto debido a las restricciones de compatibilidad entre sistemas desplegados en el contenedor.
- Como consecuencia, acceder a los servicios de una infraestructura de Cloud en AWS para entornos de desarrollo es relativamente factible ya que al ser propietaria este genera altos costos, por lo que hemos determinado que una infraestructura de Cloud híbrida es la mejor opción de implementación para tener servicios seguros y protegidos y a su vez crear ambientes de pruebas en un entorno privado y entornos de producción en un Cloud público.

RECOMENDACIONES

- Para gestionar y trabajar con EKS de AWS de manera más sencilla se recomienda utilizar el servicio de CloudFormation, dado que emplea plantillas que crean un modelo de red con las configuraciones necesarias que requiere el clúster de Kubernetes, ya que realizar estas configuraciones manualmente se vuelve una tarea muy compleja.
- Para obtener una comunicación fluida entre la infraestructura de nube privada de OpenStack y la infraestructura de nube pública de AWS, se recomienda que en OpenStack se asigne recursos mayores a los empleados en este proyecto y a partir de esto, la instalación y configuración de los elementos que se ejecutarán dentro de esta infraestructura se realicen de una manera rápida y eficiente.
- Se recomienda tomar en cuenta los puertos necesarios que requiere nuestra infraestructura de modo que al configurar los grupos de seguridad en OpenStack, se pueda tener acceso a todos los servicios desplegados como SSH, HTTPS, etc.

REFERENCIAS

[1] T. Grance y P. Mell, The NIST Definition of Cloud Computing, N. I. o. S. a. Technology, Ed., Gaithersburg, 2011, p. 7.

[2] 192-168-i-i.com, «192.168.1.1 Admin Login,» 13 11 2020. [En línea]. Available: https://192-168-i-i.com/.

[3] php, «PHP: Historia de PHP y Proyectos Relacionados - Manual,» 4 6 2020. [En línea]. Available: https://www.php.net/manual/es/history.php.

[4] SoftwareLab, «¿Qué es un firewall en informatica? La definición y los tipos principales,» 12 8 2020. [En línea]. Available: https://softwarelab.org/es/que-es-un-firewall/.

[5] IONOS , «¿Qué es Kubernetes?,» 9 12 2020. [En línea]. Available: https://www.ionos.es/digitalguide/servidores/know-how/que-es-kubernetes/.

[6] L. Joyanes Aguilar, Estrategias de Cloud Computing en las empresas, R. d. I. E. d. E. Estratégicos, Ed., España, 2013, p. 24.

[7] Evaluando Cloud, «ABC del Cloud Computing,» 2 7 2018. [En línea]. Available: https://evaluandocloud.com/abc-del-cloud-computing/#:%7E:text=Autoservicio%20bajo%20demanda,o%20proveedores%20de%20servicios%20Cloud.

[8] A. Rashid y A. Chaturvedi,.

[9] L. Sastre, «Computación en la nube. Desafío y oportunidad en la sociedad conectada,» 11 2014. [En línea]. Available: https://cet.la/estudios/cet-la/computacion-en-la-nube-desafio-y-oportunidad-en-la-sociedad-conectada/.

[10] Sage, «Nube,» 7 11 2019. [En línea]. Available: https://www.sage.com/es-es/blog/diccionario-empresarial/nube/.

[11] L. J. Aguilar, Computación en la Nube: estrategias de Cloud Computing en las empresas, Alfaomega grupo editor, 2012.

[12] C. F. Pérez V, J. E. Cleves P y L. Pallares, Computación en la nube: Un nuevo paradigma en las tecnologías de la información y la comunicación. Redes De Ingeniería, 2017, pp. 138-146..

[13] C. Beltrán, A. Miranda, D. Martínez, C. Villacis y F. Balarezo, Servicios Cloud Computing, O. d. l. E. Latinoamericana, Ed., 2018.

[14] Debashis De, Mobile Cloud Computing: Architectures, Algorithms and Applications, Kolkata: Chapman and Hall/CRC, 2016, p. 38.

[15] . V. Josyula, M. Orr y G. Page, Cloud Computing: Automating the virtualized data center, Cisco Press, 2011.

[16] E. S. Sacoto Cabrera, Análisis basado en teoría de juegos de modelos de negocio de operadores móviles virtuales en redes 4G y 5G, Valencia: Universitat Politècnica de València, 2021.

[17] E. J. Sacoto Cabrera , L. Guijarro, J. Vidal y V. Pla, «Economic feasibility of virtual operators in 5G via network slicing,» *Future Generation Computer Systems,* vol. 109, pp. 172-187., 2020.

[18] E. J. Sacoto Cabrera, L. Guijarro y P. Maille, «Game Theoretical Analysis of a Multi-MNO MVNO Business Model in 5G Networks,» *Electronics,* vol. 9, nº 6, p. 933..

[19] E. J. Sacoto Cabrera , A. Sanchis Cano, L. Guijarro , J. Vidal y V. Pla, «Strategic interaction between operators in the context of spectrum sharing for 5g networks.,» *Wireless Communications and Mobile Computing,* 2018.

[20] A. Sanchis Cano, J. Romero, E. J. Sacoto Cabrera y L. Guijarro , «Economic feasibility of wireless sensor network-based service provision in a duopoly setting with a monopolist operator,» *Sensors,* vol. 7, nº 12, p. 2727..

[21] V. Vimos y E. J. Sacoto Cabrera, «Results of the implementation of a sensor network based on Arduino devices and multiplatform applications using the standard OPC UA.,» *IEEE Latin America Transactions,,* vol. 9, nº 16, pp. 2496-2502., 2018.

[22] E. J. Sacoto Cabrera, J. Rodriguez-Bustamante, P. Gallegos Segovia, G. Arevalo-Quishpi y G. León-Paredes, «Internet of Things: Informatic system for metering with communications MQTT over GPRS for smart meters.,» de *CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, Pucón , 2017.

[23] V. Vimos, E. J. Sacoto Cabrera y D. Morales , «Conceptual architecture definition: Implementation of a network sensor using Arduino devices and multiplatform applications through OPC UA.,» de *IEEE International Conference on Automatica (ICA-ACCA)* , 2016.

[24] E. J. Sacoto Cabrera, S. Palaguachi, G. Leon Paredes, P. Gallegos Segovia y O. Bravo Quezada, «Industrial Communication Based on MQTT and Modbus Communication Applied in a Meteorological Network.,» de *The International Conference on Advances in Emerging Trends and Technologies* , 2020.

[25] R. Suárez, Eucalyptus, la nube en casa. Todo linux: la revista mensual para entusiastas de GNU/LINUX, 2009, pp. 10-17.

[26] A. González Madroño, P. M, D. López Collado y R. Viudez Corroto, An Open Cloud Computing Interface (OCCI) Management Console for the OpenNebula Toolkit, 2010.

[27] R. Kumar, K. Jain, H. Maharwal, N. Jain y A. Dadhich, Apache cloudstack: Open source infrastructure as a service Cloud Computing platform. Proceedings of the International Journal of advancement in Engineering technology, M. a. A. Science, Ed., 2014, pp. 111,116..

[28] Red Hat, Inc., «OpenStack Documentation,» 22 12 2020. [En línea]. Available: https://www.redhat.com/es/topics/openstack/.

[29] R. Kumar, N. Gupta, S. Charu, K. Jain y S. K. Jangir, Open source solution for Cloud Computing platform using OpenStack, I. J. o. C. S. a. M. Computing, Ed., 2014, pp. 89-98.

[30] R. Gupta, «Introduction to Google Cloud Platform,» 13 7 2018. [En línea]. Available: https://www.whizlabs.com/blog/google-cloud-platform/.

[31] Thiluxan, «Introduction to Microsoft Azure,» 5 5 2020. [En línea]. Available: https://medium.com/@thiluxan/introduction-to-microsoft-azure-84baa1e3efa.

[32] IBM Cloud Architecture Center, «Get started with agile development and continuous delivery,» 19 12 2020. [En línea]. Available: https://www.ibm.com/cloud/architecture/content/course/get_started_agile_cd/practice_introduction_to_bluemix/.

[33] S. Neenan, «An introduction to Alibaba cloud offerings,» 28 9 2020. [En línea]. Available: https://searchcloudcomputing.techtarget.com/feature/An-introduction-to-Alibaba-cloud-offerings.

[34] Oracle, «Welcome to Oracle Cloud Infrastructure,» 17 12 2020. [En línea]. Available: https://docs.cloud.oracle.com/en-us/iaas/Content/GSG/Concepts/baremetalintro.htm.

[35] Amazon Web Services, Inc, «What is AWS?,» 15 12 2020. [En línea]. Available: https://aws.amazon.com/what-is-aws/?nc1=h_ls.

[36] Simplilearn, «What is AWS?,» 20 11 2020. [En línea]. Available: https://www.simplilearn.com/tutorials/aws-tutorial/what-is-aws.

[37] Amazon Web Services, Inc, «AWS | Red virtual privada en la nube (VPC),» 30 12 2020. [En línea]. Available: https://aws.amazon.com/vpc/?nc1=h_ls&vpc-blogs.sort-by=item.additionalFields.createdDate&vpc-blogs.sort-order=desc.

[38] Amazon Web Services, Amazon Web Services -Información general acerca de los procesos de seguridad., 2017.

[39] Amazon Web Services, Inc, «Amazon EKS | Managed Kubernetes Service | Amazon Web Services,» 30 12 2020. [En línea]. Available: https://aws.amazon.com/eks/?nc1=h_ls&whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc&eks-blogs.sort-by=item.additionalFields.createdDate&eks-blogs.sort-order=desc.

[40] Amazon Web Services, Inc, «Introduction to AWS CloudFormation,» 30 12 2020. [En línea]. Available: https://aws.amazon.com/es/cloudformation/.

[41] Amazon Web Services, Inc, «What is Amazon EC2? - Amazon Elastic Compute Cloud,» 30 12 2020. [En línea]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html.

[42] Amazon Web Services, Inc, «What is the AWS Command Line Interface? - AWS Command Line Interface,» 24 12 2020. [En línea]. Available: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html.

[43] Amazon Web Services, Inc, «Global Infrastructure,» 30 12 2020. [En línea]. Available: https://aws.amazon.com/about-aws/global-infrastructure/?nc1=h_ls.

[44] AWS, «Información general sobre Amazon Web Services: Documento técnico de AWS,» Copyright., 2018.

[45] M. Cabrera Gómez de la Torre, «Gestion de contenedores Docker - Kubernetes,» 2015.

[46] B. B. Rad, H. J. Bhatti y M. Ahmadi, An introduction to docker and analysis of its performance, I. J. o. C. S. a. N. S. (IJCSNS), Ed., 2017, p. 228.

[47] F. López, «Kubernetes,» 16 1 2019. [En línea]. Available: https://www.conasa.es/blog/kubernetes/.

[48] Kubernetes, «¿Qué es Kubernetes?,» 17 7 2020. [En línea]. Available: https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/.

[49] Red Hat, Inc., «¿Qué es Kubernetes?,» 19 8 2018. [En línea]. Available: https://www.redhat.com/es/topics/containers/what-is-kubernetes.

[50] T. Eduard , «¿Qué es Kubernetes y cómo funciona?,» 14 3 2019. [En línea]. Available: https://www.campusmvp.es/recursos/post/que-es-kubernetes-y-como-funciona.aspx.

[51] IONOS, «Servidor web: definición, historia y programas,» 9 12 2020. [En línea]. Available: https://www.ionos.es/digitalguide/servidores/know-how/servidor-web-definicion-historia-y-programas/.

[52] I. Souza, «¿Qué es un servidor web y cuáles son sus características?,» 5 8 2020. [En línea]. Available: https://rockcontent.com/es/blog/que-es-un-servidor/.

[53] webempresa, «¿Qué es un servidor web y para qué sirve?,» 28 10 2020. [En línea]. Available: https://www.webempresa.com/hosting/que-es-servidor-web.html.

[54] S. Borges, «Servidor Web,» 7 8 2019. [En línea]. Available: https://blog.infranetworking.com/servidor-web/.

[55] A. Marín Illera , «Apache, PHP y MySQL,» 3 3 2017. [En línea]. Available: http://e-ghost.deusto.es/docs/articulo.apm.html.

[56] L. A. Santillán C, M. G. Ginestà y Ó. O. Mora, Bases de datos en MySQL, U. O. d. Catalunya, Ed., 2014.

ANEXOS

ANEXO 1: INSTALACIÓN DE OPENSTACK

En el siguiente apartado vamos a detallar paso a paso la correcta instalación de OpenStack

CARACTERÍSTICAS

- CentOS 7 x86.
- 14 Gb de memoria RAM.
- 6 procesadores.

1. Antes de empezar con la instalación de la plataforma OpenStack, debemos correr los siguientes comandos.

   yum -y update

   yum -y upgrade

   Esto nos permitirá tener todas las dependencias y repositorios actualizados
2. Asignamos una IP fija a nuestra máquina para eso podemos utilizar el comando nmtui.

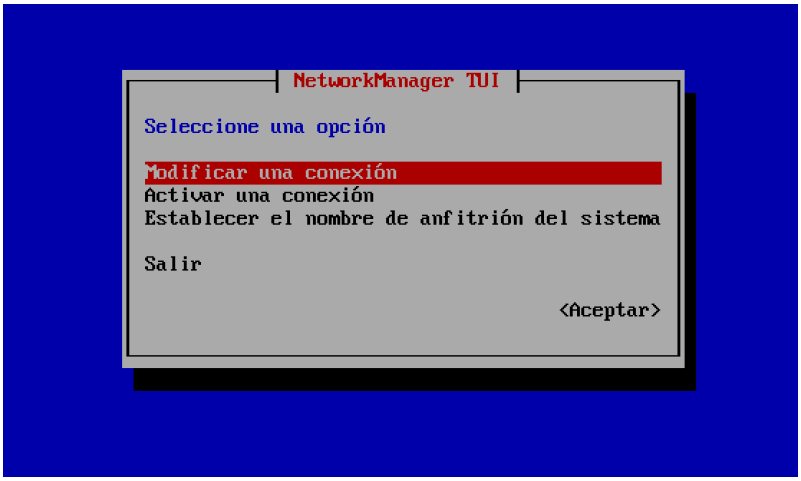


Figura 33. Configuración de IP estática. Fuente Autor

También podemos entrar en el archivo ifcfg-ens33 de la siguiente manera vi /etc/sysconfig/network-scripts/ifcfg-ens33, cabe recalcar que nuestra interfaz es ens33

3. Agregamos nuestra IP y host en vi /etc/hosts para que nos resuelva por el nombre y no como localhost
4. Desactivamos y paramos el firewall y el administrador de red caso contrario nos generará errores al momento de instalar OpenStack

```
systemctl stop firewalld
systemctl disable firewalld
systemctl stop NetworkManager
systemctl disable NetworkManager
```

5. Deshabilitar selinux, es decir cambiamos de enforcing a disabled y mandamos a reiniciar.

```
vi /etc/selinux/config
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Figura 34. Deshabilitar SELINUX. Fuente Autor

6. Descargamos los repositorios de OpenStack packstack para esta plataforma instalaremos la versión stein ya que la ocata no se encuentra disponible, así mismo después de terminar la instalación actualizamos los repositorios de nuestro sistema y reiniciamos

```
yum install -y centos-release-openstack –stein
yum update -y
reboot
```

7. En este momento continuaremos instalando los componentes y servicios de OpenStack -packstack, al terminar reiniciamos la máquina

```
yum install -y openstack –packstack
yum update -y
```

8. Configuramos los componentes de OpenStack -packstack, en esta versión instalaremos todos los componentes en un solo nodo, cabe recalcar que no instalaremos el modo demo de OpenStack, a esta instalación le agregaremos los componentes heat y magnun, que son

los componentes claves para el despliegue de contenedores y cluster de Kubernetes en OpenStack, este apartado puede durar entre unos 30 minutos.

