ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN PROTOTIPO DE UN SERVIDOR WEB CON HOSTS VIRTUALES MEDIANTE KUBERNETES

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

JEFFERSON DARIO MONTES BRAVO

DIRECTOR: FERNANDO VINICIO BECERRA CAMACHO

DMQ, AGOSTO 2022

CERTIFICACIONES

Yo, JEFFERSON DARIO MONTES BRAVO declaró que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Jefferson Montes

jefferson.montes@epn.edu.ec

jefersondario20@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por JEFFERSON DARIO MONTES BRAVO, bajo mi supervisión.

Fernando Becerra

DIRECTOR

Fernando.becerrac@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el producto resultante del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

JEFFERSON DARIO MONTES BRAVO

DEDICATORIA

Dedico este logró a toda mi familia en especial a mis padres que fueron mi mano derecha formándome como persona y estando siempre presente incluso en los momentos difíciles durante esta etapa.

Este presente proyecto, es dedicado de igual manera a mi familia, amigos y personas especiales para mi quienes han aportado con sus enseñanzas hasta convertirse en una parte fundamental de este proceso.

AGRADECIMIENTO

Agradezco de todo corazón a dios por haberme otorgado a unos padres maravillosos quienes me han brindado su confianza y apoyo incondicional, dándome un ejemplo de superación, perseverancia y humildad, de esta manera, enseñándome a valorar todo lo que tengo.

Un agradecimiento especial a mi hermano quien me ha otorgado un estado de paz y brindado momentos de alegría, además, no solo por estar presente en los buenos momentos de mi vida, sino también en momentos duros de lucha para conseguir este logro.

Ing. Fernando Becerra persona con gran sabiduría quien me ayudó a llegar a este punto en el que me encuentro, además, de transmitir sus conocimientos para lograr conseguir este objetivo que es la culminación de este proyecto.

Agradezco a mis colegas y amigos por el apoyo y compañía en estos años compartidos que me enseñaron y me dieron ánimos.

ÍNDICE DE CONTENIDOS

RESUMEN

El presente proyecto integrador, tiene como objetivo la investigación y estudio acerca de la plataforma kubernetes y la implementación de *hosts* virtuales dentro de un servidor web, de igual manera, al menos tres formas sencillas para desplegar un clúster de kubernetes. Adicional, de las tres formas antes mencionadas se seleccionará una de ellas para ser implementada en un sistema operativo Linux.

Con un clúster de kubernetes desplegado se implementó un prototipo de servidor web con *hosts* virtuales mediante un lenguaje de programación YAML que permitió la elaboración de ficheros con extensión YML que fueron empleados para levantar los diferentes componentes para brindar los servicios a los correspondientes *hosts* y de esta manera hacer énfasis al desempeño de su utilización para el despliegue de aplicaciones y servicios.

En el primer capítulo se describe el desarrollo y se expone el contenido del mismo, a su vez, el objetivo general y los objetivos específicos, el alcance y el marco teórico.

El segundo capítulo va a hacer énfasis en la metodología manejada para el desarrollo del presente proyecto.

En el tercer capítulo contiene los resultados, en donde se expone la investigación sobre DevOps, kubernetes, servidores web y *hosts* virtuales, además, algunas maneras de desplegar un clúster de kubernetes. Posteriormente, la implementación del mismo mediante una aplicación determinada, se presenta su instalación, creación de los componentes y servicios dentro de un sistema operativo Ubuntu.

Por último, el cuarto y quinto capítulo se encuentran las conclusiones y recomendaciones correspondientemente, las mismas que se fueron obteniendo conforme el desarrollo del proyecto.

PALABRAS CLAVE: Kubernetes, clúster, DevOps, servicios, *hosts* virtuales.

ABSTRACT

The objective of this integrative project is to investigate and study the kubernetes platform and the implementation of virtual hosts within a web server, as well as at least three simple ways to deploy a kubernetes cluster. Additionally, of the three ways mentioned above, one of them will be selected to be deployed on a Linux operating system.

With a kubernetes cluster deployed, a prototype web server with virtual hosts was implemented using a YAML programming language that allowed the development of files with YML extension that were used to raise the different components to provide services to the corresponding hosts and thus emphasize the performance of its use for the deployment of applications and services.

In the first chapter the development is described and the content of the same is exposed, as well as the general objective and the specific objectives, the scope and the theoretical framework.

The second chapter will emphasize the methodology used for the development of this project.

The third chapter contains the results, where the research on DevOps, kubernetes, web servers and virtual hosts is exposed, as well as some ways to deploy a kubernetes cluster. Subsequently, the implementation of the same by means of a certain application, its installation, creation of the components and services within an Ubuntu operating system are presented.