```
packstack --allinone --os-neutron-l2-agent=openvswitch --os-neutron-ml2-mechanism-drivers=openvswitch --os-neutron-ml2-tenant-network-types=vxlan --os-neutron-ml2-type-drivers=vxlan,flat,vlan --provision-demo=n --os-neutron-ovs-bridge-mappings=extnet:br-ex --os-neutron-ovs-bridge-interfaces=br-ex:ens33 --os-heat-install=y --os-magnum-install=y
```

Figura 35. Fin de instalación OpenStack. Fuente Autor

9. Al finalizar y terminar con la instalación, podemos ingresar a la interfaz de OpenStack, para esto necesitaremos un usuario y contraseña, entonces accederemos al archivo.

```
vi keystonerc_admin
```

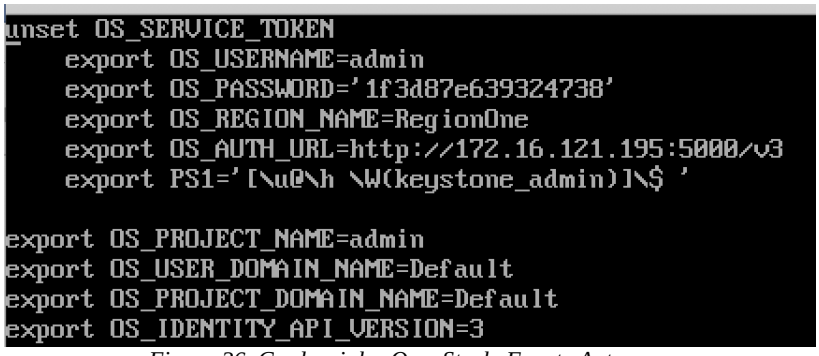


Figura 36. Credenciales OpenStack. Fuente Autor

10. Para terminar, tenemos dos formas de acceder a la administración de OpenStack en un navegador con nuestra IP ingresamos a OpenStack y desde la cli con el comando.

```
source keystonerc_admin
```

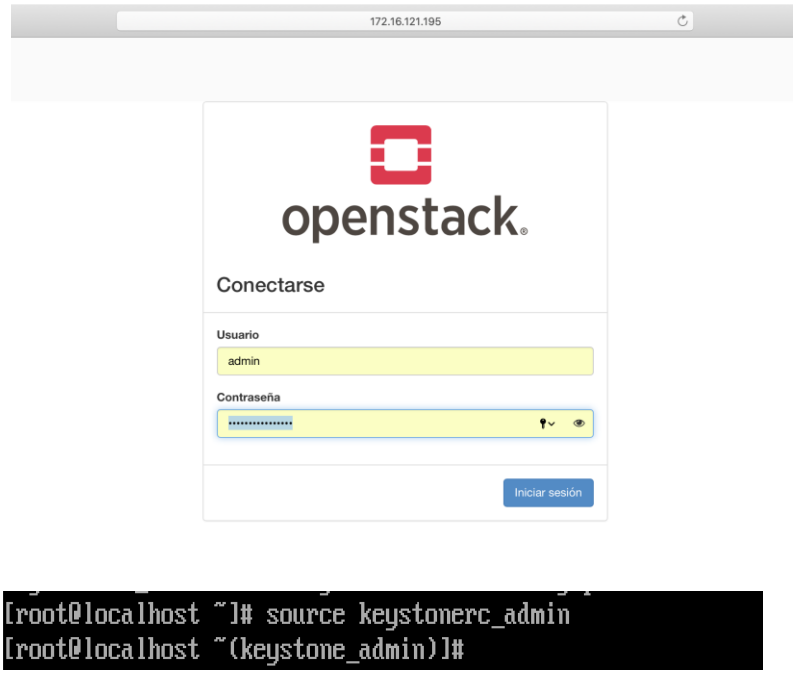


Figura 37. Acceso por dashboard y CLI. Fuente Autor

DESPLIEGUE DE INSTANCIAS EN OPENSTACK

11. Para desplegar instancias primero es necesario tener creado una red externa, un router y una red interna.

    Para crear la red externa lo tenemos que hacer mediante línea de comandos ya que por interfaz gráfica no se puede realizar una correcta comunicación con el exterior

    Entonces ingresamos a la cli de OpenStack

    ```
    source keystonerc_admin
    ```

    Creamos una red externa tipo flat

    ```
    neutron net-create kubenet --provider:network_type flat --provider:physical_network extnet  --router:external
    ```

    Segundo creamos una subred, la misma que debemos asignarla a la red externa anteriormente creada, en este comando debemos verificar si las ips tanto de red como gateway coinciden con las de nuestra red local, caso contrario esta no tendrá comunicación.

```
neutron subnet-create --name public_subnet --enable_dhcp=False --allocation-pool=start=172.16.121.50,end=172.16.121.200 --gateway=172.16.121.2 kubenet 172.16.121.0/24
```

12. Creamos un router, esto se puede realizar desde la dashboard o Cli.

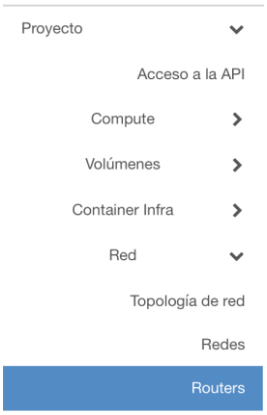


Figura 38. Creación de Router. Fuente Autor

13. Asignamos un nombre y lo anclamos a la red pública creada anteriormente, y pulsamos crear.

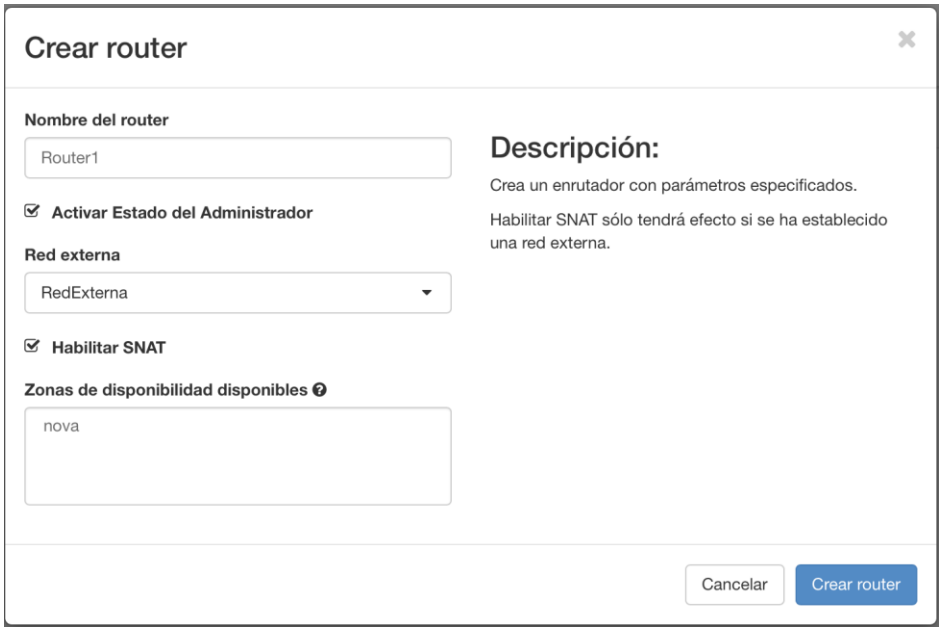


Figura 39. Detalles de Creación. Fuente Autor

14. Creamos una red interna para esto necesitaremos una red y un rango de IPs que deseamos asignar.

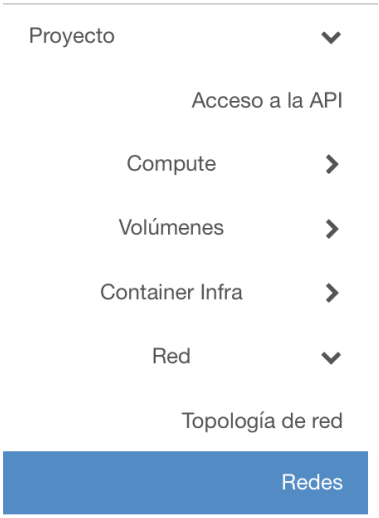


Figura 40. Creación de red interna. Fuente Autor

15. Asignamos un nombre a la red y seleccionamos siguiente

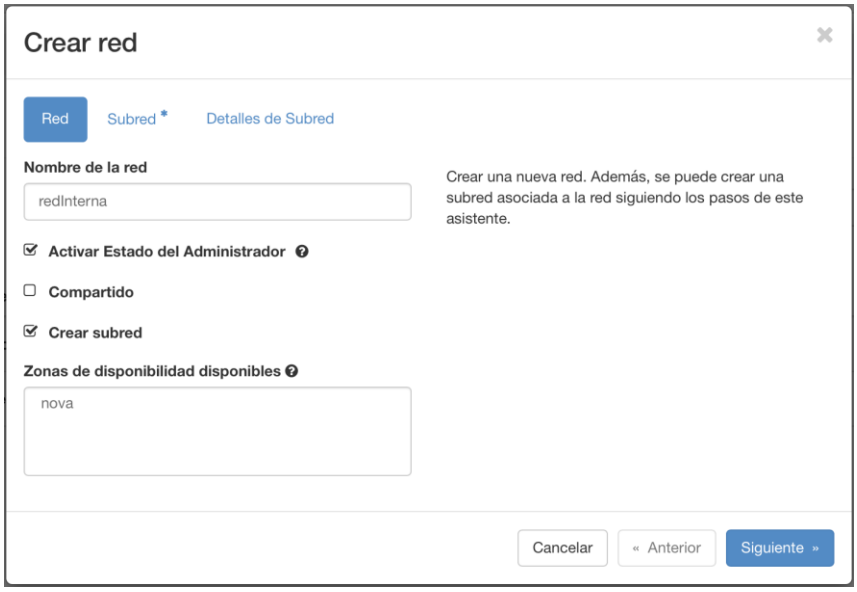


Figura 41. Detalles crear red interna. Fuente Autor

16. A continuación, seleccionamos siguiente y asignamos nuestra red de preferencia y la puerta de enlace.

Figura 42. Detalles subred interna. Fuente Autor

17. En este apartado asignamos el pool de direcciones DNS para tener conexión a internet seleccionamos crear y listo.

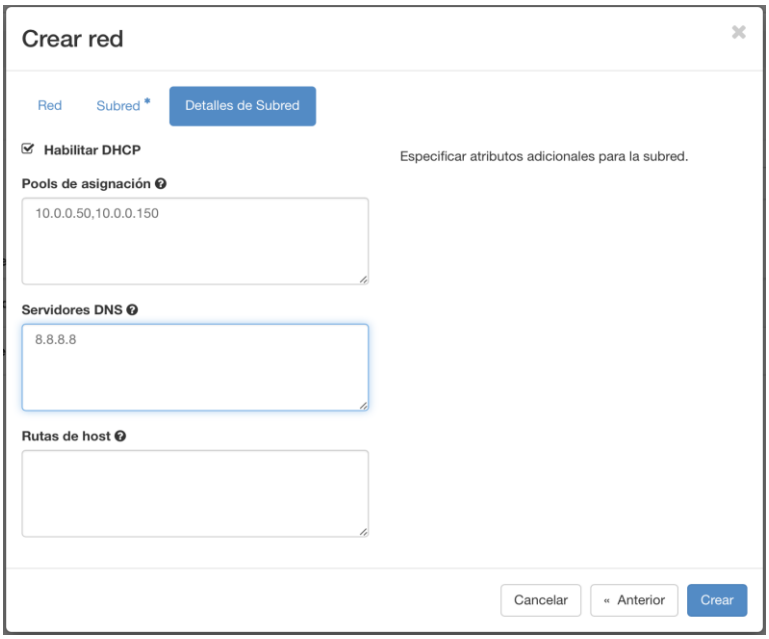


Figura 43. Detalles de direcciones. Fuente Autor

18. Ahora nos dirigimos a la topología y anclamos nuestra red interna al router pulsamos en añadir interfaz, seleccionamos el nombre de la red interna y en IP le dejamos en blanco para que se asigne una de forma automática.

Figura 44. Agregar interfaz. Fuente Autor

19. Ahora debemos crear un grupo de seguridad el cual nos permitirá la conectividad con el exterior

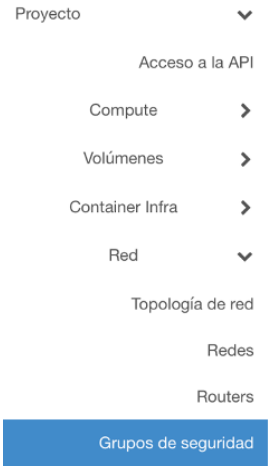


Figura 45. Seleccionar grupos de seguridad. Fuente Autor

20. Seleccionamos crear, le asignamos un nombre, clic en crear grupo de seguridad, es decir agregamos IPTABLES.

Figura 46. Detalles de grupos de seguridad. Fuente Autor

21. A continuación, seleccionamos administrar reglas y agregamos todas las reglas necesarias como, por ejemplo: puerto SSH, https, MySQL, es decir todos los puertos que deseamos

tener abiertos para que nuestros servicios se ejecuten correctamente, en mi caso tengo los siguientes.

Administrar Reglas de Grupo de Seguridad: kubesg (31ba5cd6-2578-49ef-8077-949647e52f7d)

Mostrando 32 articulos

Dirección	Tipo Ethernet	Protocolo IP	Rango de puertos	Prefijo de IP Remota
Saliente	IPv4	Cualquier	Cualquier	0.0.0.0/0
Saliente	IPv4	TCP	179	0.0.0.0/0
Saliente	IPv4	TCP	2379 - 2380	0.0.0.0/0
Saliente	IPv4	TCP	6443	0.0.0.0/0
Saliente	IPv4	TCP	6781 - 6784	0.0.0.0/0
Saliente	IPv4	TCP	6783 - 6784	0.0.0.0/0
Saliente	IPv4	TCP	9099	0.0.0.0/0
Saliente	IPv4	TCP	10250	0.0.0.0/0
Saliente	IPv4	TCP	10251	0.0.0.0/0
Saliente	IPv4	TCP	10252	0.0.0.0/0
Saliente	IPv4	TCP	10255	0.0.0.0/0
Saliente	IPv4	TCP	30000 - 32767	0.0.0.0/0
Saliente	IPv4	UDP	8285	0.0.0.0/0
Saliente	IPv4	UDP	8472	0.0.0.0/0
Saliente	IPv6	Cualquier	Cualquier	::/0
Entrante	IPv4	ICMP	Cualquier	0.0.0.0/0
Entrante	IPv4	TCP	22 (SSH)	0.0.0.0/0
Entrante	IPv4	TCP	80 (HTTP)	0.0.0.0/0
Entrante	IPv4	TCP	179	0.0.0.0/0
Entrante	IPv4	TCP	443 (HTTPS)	0.0.0.0/0
Entrante	IPv4	TCP	2379 - 2380	0.0.0.0/0
Entrante	IPv4	TCP	6443	0.0.0.0/0
Entrante	IPv4	TCP	6781 - 6784	0.0.0.0/0
Entrante	IPv4	TCP	6783 - 6784	0.0.0.0/0
Entrante	IPv4	TCP	9099	0.0.0.0/0
Entrante	IPv4	TCP	10250	0.0.0.0/0
Entrante	IPv4	TCP	10251	0.0.0.0/0
Entrante	IPv4	TCP	10252	0.0.0.0/0

Figura 47. Puertos abiertos de seguridad. Fuente Autor

22. A continuación, creamos las IPs flotantes y asignamos a la red externa.

Figura 48. Asignación de IP flotantes. Fuente Autor

23. Continuamos por crear un volumen y un sabor, en este apartado existen sabores ya existentes que nos da la plataforma, pero si ninguno se adecua a los recursos que necesitamos podemos crearlo.

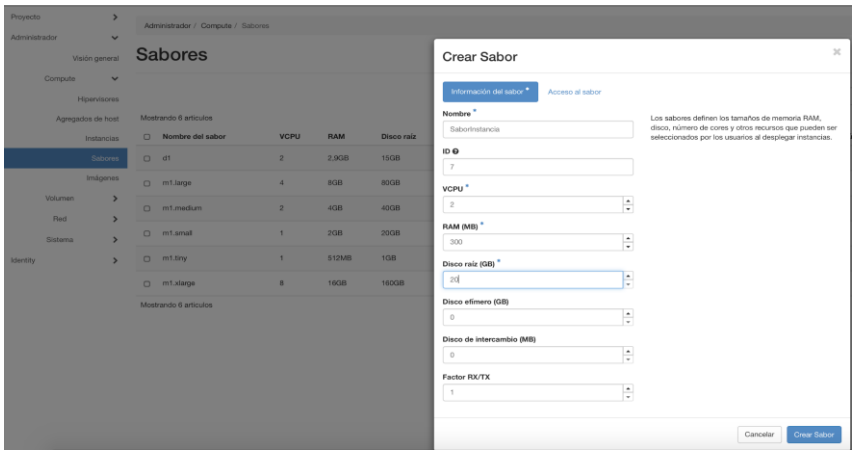


Figura 49. Creación de sabores. Fuente Autor

24. En los que se refiera el volumen es el que guardará todos los datos que se generen en la instancia, por lo que es muy importante la creación de dicho elemento.

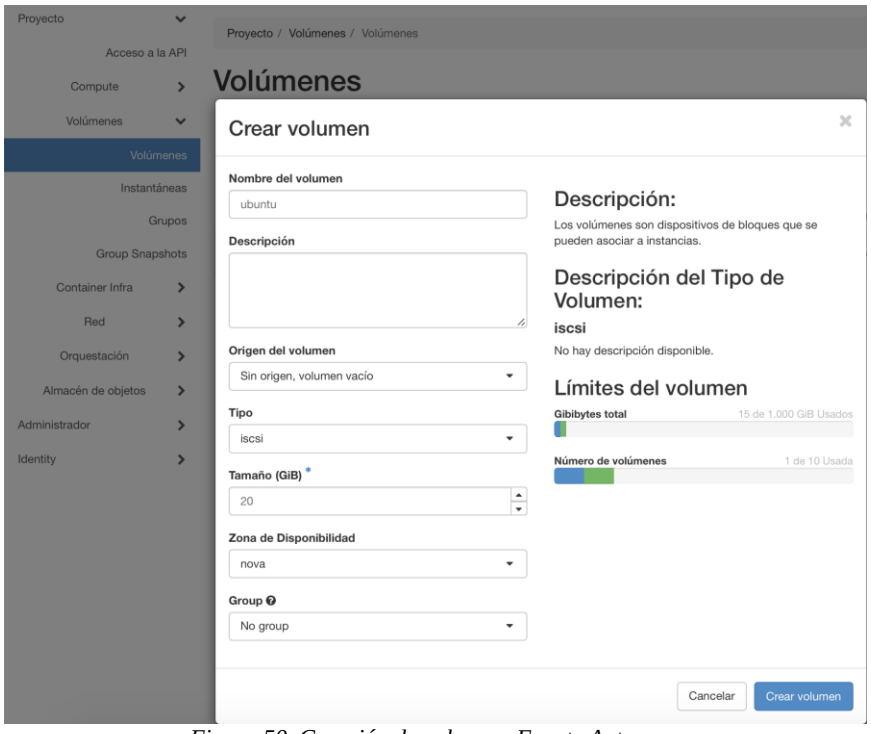


Figura 50. Creación de volumen. Fuente Autor

25. Ahora tenemos listo para comenzar a descargar nuestra imagen o seleccionarla y cargarla, para poder desplegar una instancia, estas imágenes a seleccionar tienen que ser tipo qcow2 propias para entornos de cloud en OpenStack, estas imágenes se pueden encontrar en el siguiente link https://docs.openstack.org/image-guide/obtain-images.html, también se puede generar una imagen a partir de un ISO, pero se tiene que seguir un proceso completamente diferente que mostramos a continuación, ya que se tiene que convertir la imagen tipo ISO a qcow2 y de ahí desplegarla en OpenStack.

    Entonces completamos los campos y asignamos un nombre, seleccionamos la imagen descargada y en el apartado disco y memoria RAM lo recomendable es dejarle en 0, después seleccionamos crear imagen esperamos unos minutos y estará lista para poder iniciar instancias.

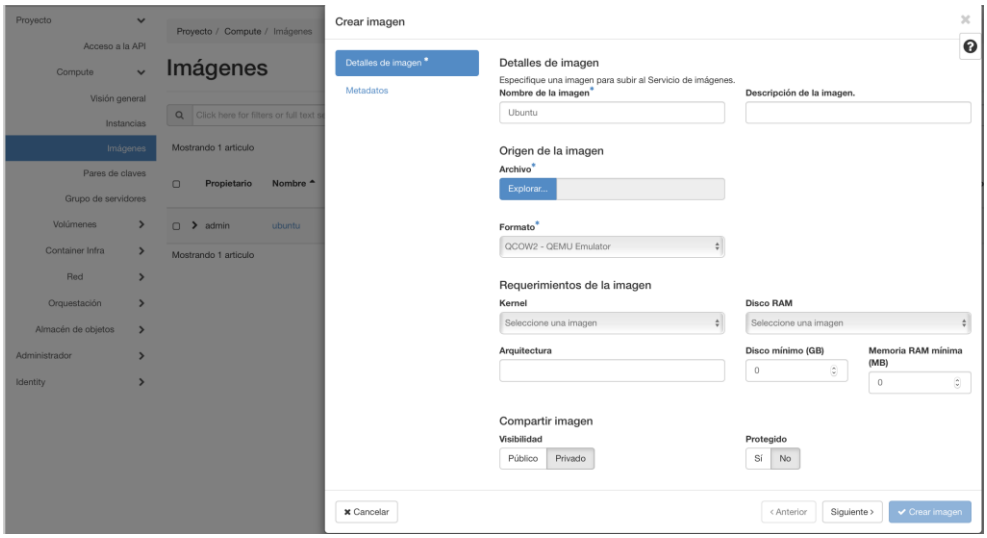


Figura 51. Creación de una imagen. Fuente Autor

26. A continuación, iniciaremos por fin con nuestra instancia para esto asignamos un nombre y seleccionamos siguiente.

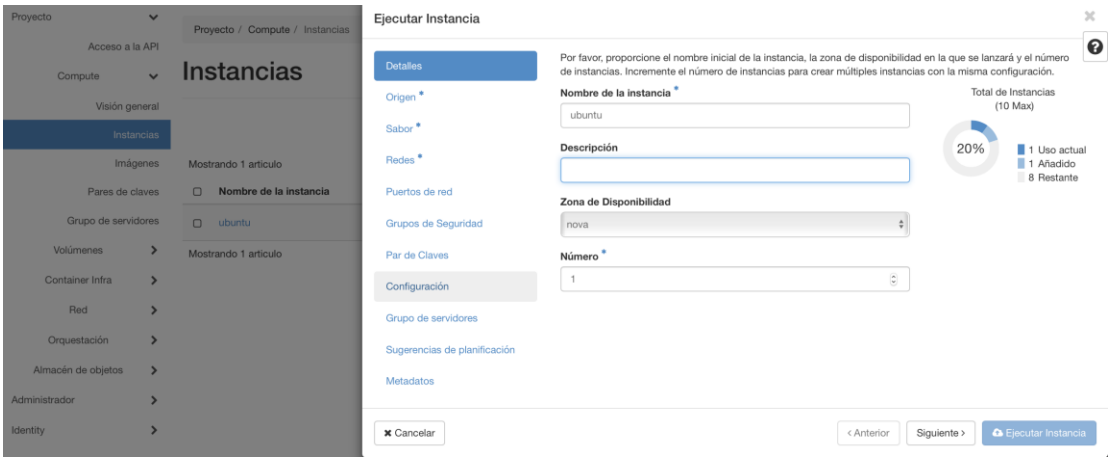