Finally, the fourth and fifth chapters contain the corresponding conclusions and recommendations, which were obtained during the development of the project.

KEYWORDS: *kubernetes, cluster, DevOps, services, virtual hosts.*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El presente proyecto integrador está compuesto por varias etapas esenciales para su desarrollo, en la primera parte abarca una investigación sobre definiciones básicas acerca de DevOps, Kubernetes, servidores web y *hosts* virtuales. En esta etapa se detallan las características de manera específica de cada uno de los elementos antes mencionados, entre las especificaciones que se investigaron fueron: funcionamiento, definiciones acerca de los elementos que los componen y por último la manera de implementar cada uno de ellos. De igual manera, se realizó la comparación entre dos herramientas de DevOps que fueron Docker y Ansible con la plataforma Kubernetes, para diferenciar las características, ventajas y desventajas de cada una de ellas.

Comprendido las definiciones y singularidades de cada componente antes mencionado se presenta una investigación acerca de tres maneras de implementar un clúster de kubernetes de manera sencilla y ágil, estas son: Play With Kubernetes, Minikube y Google Kubernetes Engine, se define los conceptos, ventajas y su forma de implementar en su respectivo ambiente.

Luego de conocer cada opción que ofrece kubernetes para su despliegue, se determina que la herramienta minikube cumple con los estándares para implementar un prototipo de servidores web con *hosts* virtuales dentro de un clúster de kubernetes.

Finalizada la etapa de investigación y determinación de la aplicación a manejar para el despliegue de un clúster de kubernetes se procede a la implementación del mismo dentro de una máquina virtual con un sistema operativo Ubuntu. En esta etapa se detalla paso a paso la instalación de la herramienta minikube con los componentes necesarios para su óptimo funcionamiento. Se verifica la instalación mediante el arranque del clúster y se accede al plano de control para verificar que no existe ningún elemento extra ejecutándose, además, se coloca algunos comandos propios de minikube mediante la *shell* para verificar su desempeño.

Concluida la etapa del despliegue del clúster y verificado el funcionamiento del mismo, se desarrolla la creación de ficheros con extensión YML (*Java Modeling Language*), para el levantamiento de los servicios y el controlador de ingreso para direccionar el servicio a los pods. En esta etapa se encuentra detallada la elaboración de los objetos como: pods, servicios y despliegues para la construcción de servidores webs por medio de instrucciones con ayuda de un lenguaje de programación YAML (*Yet Another Markup Language*). Adicional, se asignan los nombres de dominio que se va a hacer referencia

para acceder a cada uno de los servicios webs, asimismo, se presenta la ejecución de los ficheros para la creación de los servicios, pods y despliegues.

1.1 Objetivo general

Implementar servidores con herramientas de DevOps.

1.2 Objetivos específicos

- Analizar kubernetes y servidores web mediante host virtuales.
- Diseñar una aplicación en Kubernetes que permita la implementación de servidores web.
- Implementar el diseño de kubernetes de servidores web virtuales.
- Verificar el funcionamiento de la aplicación desarrollada en Kubernetes.

1.3 Alcance

Por medio del presente proyecto integrador se realizará un estudio de DevOps y servidores web para conseguir aplicarlo. Además, se busca obtener conocimiento sobre la plataforma portable llamada kubernetes a su vez características, elementos y funcionamiento. Asimismo, se realiza una investigación acerca del despliegue de un clúster en un sistema operativo Linux.

Por otro lado, se determinará la opción rápida y sencilla para el despliegue del clúster de kubernetes que será implementado dentro de una máquina virtual con un sistema operativo Ubuntu. Posteriormente, se implementará un prototipo que permita el manejo del mismo con la finalidad de desplegar un servidor web y de igual manera *hosts* virtuales.

1.4 Marco Teórico

Contenedores

Los contenedores son una tecnología utilizada para aislar las aplicaciones virtualmente, de igual forma, los entornos de ejecución con el objetivo de ejecutar aplicaciones y facilitar su traslado. Asimismo, permite levantar microservicios de forma rápida para que puedan acceder al *kernel* del sistema operativo sin la necesidad de máquinas virtuales garantizando que se ejecute correctamente cuando ocurra un cambio del entorno [1].

Los contenedores permiten proporcionar un mejor funcionamiento en comparación a una virtualización, dado que, la tecnología de virtualización permite simular un sistema operativo, entre las características que son simuladas se puede mencionar: CPU,

memoria, almacenamiento y conexiones de red. Generalmente, se la suele implementar con un software llamado hipervisor, en cambio, al utilizar un contenedor simplemente hay que llevar el software necesario dentro del mismo permitiendo un consumo computacional menor a comparación de una virtualización, convirtiéndolos en una opción más ligera y portátil [1].