Figura 52. Creación de una instancia. Fuente Autor

27. De la misma manera seleccionamos el origen de arranque como una imagen, en crear volumen seleccionamos no ya que anteriormente ya creamos uno, seleccionamos la imagen descargada anteriormente y seleccionamos siguiente.

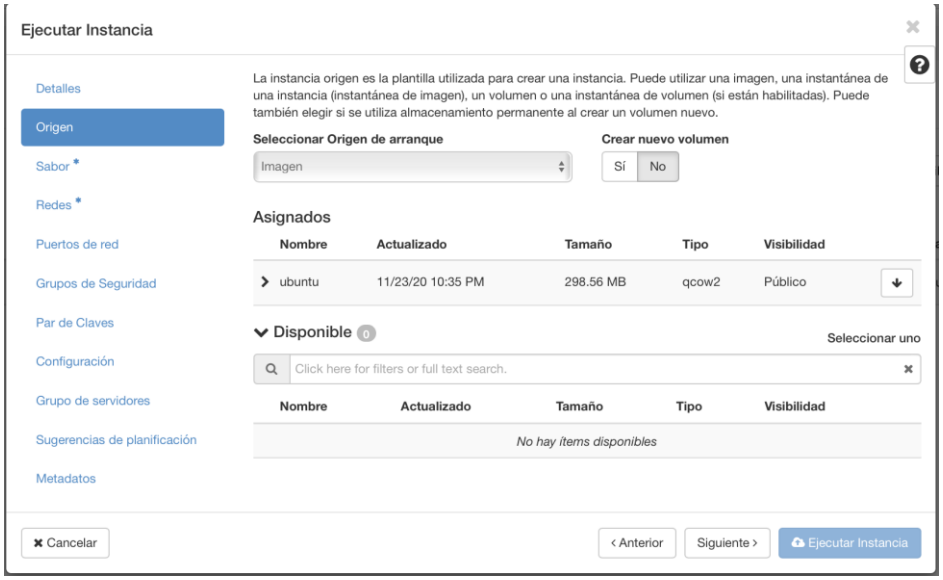


Figura 53. Selección de origen. Fuente Autor

28. Ahora seleccionamos el sabor creado en la sección anterior y seleccionamos siguiente.

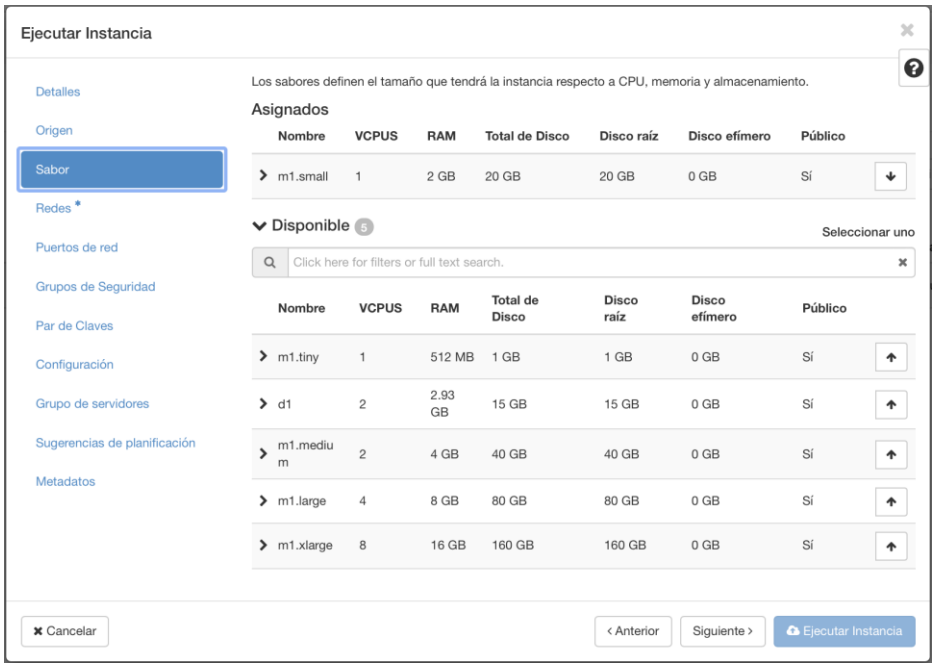


Figura 54. Selección de sabor. Fuente Autor

29. A continuación, seleccionamos la red interna que creamos y pulsamos siguiente.

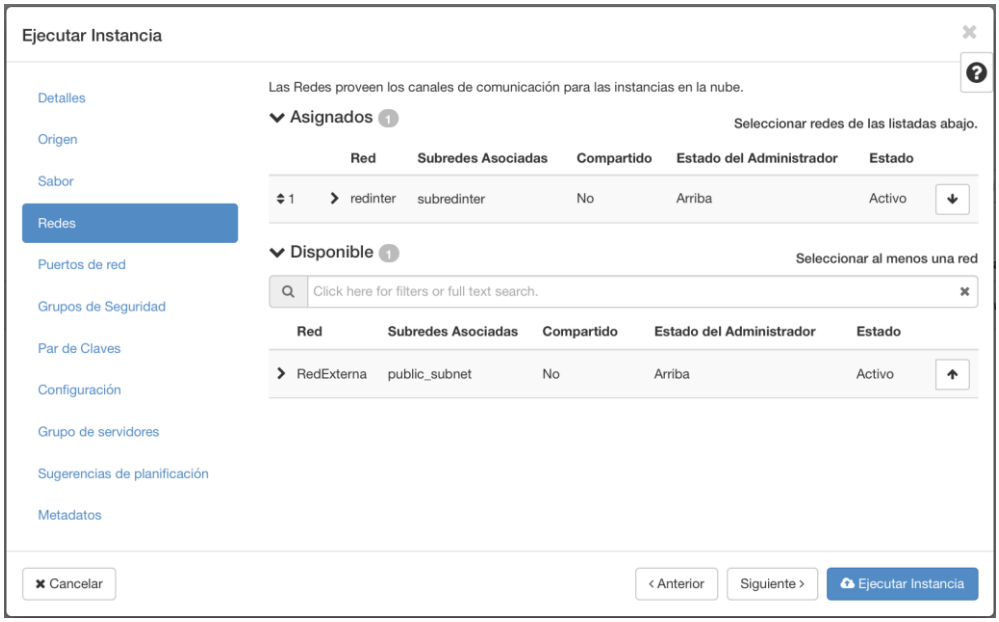


Figura 55. Selección de la red. Fuente Autor

30. En la sección de puertos de red no realizamos nada y continuamos con los grupos de seguridad, el cual seleccionaremos kubesg, el mismo que creamos anteriormente.

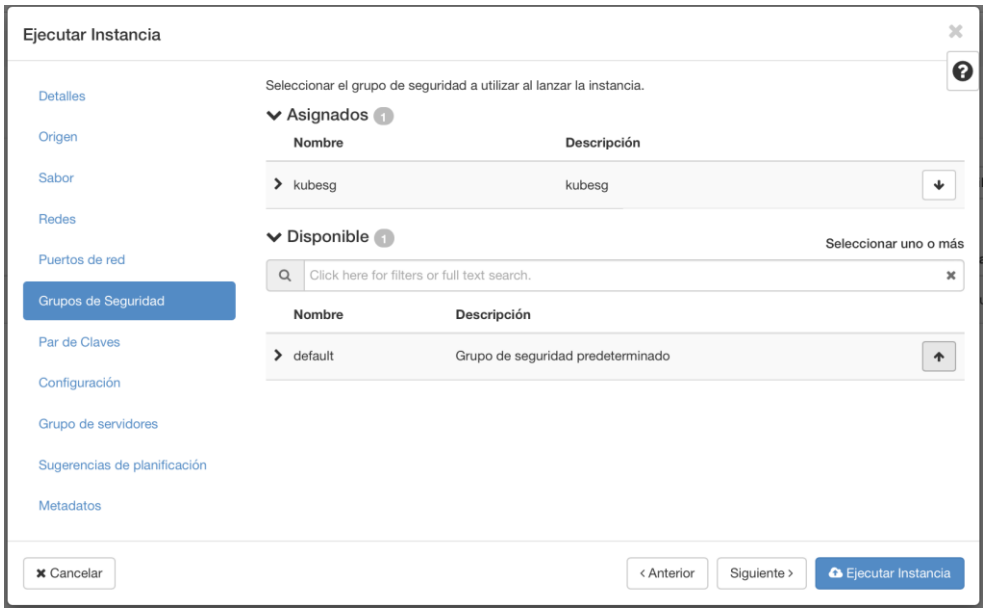


Figura 56. Selección de grupo de seguridad. Fuente Autor

31. En la sección de claves no realizamos ninguna acción, seleccionamos siguiente y en configuración agregamos las siguientes líneas, estas son muy importantes ya que son

credenciales de usuario para entrar en nuestra imagen Ubuntu y para terminar seleccionamos ejecutar instancia.

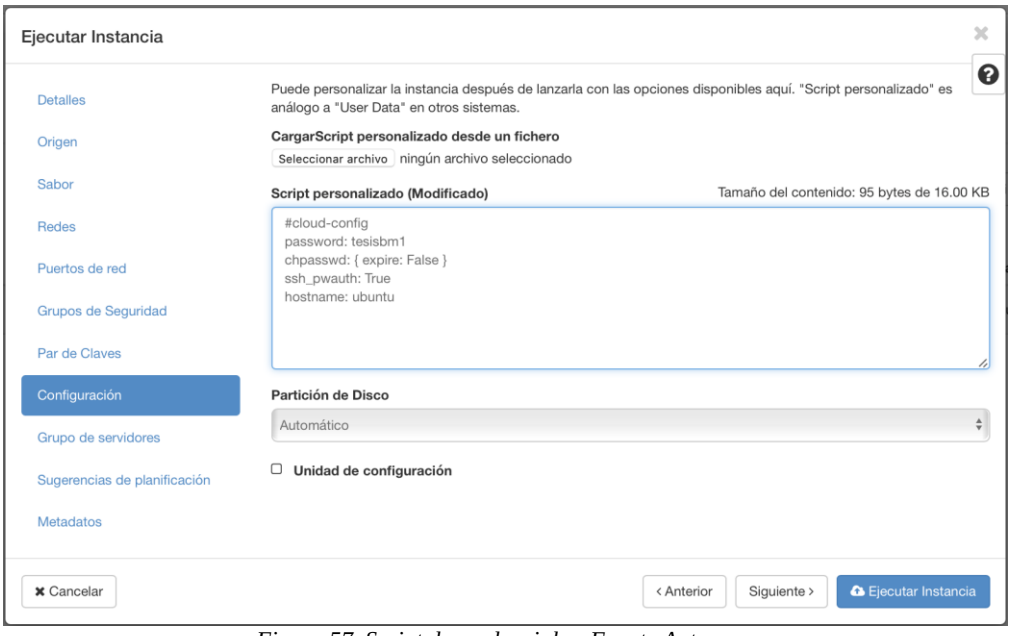


Figura 57. Script de credenciales. Fuente Autor

32. Ahora mientras nuestra instancia se genera lo que hacemos es asignar nuestro volumen y la IP flotante.

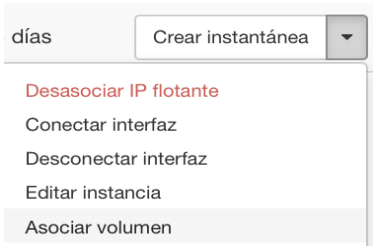


Figura 58. Asociar volumen. Fuente Autor

De la misma forma seleccionamos y asociamos nuestra IP flotante y esperamos a que se termine de crear.

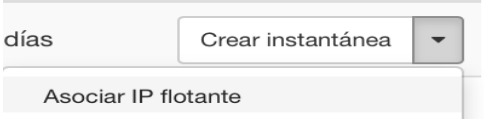


Figura 59. Asociar IP flotante. Fuente Autor

33. A continuación, damos clic en la instancia y nos dirigimos al apartado consola donde la instancia estará lista y podremos loguearnos con las credenciales dadas anteriormente.

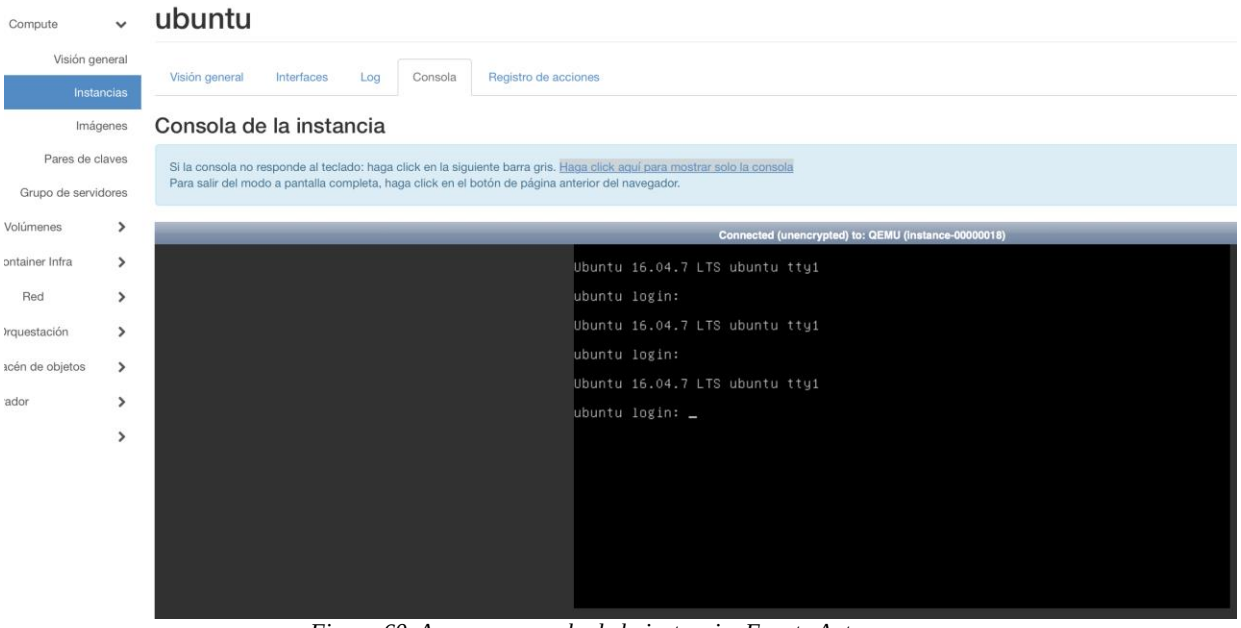


Figura 60. Acceso a consola de la instancia. Fuente Autor

ANEXO 2: CONFIGURACIÓN EN AMAZON WEB SERVICES (AWS)

Para configurar AWS seguimos el siguiente proceso:

1. En la pestaña Servicios buscamos el servicio de CloudFormation.

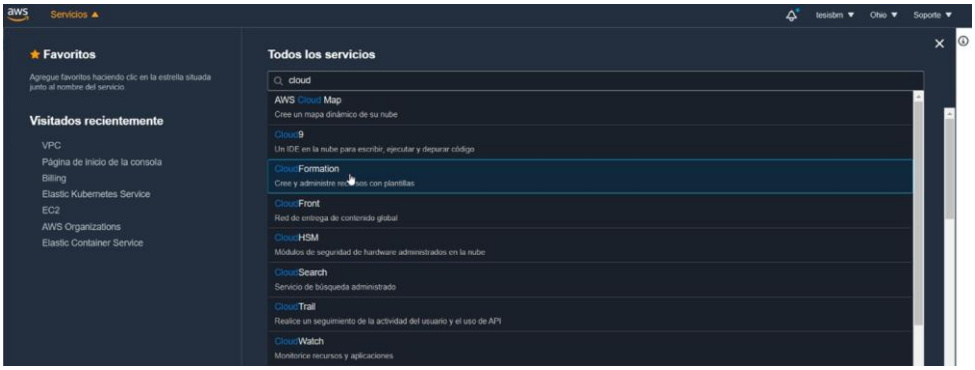


Figura 61. Servicio de CloudFormation. Fuente Autor

2. Presionamos sobre el botón Create stack y seguimos los siguientes pasos para crearlo.

Figura 62. Crear un CloudFormation stack. Fuente Autor

Paso 1. Especificar plantilla: vamos a usar la plantilla que ya está creada, luego procedemos a agregar la plantilla para crear toda la configuración de la red con subredes públicas y privadas, que además estarán replicadas, esta información se encuentra en la documentación de Amazon.

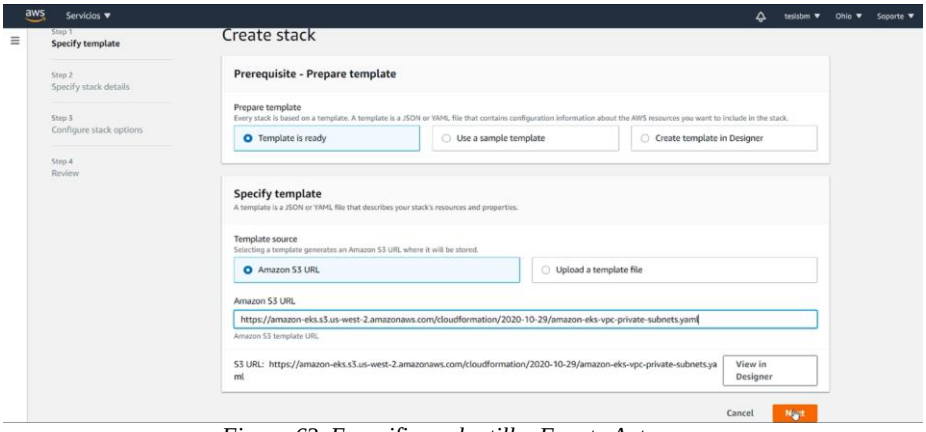


Figura 63. Especificar plantilla. Fuente Autor

Presionamos sobre el botón Next, si todo está correcto nos va a permitir crear el stack.

Paso 2. Especificar detalles del stack: debemos proporcionar un nombre para el stack, aquí se va a crear una serie de subredes necesarias para los recursos del clúster de Kubernetes.

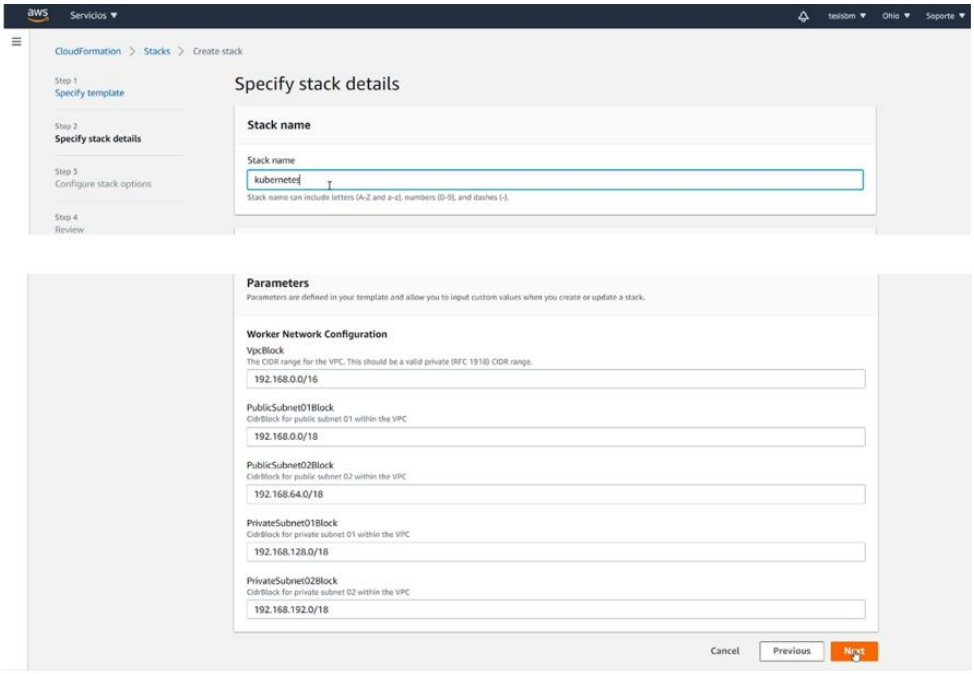


Figura 64. Especificar detalles del stack. Fuente Autor

Damos en Next.

Paso 3. Configurar opciones del stack: lo podemos dejar vacío.

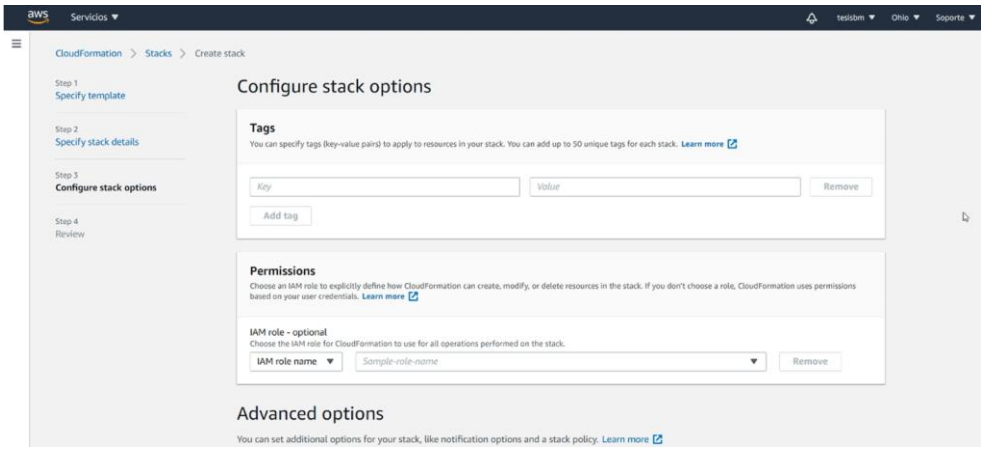


Figura 65. Configurar opciones del stack. Fuente Autor

Damos clic en el botón Next.

Paso 4. Revisión: se hace una revisión de la plantilla que se va a crear y finalmente presionamos sobre el botón Create stack.

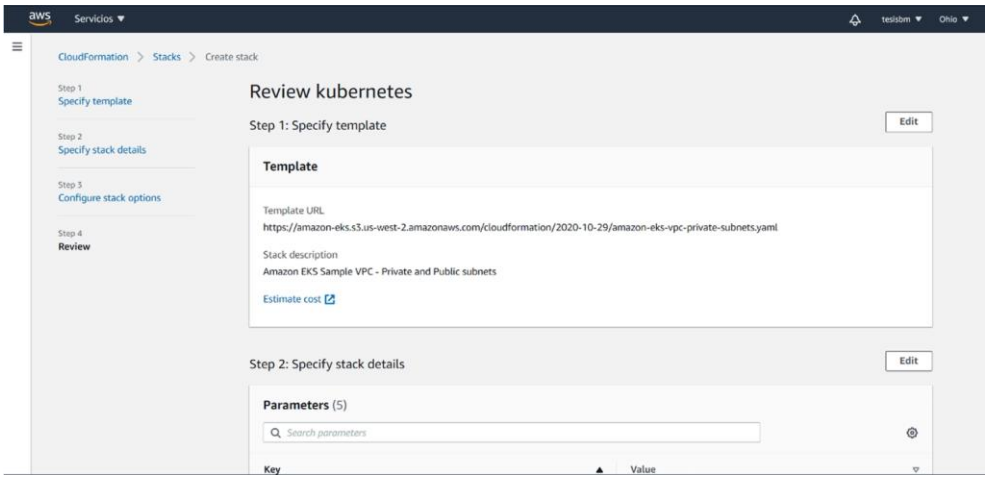


Figura 66. Revisión antes de crear el stack. Fuente Autor

3. En servicios buscamos VPC y verificamos que se haya creado la VPC y el conjunto de subredes del clúster de Kubernetes.

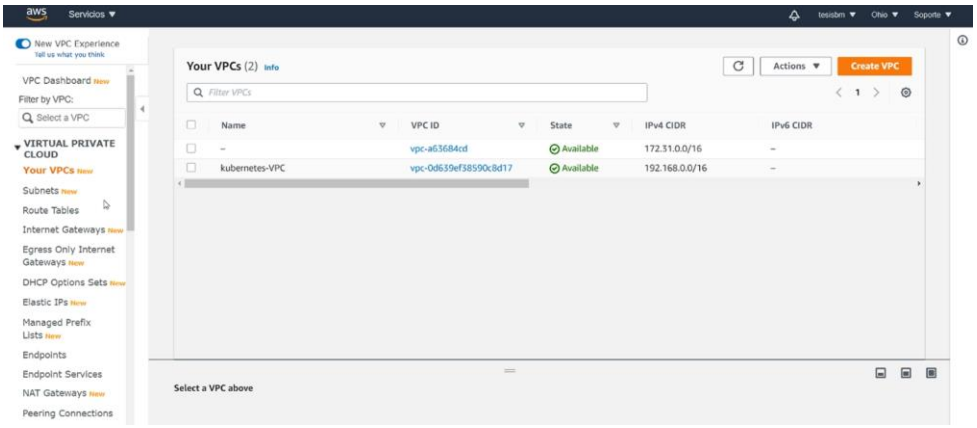


Figura 67. VPC creada. Fuente Autor

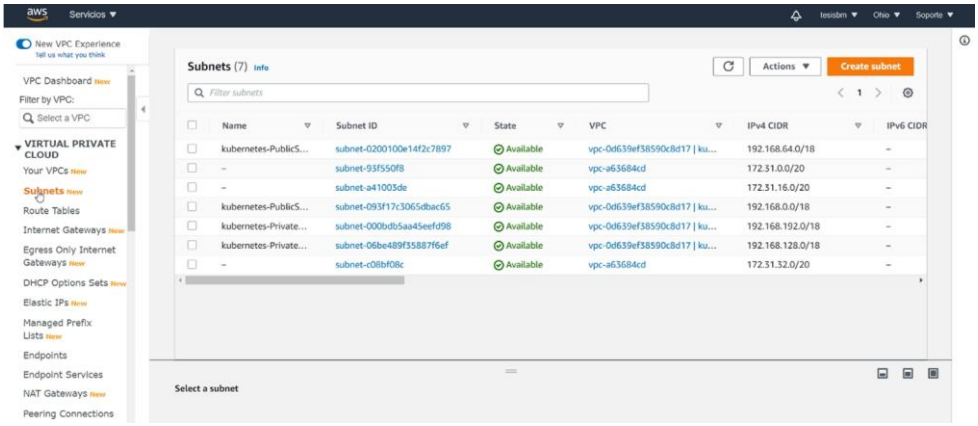


Figura 68. Subredes creadas. Fuente Autor

4. Verificamos en CloudFormation que el estado de la creación del stack sea CREATE_COMPLETE, para posteriormente evitar fallos.

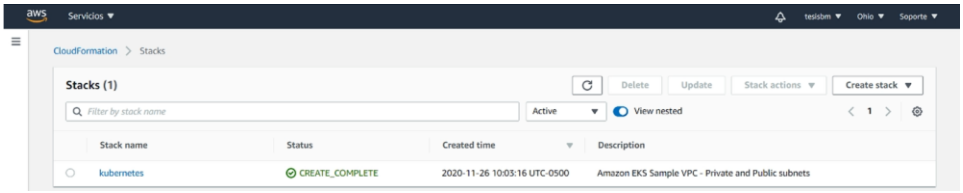


Figura 69. Estado de la creación del stack. Fuente Autor

ANEXO 3: INSTALACIÓN DE DOCKER

En la siguiente sección vamos a detallar la forma correcta de desplegar Docker.

CARACTERÍSTICAS

- 1 maquina con Ubuntu 18.04 con al menos 2,5 G de RAM y 20 HDD de espacio en disco

PREPARACIÓN INICIAL DEL SERVIDOR

1. Antes de empezar con la instalación del servidor Docker, debemos correr los siguientes comandos de actualización del sistema.

   ```
   apt-get update
   apt-get upgrade
   ```

   Asignamos una IP fija a nuestra máquina para eso podemos utilizar el comando nmtui.
2. Agregamos nuestra IP y host en vi /etc/hosts para que nos resuelva por el nombre y no como localhost.
3. Desactivamos y paramos el firewall

   ```
   sudo ufw disable
   ```
4. Deshabilitamos el swap. Podemos deshabilitar con el comando

   ```
   swapoff –a
   ```

   Sin embargo, debemos desactivar del sistema para que no se active al reiniciar el servidor
   Para ello modificamos el fichero /etc/fstab
   Debemos comentar la línea donde aparece el swap de la siguiente manera
   Archivo fstab

```
#  /etc/fstab: static file system information.
#
#  Use 'blkid' to print the universally unique identifier for a
#  device; this may be used with UUID= as a more robust way to name devices
#  that works even if disks are added and removed. See fstab(5).
#
#  <file system> <mount point> <type> <options>
#  / was on /dev/sda1 during installation
<dump> <pass>
UUID=e51dac01-e63f-4cb9-b10d-6b9d7b07a53b / ext4
errors=remount-ro 0 1
# /swapfile                 none       swap sw         0          0
```

Ahora podemos configurar IPTABLES para poder recibir tráfico de tipo bridge, para esto editamos el archivo /etc/ sysctl.conf y agregamos las siguientes líneas al final del archivo.

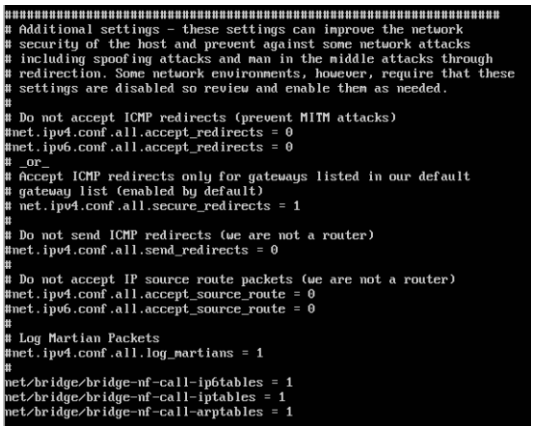


Figura 70. IPTABLES. Fuente Autor

5. A continuación, instalamos algunos paquetes necesarios para Docker.

   ```
   apt-get install ebtables ethtool
   ```

6. Primero debemos actualizar el sistema y después instalamos Docker.

   ```
   apt-get update
   apt-get install -y docker.io
   ```

7. Comprobamos que se haya instalado correctamente y su versión

   ```
   systemctl docker status
   ```

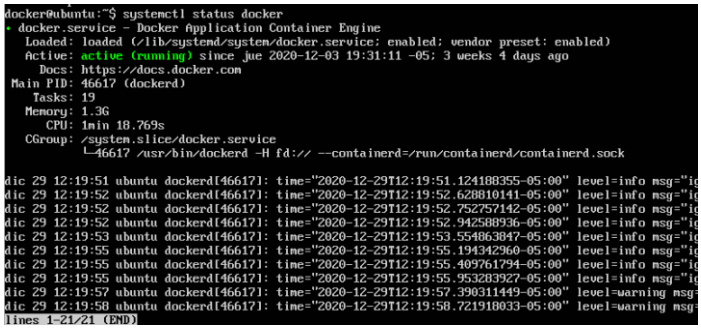


Figura 71. Docker en ejecución. Fuente Autor