Las tecnologías que se encuentran basadas en contenedores nacieron a partir del concepto de particionamiento del hardware y del proceso de aislamiento llamado *chroot*, el cual consiste en aislar un proceso de todos los demás recursos de los que son permitidos [1].

Para aplicaciones que exigen una mayor capacidad se pueden implementar varios contenedores o a su vez varios clústeres que puedan ser gestionados mediante un orquestador, como lo es Kubernetes. En la **Figura 1.1** se puede observar que se tiene una independencia de las librerías como también de las aplicaciones las cuales están aisladas en el sistema operativo [1].

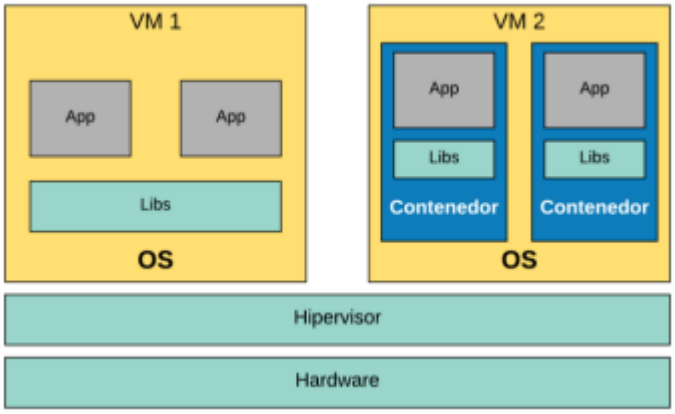

Figura 1.1 Comparación entre contenedores y Virtualización [1]

Docker

Docker es una plataforma que permite la construcción, implementación y verificación de aplicaciones de una manera sencilla y ágil. El funcionamiento de Docker consiste en el empaquetamiento de software en unidades denominadas contenedores, asimismo, los parámetros necesarios para que puedan ejecutar el software correspondiente, están incluidos en ellos: bibliotecas, códigos, herramientas del sistema y de igual forma el tiempo de ejecución [1], [2].

Docker es considerado un sistema operativo enfocado para contenedores es relativamente parecido a una máquina virtual, por lo que se caracteriza por ser instalado en un servidor y por los comandos sencillos para la creación, ejecución y detención de contenedores [1], [2].

Por otro lado, en cuanto a los beneficios que aporta esta herramienta se puede mencionar que posee una gran adaptabilidad por parte de las empresas sobre todo por la entrega de código con mayor rapidez. Además, Docker tiene una amplia gama de herramientas, aplicaciones y estandarización de las mismas conservando la facilidad, el ahorro de tiempo y el uso de recursos. Se aprecia en **Figura 1.2** la comparación de la administración de contenedores versus una máquina virtual y los factores que influyen en cada una de estas tecnologías [2].

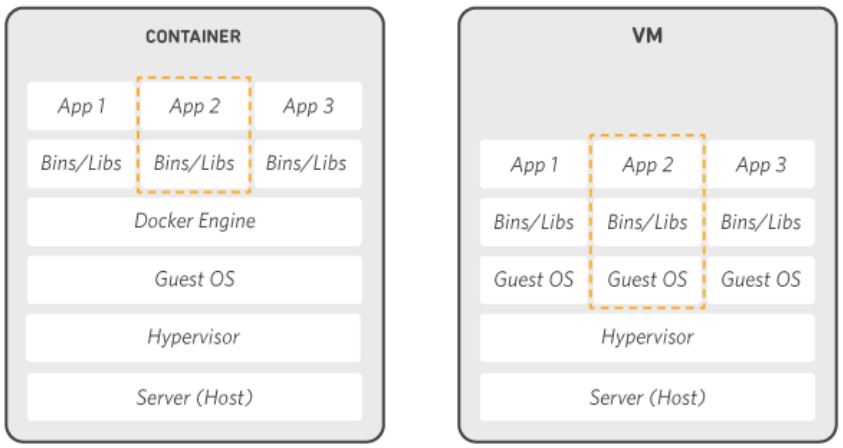

Figura 1.2 Administración entre contenedores y máquinas virtuales [2]

Kubernetes

Kubernetes o también conocido como k8s o kube, es una plataforma portable de código abierto que brinda la facilidad de gestionar las diversas tareas y servicios, permitiendo un entorno más fácil para la automatización y configuración de aplicaciones en múltiples *hosts*. Además, esta plataforma se centra en la administración de servidores para que las cargas de trabajo de los administradores de red sean realizadas de forma automática sin que ellos tengan que desplegarlo permitiendo la portabilidad de los servicios de infraestructura y aumentando la eficiencia del departamento de TI (tecnología e información) [1], [3], [4].