```
docker version
```

Figura 72. Estado Docker. Fuente Autor

ANEXO 4: INSTALACIÓN DE KUBERNETES EN OPENSTACK

En la siguiente sección vamos a detallar la forma correcta de desplegar un clúster de Kubernetes

Para proceder con la instalación primero seguimos los pasos que se encuentran en el Anexo 3.

INSTALACIÓN DE KUBECTL, KUBEADM Y KUBELET

1. Instalamos soporte para HTTPS, para esto primero actualizamos el sistema.
   ```
   apt-get update
   apt-get install -y apt-transport-https
   ```
2. Instalamos "curl" si no lo tenemos instalado.
   ```
   apt-get install curl
   ```
3. Recuperamos la clave para el repositorio de Kubernetes y lo añadimos con apt-key:
   ```
   curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg
   | apt-key add -
   ```

Figura 73. Respuesta curl. Fuente Autor

4. Añadimos el repositorio a nuestro system
   ```
   cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
   deb http://apt.kubernetes.io/kubernetes-xenial main
   EOF
   ```
5. Instalamos los 3 componentes principales, pero antes debemos actualizar el sistema: kubeadm, kubelet, y kubectl
   ```
   apt-get update
   apt-get install -y kubelet kubeadm kubectl
   ```

CREACIÓN DEL CLUSTER

6. Creamos el cluster. Por ejemplo, si queremos usar un POD Network Calico, necesitamos añadir el parámetro –pod-network-cidr switch
   ```
   kubeadm init --pod-network-cidr=192.168.0.0/16
   ```

Al finalizar la instalación nos devolverá una serie de comandos para crear una carpeta y archivos que a continuación lo usaremos.

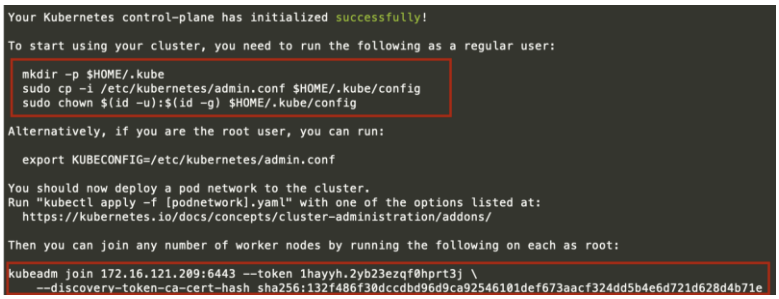


Figura 74. Salida de Kubeadm. Fuente Autor

7. Abrimos una nueva pestaña o un nuevo terminal y nos conectamos y como el usuario con el que vamos a trabajar, en mi caso “tesisbm”, este proceso lo hacemos sin usuario root.

   ```
   mkdir -p $HOME/.kube
   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
   sudo chown $(id -u):$(id -g) $HOME/.kube/config
   ```

8. Instalamos el plugin de calico.

   ```
   kubectl apply -f https://docs.projectcalico.org/v3.11/manifests/calico.yaml
   ```

9. Comprobamos que los pods del Sistema están ejecutándose

   ```
   kubectl get pods --all-namespaces
   ```

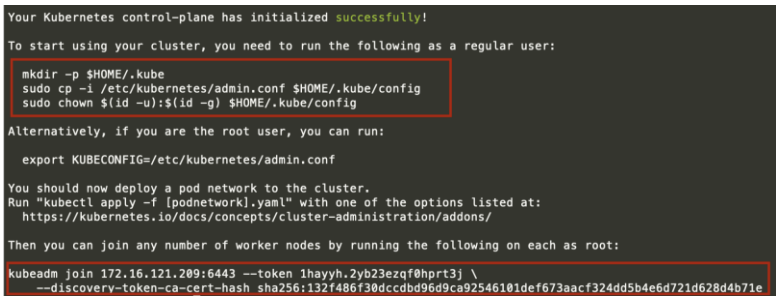


Figura 75. Estado Names Spaces. Fuente Autor

10. Después de un tiempo cuando ya todo se haya creado, probamos el clúster

    ```
    kubectl get nodes
    ```

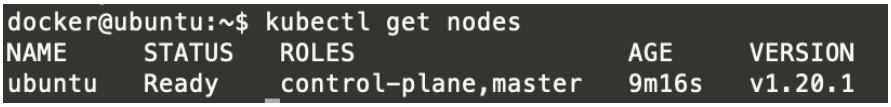


Figura 76. Estado de nodos. Fuente Autor

AÑADIR NODOS AL CLÚSTER

11. Nos conectamos al nodo que queremos incorporar al clúster. Es importante seguir los pasos que se encuentran en el Anexo 3 y ejecutar los pasos del 1 al 5 del Anexo 4.

    Ejecutamos el join que se ha indicado en el momento de hacer el

    `kubeadm init --pod-network-cidr=192.168.0.0/16.`

    Por ejemplo (este será diferente en cada instalación)

    ```
    kubeadm join 172.16.121.209:6443 --token 1hayyh.2yb23ezqf0hprt3j \
        --discovery-token-ca-cert-hash sha256:132f486f30dccdbd96d9ca92546101def673aacf324dd5b4e6d721d628d4b71e
    ```

12. Nos dirigimos al nodo maestro y comprobamos que se hayan agregado los nodos correctamente

    `kubectl get nodes`

ANEXO 5: INSTALACIÓN DE KUBERNETES EN AMAZON WEB SERVICES (AWS)

Vamos a crear el clúster, para ello en servicios buscamos EKS y asignamos un nombre para nuestro clúster y damos en Next.

Figura 77. Agregar un nombre para crear el clúster EKS. Fuente Autor

Paso 1. Configurar el clúster: obtiene el nombre que se ingresó y automáticamente se asigna la versión de Kubernetes en Amazon, que va siempre una versión por detrás de la release oficial.

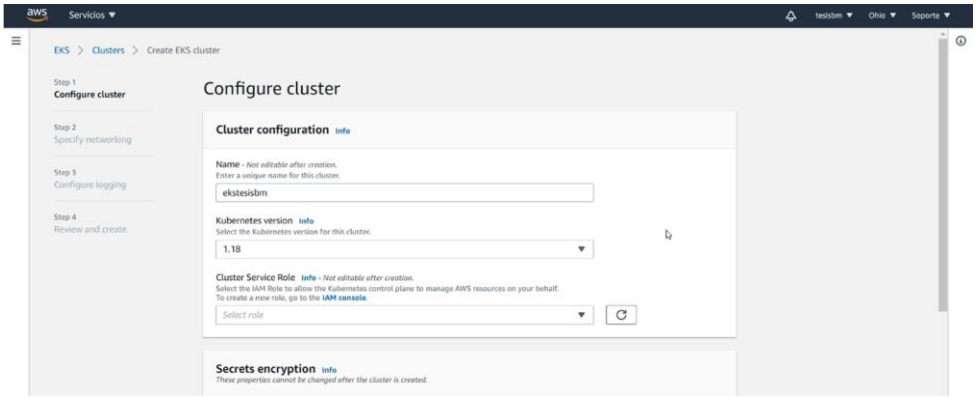


Figura 78. Configurar el clúster. Fuente Autor

En este paso necesitamos crear un rol de servicio para realizar tareas de control y gestión sobre el cluster, para ello, damos clic en la opción IAM console y se abrirá una ventana llamada Identity and Access Management (IAM) y damos clic en Crear un rol.

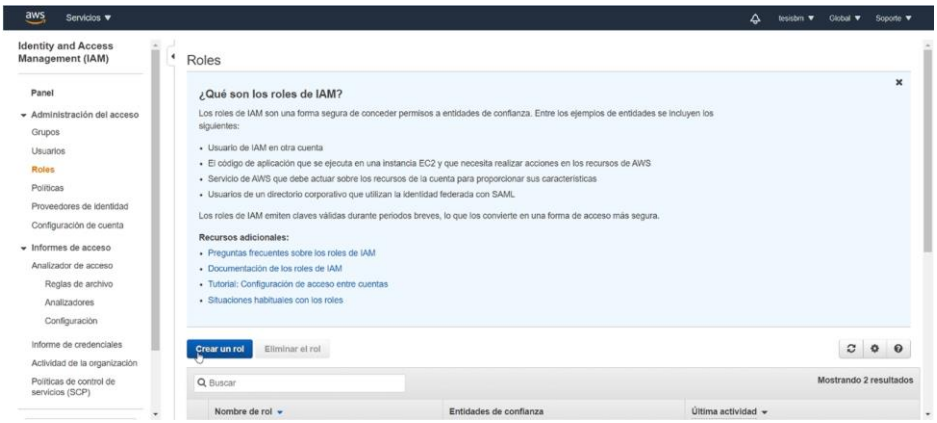


Figura 79. Crear un rol. Fuente Autor

Elegimos el servicio EKS y de los casos de uso que aparecen seleccionamos EKS-Cluster y presionamos en el botón siguiente.

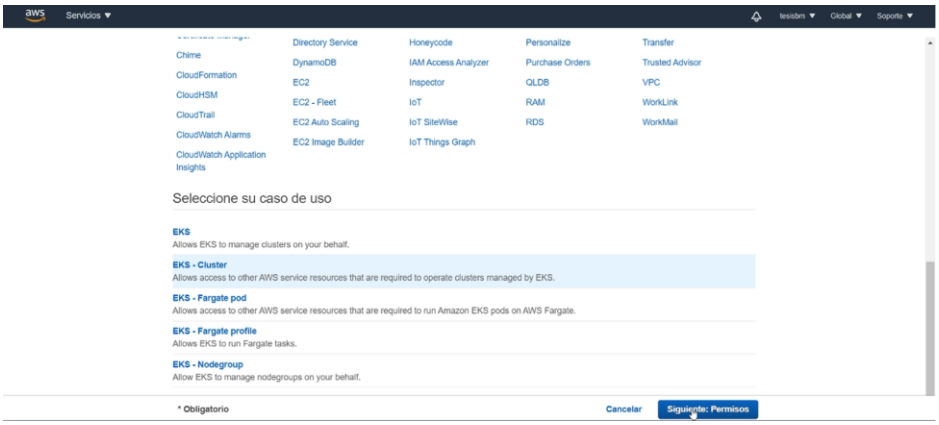


Figura 80. EKS – Cluster. Fuente Autor

Automáticamente debe salir la política asociada al rol, posteriormente damos clic en siguiente.

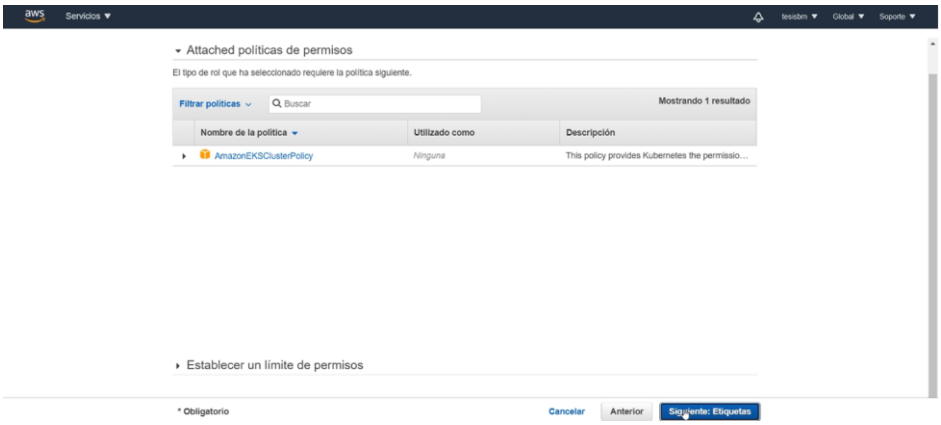


Figura 81. Política asociada al rol. Fuente Autor

El paso de añadir etiquetas podemos saltarnos

Figura 82. Añadir etiquetas. Fuente Autor

Revisamos el rol antes de crearlo. Debemos asignar un nombre al rol y finalmente damos clic en la opción Crear un rol.

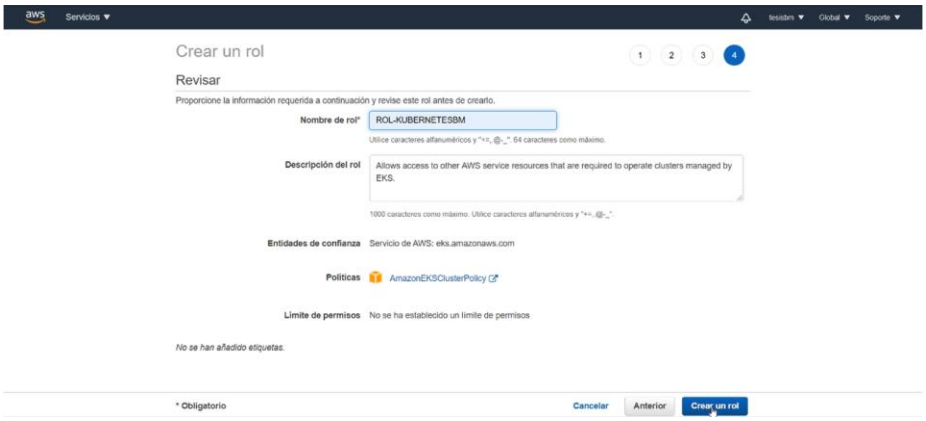


Figura 83. Revisar rol antes de crear. Fuente Autor

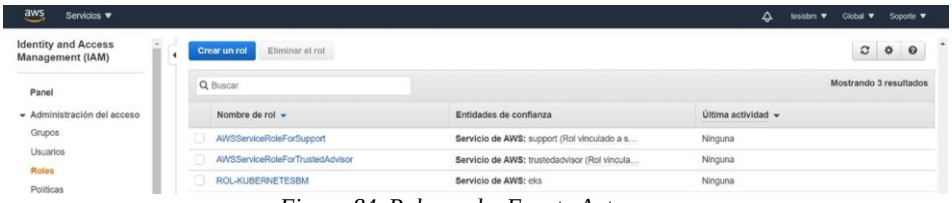


Figura 84. Rol creado. Fuente Autor

Una vez creado el rol, regresamos a la pestaña donde estábamos creando el clúster, al refrescar la lista de roles, deberá aparecer el rol creado, lo seleccionamos y damos clic en Next.

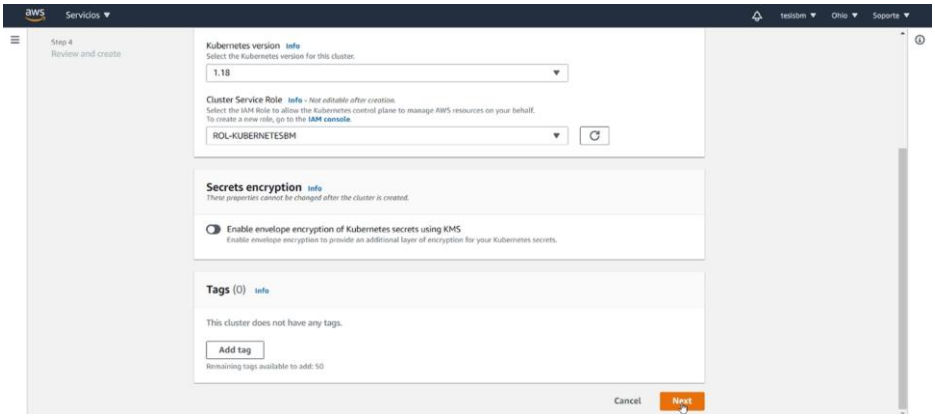


Figura 85. Seleccionar el rol. Fuente Autor