Al igual que el aplicativo docker, Kubernetes tiene la capacidad de ser desplegado desde cero desde cualquier ambiente ya sean estos una *cloud computing* o a su vez una máquina local. La utilización de esta herramienta no se encuentra limitada a grandes empresas, sino también, para pretensiones a menor escala. Los desarrolladores optan por kubernetes para crear aplicaciones fuera del mercado, dado que, posee una interfaz capaz de satisfacer las diferentes necesidades para el desarrollo de una aplicación. Por otro lado, la herramienta kubernetes ofrece funcionalidades como: el despliegue de contenedores, almacenamiento, monitorización de contenedores y servicios [1], [3], [4].

En cuanto a la orquestación de contenedores, Kubernetes permite el escalonado de contenedores y servicios, también, posee un clúster robusto para brindar confiabilidad y seguridad en cuanto a su uso. En la **Figura 1.3** se visualiza el funcionamiento y arquitectura que posee Kubernetes cuando se encuentra desplegado un servicio.

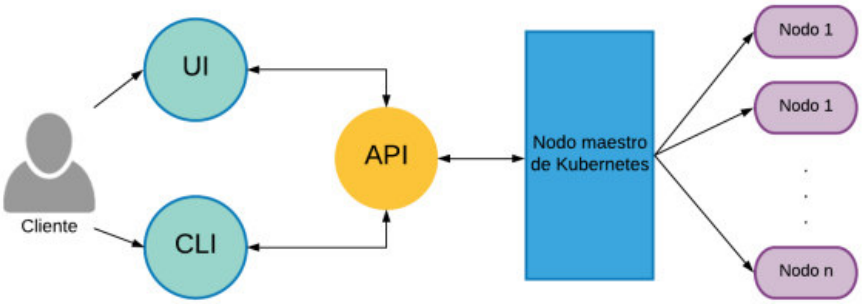

Figura 1.3 Funcionamiento y arquitectura de kubernetes [1]

DevOps

El término DevOps nace de la unión de dos terminaciones inglesas *development* y *operations* qué significa desarrollo y operaciones correspondientemente. DevOps es un modelo que se enfoca en el trabajo conjunto de personas, procesos y tecnologías. Para que de esta manera el valor empresarial se vea aumentado de una manera constante proporcionando servicios de TI de alta calidad. Esto quiere decir, que los roles que antes se consideraban aislados como el desarrollo, la calidad y la seguridad se encuentren coordinados y trabajen juntos para crear mejores productos o servicios. Por ese motivo los equipos que trabajen bajo este modelo obtienen la capacidad de dar una adecuada respuesta a las necesidades del usuario aumentando la fiabilidad y el alcance de la aplicación [5], [6].

2 METODOLOGÍA

En primera instancia se analizó las características acerca del modelo de DevOps, en el cual se destacó las características principales que fue obtenida de materiales didácticos como lo son libros, artículos científicos de esta metodología. Además, los conceptos que se encuentran adjuntos a dicha herramienta y a su vez el correcto funcionamiento de la misma, por consiguiente, se obtuvo conocimiento previo acerca del despliegue y funcionamiento de DevOps para su ejemplificación posteriormente.

Del mismo modo se investigó acerca de la herramienta kubernetes la cual se hizo énfasis en los elementos que incorpora esta tecnología y las características básicas que se obtuvieron de libros, guías de certificación y desde su plataforma web oficial para obtener información verídica para después comprender el funcionamiento de dicha

plataforma. Posteriormente, al conocer sobre la herramienta DevOps y al percatarse de ser un entorno muy amplio se procedió a comparar dicho aplicativo con dos herramientas que fueron docker y Ansible con Kubernetes para tener en cuenta las ventajas y las diferencias entre ellas.

Por otro lado, se realizó una investigación sobre las características de los servidores e implementación de *hosts* virtuales en dichos servidores para posteriormente realizar su despliegue de manera automática en un clúster de kubernetes mediante la elaboración de ficheros YML que contenga las características e instrucciones para su desarrollo.

Para la verificación del archivo que contiene las instrucciones del despliegue de los servicios web se lo realizó mediante comandos propios de kubernetes desde la *shell* del sistema operativo Ubuntu, además, se verificó por medio del plano de control que ofrece la herramienta que contribuyó al despliegue de un clúster de kubernetes la cual proporciona de manera visual los servicios ejecutándose correctamente.

3 RESULTADOS

En primera instancia, se realiza una investigación acerca de DevOps y algunas de las singularidades, también, se indagará conceptos de kubernetes y servidores web en los que se detallan los componentes y características. Comprendido el concepto de los elementos antes mencionados se plantea la incorporación de un clúster de kubernetes de diferentes maneras, las cuales fueron: Play With Kubernetes, Minikube y Google Kubernetes Engine, seleccionando la herramienta minikube para el despliegue. Con el clúster de kubernetes instalado en el sistema operativo Ubuntu se tiene un único nodo siendo capaz de realizar las funciones de maestro y esclavo para la orquestación de contenedores, siendo el mismo en el cual se implementa servidores web con virtual *host* mediante una elaboración de un código en un archivo YML.

3.1 Analizar kubernetes y servidores web mediante host virtuales.

DevOps

La cultura DevOps representa a una combinación de *Development* (Dev) y *Operations* (Ops) que fue introducido en los años 2007- 2009 por Patrick Debois, Gene Kim y John Wilis, la cual se considera un conjunto de prácticas enfocadas al desarrollo de aplicaciones y las operaciones de TI.

El objetivo de DevOps es mejorar la comunicación continua mediante la colaboración entre las dos entidades que conforman esta cultura para impulsar el ciclo de

retroalimentación con los clientes de una forma continua y rápida. Gracias a que la relación entre Dev y Ops ha mejorado, las barreras se encuentran reducidas entre desarrolladores y operaciones de TI, proporcionando una mejora en la innovación, asimismo, en la entrega de servicio de forma que sea más rápida y confiable garantizando de esta manera la estabilidad en los sistemas de producción. Por consiguiente, ahorrando recursos para la empresa dado al rápido despliegue y la calidad gracias a la inclusión de la cultura DevOps en ella aumentando el valor comercial [7]. Para introducción esta cultura de DevOps necesita varios parámetros para su óptimo funcionamiento como lo son:

- Integración Continua. – La integración continua o *Continuous integration* (CI) en inglés, es un procedimiento automático que consiste en la verificación de la integridad de un código cada vez que este cambie o sea modificado por parte del equipo de trabajo [7].

  Como lo menciona Martin Fowler "*La integración continua es una práctica de desarrollo de software donde los miembros de un equipo integran su trabajo con frecuencia... Cada integración es verificada por una compilación automatizada (incluida la prueba) para detectar errores de integración tan rápido como sea posible*". En otras palabras, la IC se define como la forma de automatizar los procesos de compilación y verificación para que de esta manera el software se encuentre correcto y sin errores. CI permite brindar una ayuda extra a los equipos de desarrolladores para que cuando se modifique el código o falle evite problemas en su integración [7]. En la **Figura 3.1** se puede apreciar la ejecución de una CI completa y a su vez optimizada con el cual el grupo de desarrolladores puede solucionar problemas en cuanto a su integración omitiendo el paso de debatirlo entre ellos.

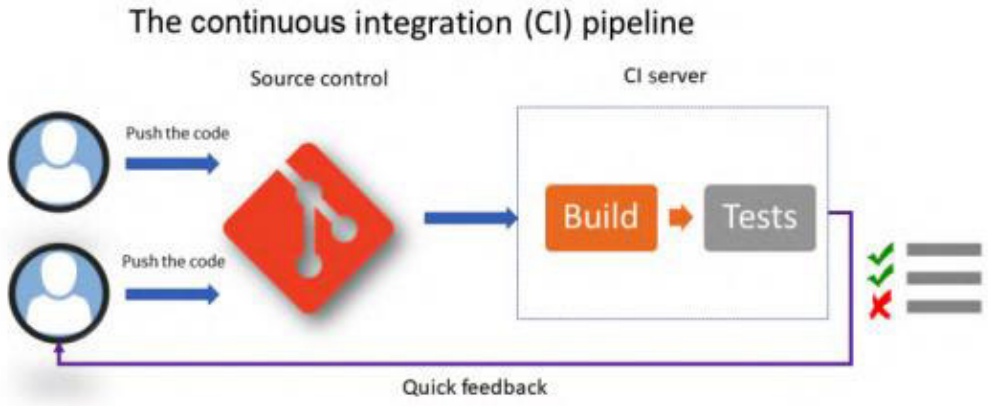

Figura 3.1 Fase de CI [7]

- Entrega Continua. – La entrega continua o *Continuous delivery* (CD) en inglés, se encuentra enfocado más allá de la automatización de la versión de software.

Por lo general, el proceso comienza con un paquete de aplicación preparado por parte de la CI que se instalará basado en una lista de acciones o tareas automatizadas, por ejemplo: descomprimir, copiar archivos, reiniciar y entre otros. El propósito de la CD es probar la aplicación con todos los parámetros y dependencias en un ambiente de pruebas [7]. En la **Figura 3.2** se puede estimar el ciclo de funcionamiento de la fase de entrega continua.