Paso 2. Especificar redes: seleccionamos la VPC que habíamos creado, automáticamente muestra las dos subredes públicas, las dos subredes privadas y también seleccionamos el grupo se seguridad.

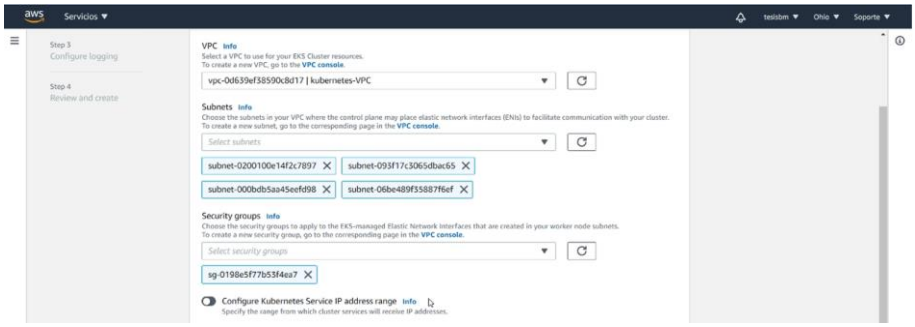


Figura 86. Seleccionar la VPC. Fuente Autor

En Acceso al punto final del clúster, seleccionamos Public and Private que es la opción que hemos seleccionado para crear el clúster.

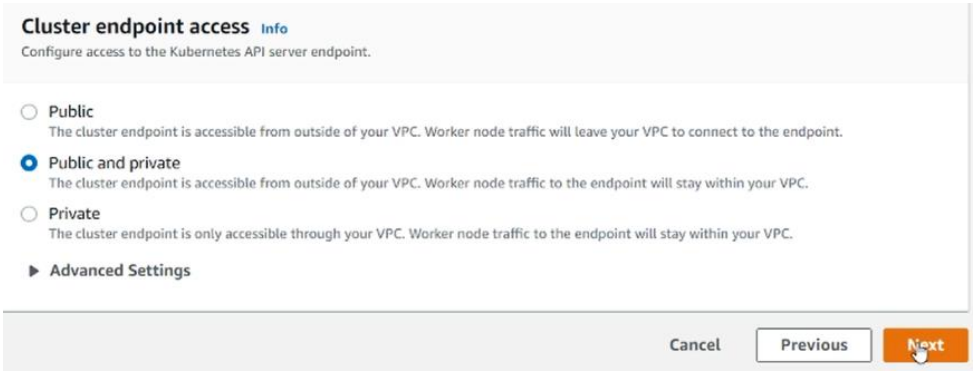


Figura 87. Acceso al punto final del clúster. Fuente Autor

Paso 3. Configurar el registro: este paso lo podemos saltar, considerando que parte de estos productos son de pago, por tanto, nos consulta si queremos dejar la información de login en un producto llamado Cloud Watch, que básicamente gestiona logs.

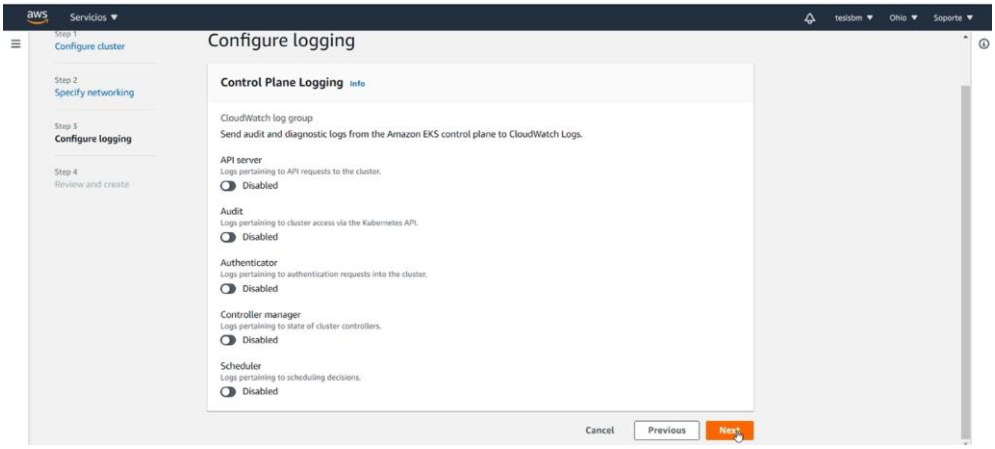


Figura 88. Configurar el registro. Fuente Autor

Paso 4. Revisar y crear: hacemos una revisión de los pasos configurados anteriormente y finalmente presionamos sobre el botón Create, este proceso de creación puede tardar varios minutos y como resultado se creará nuestro máster de Kubernetes.

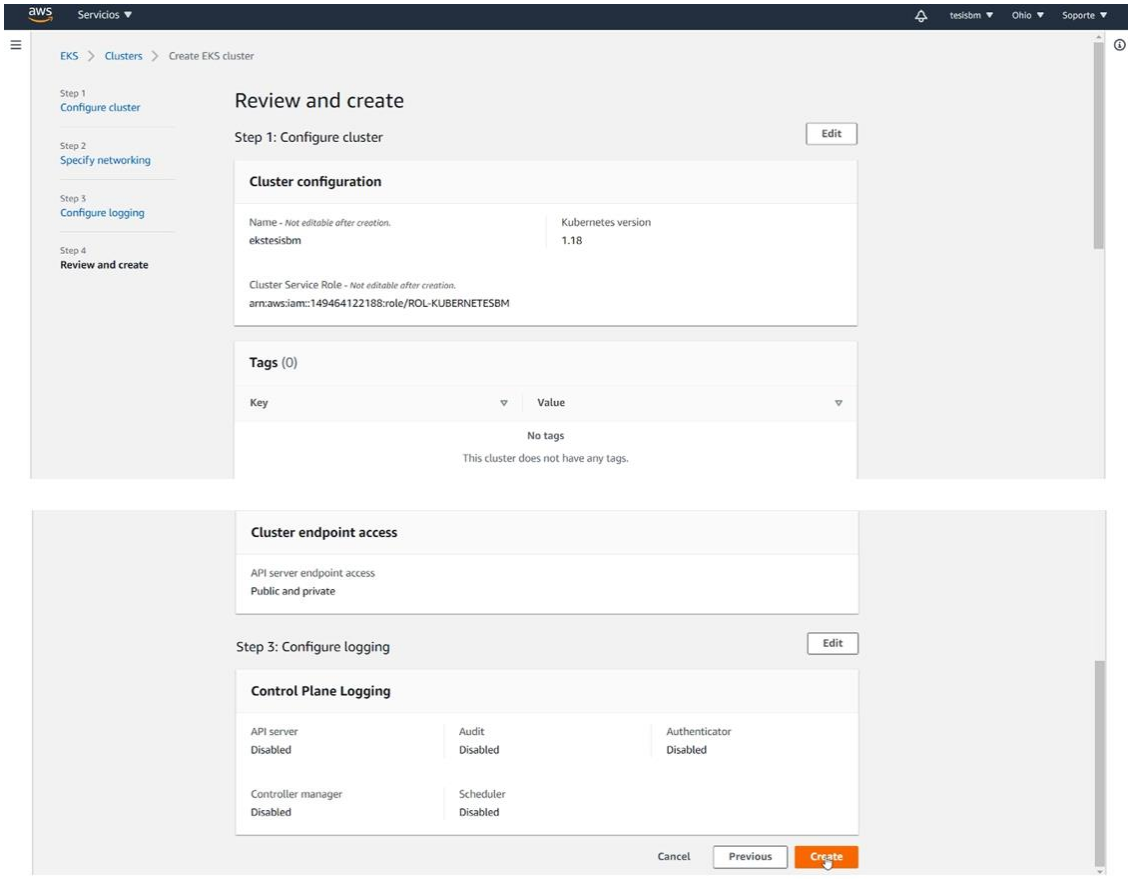


Figura 89. Revisar y crear el clúster. Fuente Autor

CREAR NODOS EN AMAZON (AWS)

1. Ingresamos al clúster creado y nos dirigimos a la pestaña Compute.

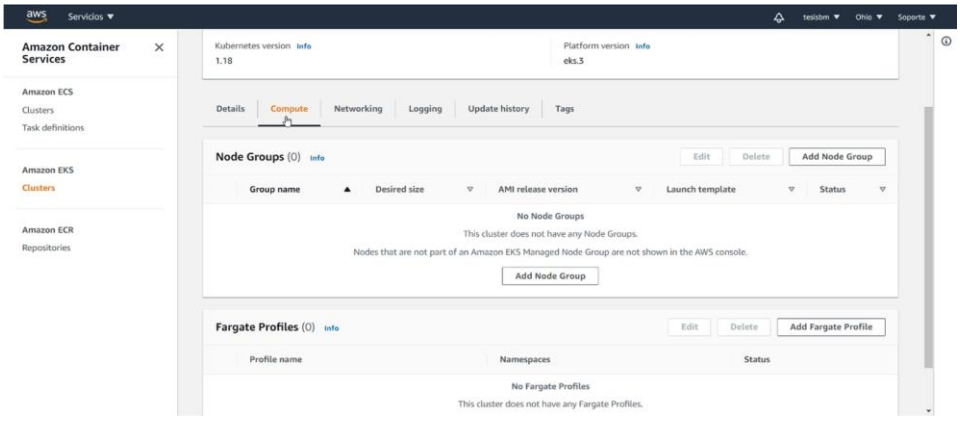


Figura 90. Pestaña Compute. Fuente Autor

2. Para añadir el grupo de nodos, damos clic en Add Node Group y seguimos los siguientes pasos.

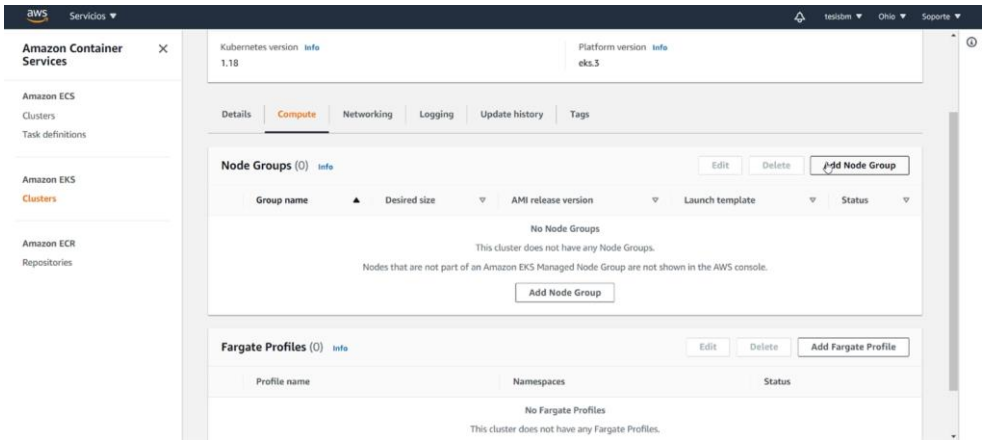


Figura 91. Agregar grupos de nodos. Fuente Autor

Paso 1. Configurar el grupo de nodos: proporcionamos un nombre para el grupo.

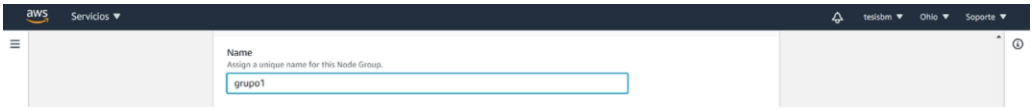


Figura 92. Asignar nombre para el grupo de nodos. Fuente Autor

En Node IAM Role, debemos seleccionar un rol para gestionar la parte de los nodos, antes necesitamos crearlo dirigiéndonos a la pestaña IAM console.

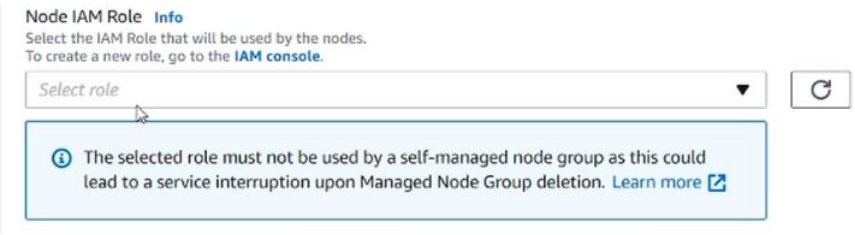


Figura 93. Rol para gestionar los nodos. Fuente Autor

Se abrirá una nueva pestaña. Aquí damos clic en el botón Crear un rol.

Figura 94. Crear un rol. Fuente Autor

Elegimos el caso de uso común EC2 y presionamos en siguiente.

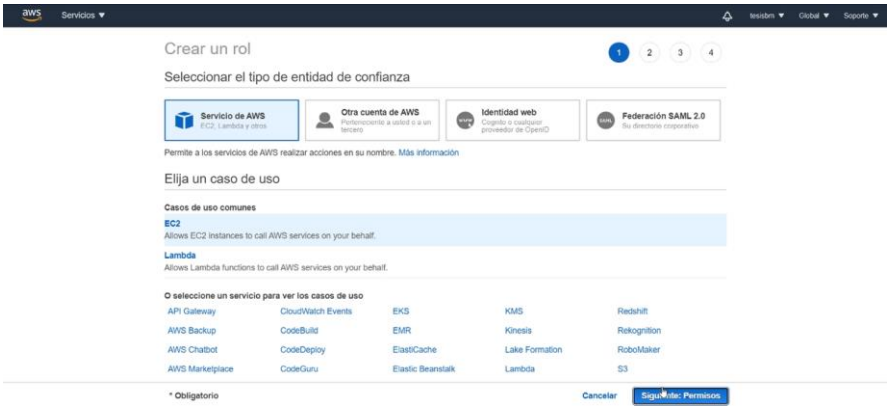


Figura 95. Caso de uso común EC2. Fuente Autor

En este caso vamos a seleccionar tres políticas para que funcione el rol, estas son:

AmazonEC2ContainerRegistryReadOnly

AmazonEKS_CNI_Policy

AmazonEKSWorkerNodePolicy

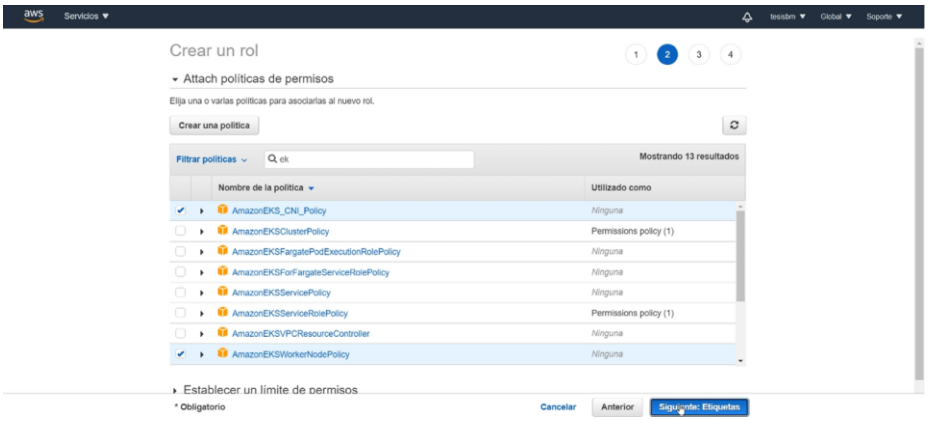


Figura 96. Políticas para que funcione el rol. Fuente Autor

Damos en siguiente.

Nos saltamos el paso de Añadir etiquetas.

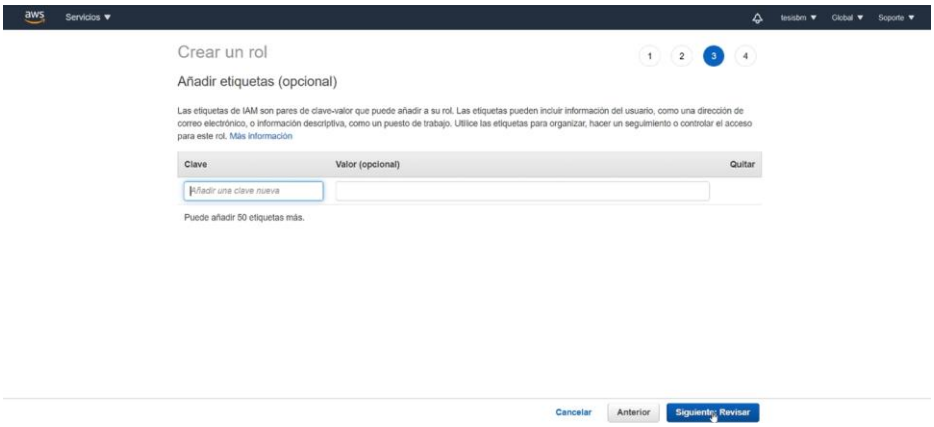


Figura 97. Añadir etiquetas del rol para gestionar los nodos. Fuente Autor

Antes de crear el rol, asignamos un nombre, revisamos y finalmente presionamos sobre el botón Crear un rol.

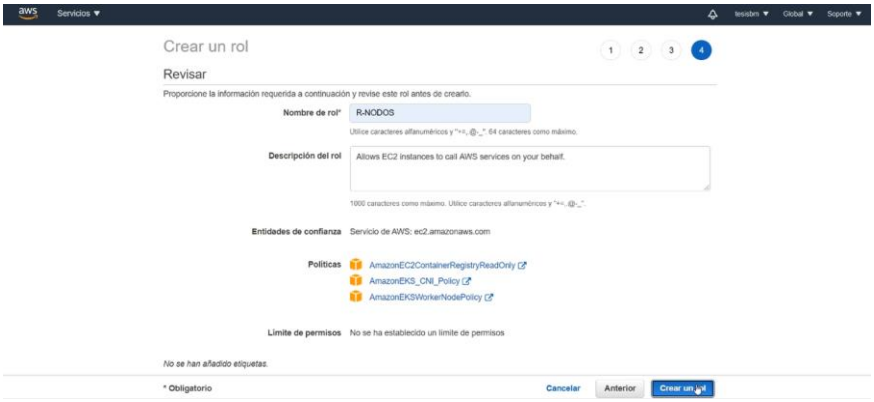


Figura 98. Asignar un nombre y crear un rol. Fuente Autor

Regresamos a la pestaña donde estamos creando el grupo de nodos, refrescamos Node IAM Role, seleccionamos el rol y damos en Next.

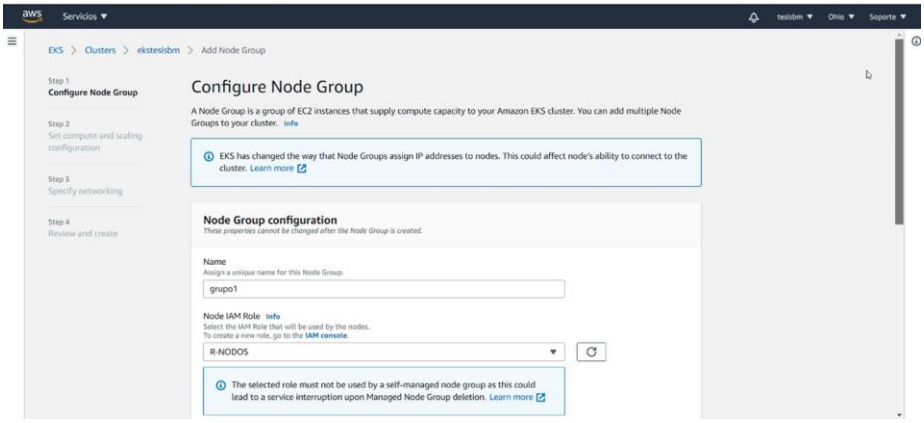


Figura 99. Configurar el grupo de nodos. Fuente Autor

Paso 2. Establecer la configuración de computación y escalado: debemos seleccionar el tipo de máquina para los nodos, aunque por defecto solo se soportan máquinas de tipo Amazon Linux, luego en tipo de instancia seleccionamos t3.medium que son de 4 GiB, no debemos poner más bajo porque caso contrario no van a arrancar, dejamos los 20 GB de disco. En este paso, también indicamos el número de nodos que vamos a crear y damos Next.

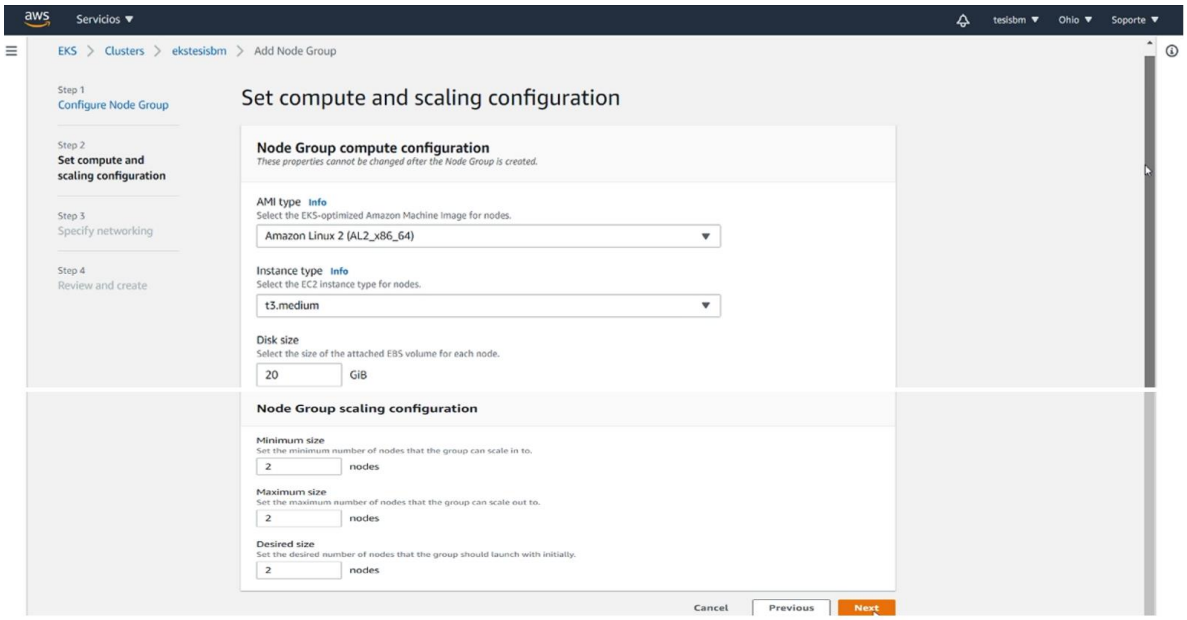


Figura 100. Configuración de computación y escalado. Fuente Autor

Paso 3. Especificar redes: se cargarán automáticamente las subredes, necesitamos seleccionar una SSH key pair, para ello vamos a crearla dando clic en EC2 console, se abrirá una nueva pestaña y damos clic en Crear par de claves.

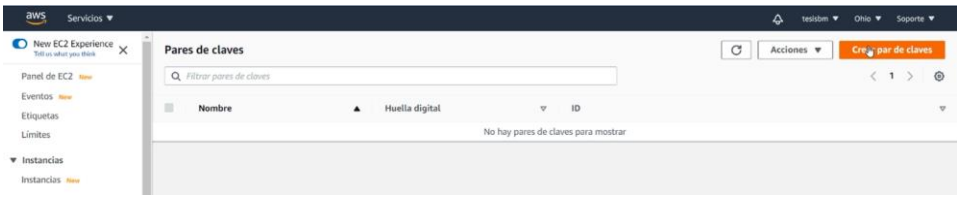


Figura 101. Crear par de claves. Fuente Autor

Asignamos un nombre y seleccionamos el formato para que se descargue, pem para OpenSSH o formato ppk para PuTTY que es la que descargamos en nuestro caso, damos clic en Create key pair.

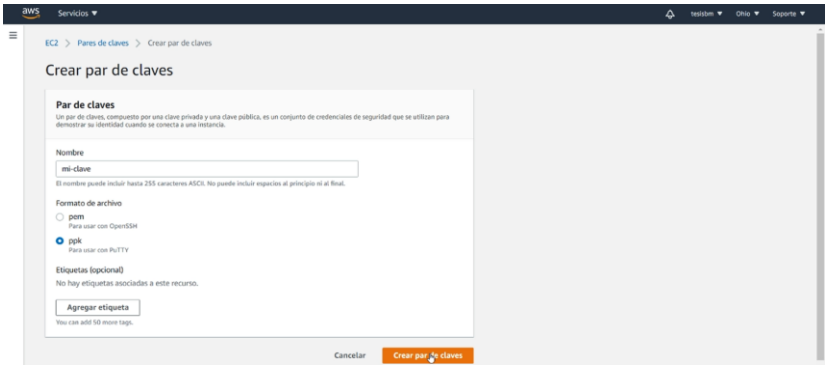


Figura 102. Asignar un nombre y seleccionar el formato. Fuente Autor

Se crea y se descarga dentro de nuestro PC.

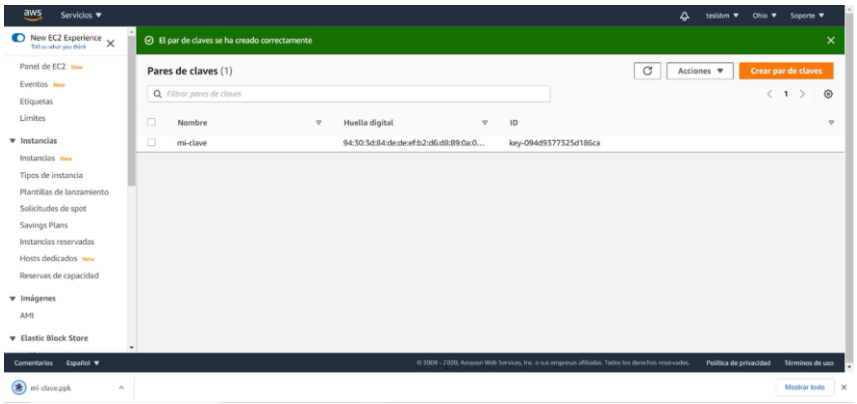


Figura 103. Clave creada y descargada. Fuente Autor

Volvemos a la pestaña donde estamos creando el grupo de nodos, refrescamos SSH key pair y seleccionamos la clave creada y damos clic en Next.

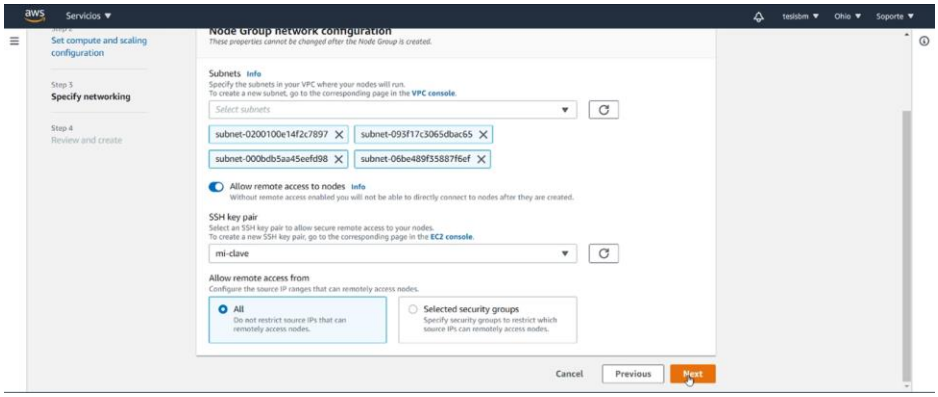


Figura 104. Especificar redes. Fuente Autor

Paso 4. Revisar y crear: antes de crear el grupo de nodos revisamos y finalmente damos clic en Crear, este proceso puede llevar tiempo para crear los nodos.

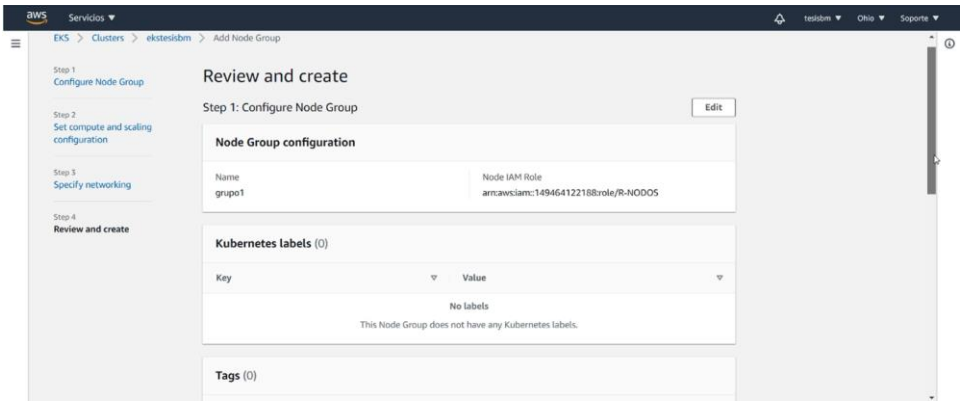


Figura 105. Revisar y crear el grupo de nodos. Fuente Autor

ANEXO 6: ARCHIVO DOCKERFILE

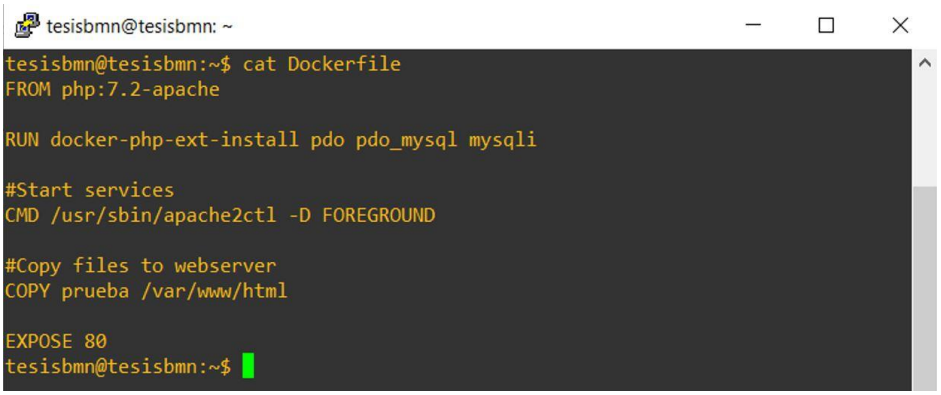


Figura 106. Archivo Dockerfile
Fuente Autor

ANEXO 7: ARCHIVO PHP-DEPLOYMENT.YAML