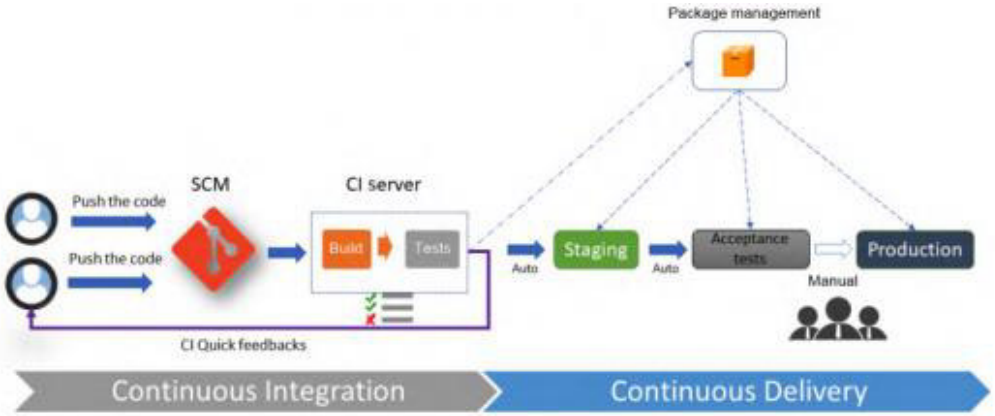

Figura 3.2 Fase de CD [7]

- Despliegue Continuo. – En el despliegue continuo cada cambio que se realiza pasa por todas las etapas de su canalización, dado que, es un proceso donde se automatiza la canalización de procesos anteriores de CI/CD provocando que no exista una interacción humana [7].

La cultura de DevOps se fundamenta en tres conceptos:

- La cultura de la colaboración. – Es considerada la particularidad de DevOps en pocas palabras, es el motivo de que los equipos ya no se encuentren separados de las funciones, por ejemplo: un equipo de desarrolladores y equipo de operaciones se encuentren conformadas por un solo equipo consiguiendo un equipo multidisciplinario [7].
- Procesos. – Para obtener un rápido despliegue se debe realizar procedimientos a partir de los cuales cumplan los objetivos de una mejor funcionalidad, mejorando la calidad y retroalimentación rápida [7].
- Herramientas. – Las herramientas de implementación son de suma importancia en DevOps, dado que, cuando los conceptos se separan cada equipo utiliza sus propias herramientas, por este motivo, las herramientas que se adopten deben ser usadas por todos los miembros [7].

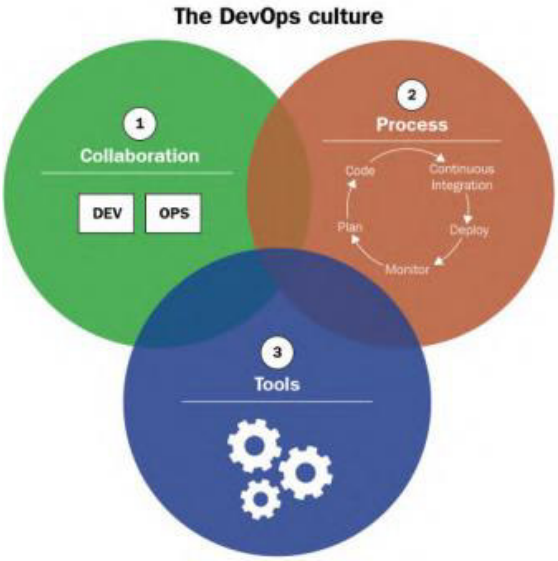

Figura 3.3 Pilares fundamentales de DevOps [7]

Del mismo modo, la implementación de la cultura DevOps en el ámbito empresarial trae consigo varios beneficios que ayudan a conseguir los objetivos.

- Aumento en la rentabilidad. – DevOps mejora la productividad mediante optimización de trabajo para dar apertura a nuevos proyectos [7].
- Mejora en la comunicación y colaboración. – Los diferentes departamentos son capaces de realizar una comunicación constante y retroalimentación con las diversas técnicas que permiten intercambiar información conforme el flujo de trabajo [7].
- Publicaciones de aplicaciones más rápidas. – Gracias a la práctica de IC y CD provocando que las tareas se automaticen y las pruebas de control se las realicen con mayor agilidad ha generado menor plaza de entrega lo cual provoca que exista una mejor eficiencia y rendimiento [7].
- Reducción de costos. – Con un aumento en la eficiencia y productividad de los empleados y el tiempo de entrega de las aplicaciones, esto conduce a que exista menos pérdidas y los recursos se encuentren aprovechados al máximo [7].
- Escalabilidad. – El sistema de DevOps brinda la facilidad de que un nuevo código sea añadido adaptándose a circunstancias cambiantes que requieran ampliación del mismo sin arriesgar la producción [7].