```
apiVersion: v1
kind: Service
metadata:
  name: webapp-sql
spec:
  selector:
    app: webapp-sql
    tier: frontend
  ports:
  - protocol: "TCP"
    port: 80
    targetPort: 80
  type: LoadBalancer

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp1
  labels:
    app: webapp-sql
    tier: frontend
spec:
  selector:
    matchLabels:
      app: webapp-sql
      tier: frontend
  replicas: 1
  template:
    metadata:
      labels:
        app: webapp-sql
        tier: frontend
    spec:
      containers:
      - name: webapp1
        image: dhtesisbm/prueba:latest
        imagePullPolicy: Always
      ports:
      - containerPort: 8081
```

ANEXO 8: ARCHIVO MYSQL-DEPLOYMENT.YAML

```
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  selector:
app: webapp-sql
tier: backend
  ports:
  - protocol: "TCP"
port: 3306
    targetPort: 3306
  clusterIP: None
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: sqldb
  labels:
app: webapp-sql
tier: backend
spec:
  selector:
    matchLabels:
       app: webapp-sql
       tier: backend
  strategy:
type: Recreate
  template:
metadata:
       labels:
       app: webapp-sql
       tier: backend
spec:
      containers:
       - image: mysql:5.6
       name: mysql
       env:
       - name: MYSQL_ROOT_PASSWORD
          value: bdpasstesisbm
       - name: MYSQL_DATABASE
          value: db_banco
       - name: MYSQL_USER
          value: db_user
       - name: MYSQL_PASSWORD
          value: bdpasstesisbm
       args: ["--default-authentication-plugin=mysql_native_password"]
       ports:
       - containerPort: 3306
        volumeMounts:
       - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
       volumes:
       - name: mysql-persistent-storage
        persistentVolumeClaim:
       claimName: mysql-pv-claim
```

ANEXO 9: ARCHIVO MYSQL-PV.YAML

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv-volume
  labels:
type: local
spec:
  storageClassName: manual
  capacity:
storage: 20Gi
  accessModes:
- ReadWriteOnce
  hostPath:
path: "/mnt/data"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
spec:
  storageClassName: manual
  accessModes:
- ReadWriteOnce
  resources:
requests:
      storage: 1Gi
```