Kubernetes

El nombre Kubernetes proviene de una terminación griega *Helmsman* que hace referencia a la persona que gobierna un barco de navegación haciendo referencia a su logo, se lo puede apreciar en la **Figura 3.4**. Generalmente se lo abrevia como K8S, en el cual la numeración 8 representa a los 8 caracteres en medio de la K y la S [1], [8].

Kubernetes fue diseñado por google a partir del código de gestión de los centros de datos denominada *borg*, el cual más tarde fue donado a la comunidad como proyecto de código abierto, por consiguiente, se convirtió en la API (*Application Programming Interfaces*) estándar de la industria para la implementación y administración de aplicaciones de la nube nativa [1], [8].

Figura 3.4 Logo de Kubernetes [8]

Kubernetes es una plataforma de código abierto para la coordinación y administración de contenedores, además, permite utilizar las funciones de los servidores web para aplicaciones de la nube. K8S adicionalmente cumple funciones de un orquestador, es decir, es el encargado de administrar el ciclo de vida de los contenedores y agregar la funcionalidad necesaria para la ejecución de aplicaciones, en efecto: la infraestructura de redes, almacenamiento y recursos computacionales para que de esta manera la carga de trabajo sea alivianada. De igual modo, Kubernetes permite la automatización de muchos de los procesos manuales que están involucrados en la gestión, implementación y ampliación de aplicaciones en contenedores [1], [9], [10].

Clúster

El clúster de kubernetes consta de dos partes: el plano de control y el plano de aplicación o nodos. El plano de control es el que contiene a la API, la cual es cómo el usuario interactúa con el clúster y dice que hacer mediante la línea de comandos usando un organizador (kubectl). A su vez, cada nodo puede ser a su vez una máquina física o una virtual, realizando la misma máquina las tareas requeridas por el plano de control [9].

Objetos de Kubernetes

Kubernetes contiene varias entidades que representan el estado del sistema las cuales permiten obtener información sobre las aplicaciones, las cargas de trabajo que se encuentran en el contenedor, acerca de la red y recursos de almacenamiento asociados, así como información adicional sobre las operaciones actuales del clúster. Estas entidades están representadas por objetos API de Kubernetes [1], [3], [10].

La API de Kubernetes es una interfaz para el plano de control de Kubernetes y administra la interacción entre el usuario y el clúster. La API es utilizada para la gestión, creación y configuración de los clústeres de kubernetes de forma que los usuarios, elementos y partes externas pueden comunicarse entre sí; mediante verificación y procesamiento de solicitudes [1], [9].

Un objeto dentro de un clúster de kubernetes se encuentra definido por dos factores, la especificación y el estado. La especificación es la encargada de describir la condición deseada, es decir, la función y configuración recomendable. El estado describe la posición actual del objeto. Kubernetes actualiza ambos elementos para garantizar que el estado actual sea siempre el mismo y coincida con el estado deseado [1], [3]. Entre los objetos básicos se incluyen los siguientes: pod*, service, volumen, namespace.* Además, posee otras abstracciones de un nivel mayor denominadas controladores.

Pod

Es un grupo de uno o varios contenedores donde cada uno de ellos encapsulan una aplicación con almacenamiento y red compartidos, de igual forma especificaciones de cómo ejecutarlos. El contenido de un pod se ubica, programan e implementan en un contexto común, es decir, representa una única instancia de aplicación en kubernetes que se origina en uno o más contenedores de recursos compartidos. Aquellos recursos que necesitan acceder los contenedores existentes son de red y de almacenamiento, a su vez, la red de cada pod se le encuentra asignada una IP interna única y cada contenedor utiliza la misma dirección IP y puerto [1], [3].

Análogamente, si existe más de un contenedor la conexión que se realiza entre ellos es mediante *localhost*, caso contrario si la conexión se la realiza con una entidad externa al pod, son los contenedores que coordinan el uso de recursos de red compartidos [1].

En cuanto al almacenamiento cada pod puede definir su memoria compartida denominada en este caso volumen, la cual puede ser accedida por todos los contenedores que existan y de esta forma compartir los datos [1], [3]. En la siguiente **Figura 3.5** se puede apreciar en el diagrama de un pod con varios contenedores que en su interior está compuesto por un extractor de archivos y a su vez un servidor web que utilizan volúmenes para su almacenamiento y compartir información entre contenedores.

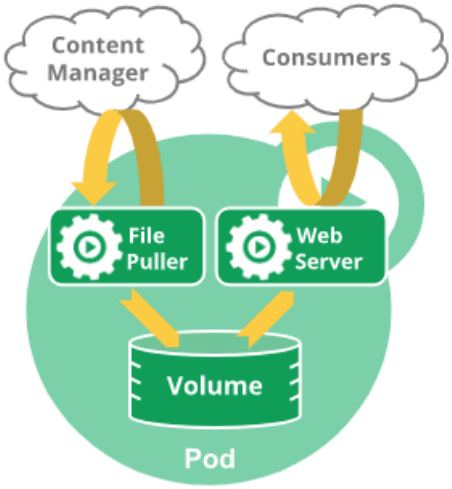

Figura 3.5 Diagrama de un Pod [3]

Volumen

La información que contienen los contenedores de los diferentes pods es temporal, de ahí ha surgido dos problemas importantes, el primero de ellos es ocasionado cuando el organizador (Kubectl) reinicia el contenedor y en él existe pérdida de su contenido provocando que se quede limpio o en el estado inicial. El segundo ocurre cuando dos contenedores se encuentran ejecutándose simultáneamente dentro del mismo pod y se encuentren compartiendo archivos entre ellos. Gracias a la incorporación de kubernetes estos problemas comunes en las aplicaciones se han visto solucionados [1], [3].

En el entorno de kubernetes, un pod puede hacer uso de diversos números de tipos de volúmenes a la vez para almacenar toda la información necesaria, por otro lado, un volumen posee una vida útil idéntica al pod que los contiene. Sin embargo, existen volúmenes que prolongan su existencia más allá del periodo de vida útil de un pod denominados volúmenes persistentes [1], [3]. En cuanto a los tipos de volúmenes más utilizados son:

- EmptyDir. – Se crea cuando un pod apenas es agregado a un nodo y su tiempo de vida útil es el mismo del pod dentro del nodo, hasta que él permanezca activo. Por lo general, este tipo de volumen es un fichero vacío haciendo referencia a su nombre y los contenedores que pertenecen al pod son los encargados de escribir dentro la información necesaria dentro de él [3].
- NFS. – Es un tipo de volumen que permite desplegar un sistema de ficheros de red o por sus siglas en inglés NFS (*Network File System*) en el pod, en comparación de emptyDir, el cual en vez de borrase cuando se remueve el pod simplemente se desmonta. En otras palabras, se puede reutilizar en otros pod la información almacenada e incluso de manera simultánea entre pods [3].

- PersistentVolumeClaim. – Es un tipo de volumen que es utilizado para la creación de volúmenes duraderos y persistentes [3].
- Secret. – Es un tipo de volumen que brinda la facilidad de compartir información sensible, como lo son contraseñas a los pods. Además, se las pueden almacenar en la API de kubernetes y posteriormente desplegarlos de manera de fichero para ser usado con los pods. Generalmente, utilizan memoria volátil por lo que no son escritos en almacenamiento no fugaz [3].

Servicio

En el entorno de kubernetes se define al objeto servicio como una entidad que es utilizada para el enrutamiento de tráfico a los pods correspondientes. Por lo tanto, no es necesario buscar la dirección IP para admitirlos a todos, por añadidura, cada uno de ellos soporta TCP y UDP, además, proporciona una descripción de puertos y balanceadores de carga. En otros casos, los servicios suelen emplear etiquetas y selectores para apuntar el servicio a un conjunto de pods, siendo las etiquetas y selectores capaces de identificar los parámetros más relevantes y significativos para los diversos usuarios [1], [3].

Con kubernetes puedes especificar qué tipo de servicio se quiere implementar, dado que existen diferentes tipos de métodos de brindar dicho servicio, como: ClusterIP, NodePort y LoadBalancer, se lo aprecia en la **Figura 3.6**.

- ClusterIP. – Es el tipo de servicio por defecto, expone el servicio a una dirección IP interna del clúster por lo cual sólo es accesible para los objetos que se encuentran dentro del clúster [3].
- NodePort. – Ofrece el servicio a cada uno de los nodos mediante utilizando la dirección IP del nodo y un puerto fijo, posteriormente se crea automáticamente un servicio de ClusterIP al que es enrutador el servicio NodePort haciendo que el servicio sea accesible desde fuera por medio de la dirección **IP del nodo: puerto estático** [1], [3].
- LoadBalancer. – Ofrece el servicio de manera externa utilizando un balanceador de carga proporcionado por la nube. Los servicios de NodePort y ClusterIP son creados de forma automática para que se prevalezca el camino entre el pod y el exterior [1], [3].

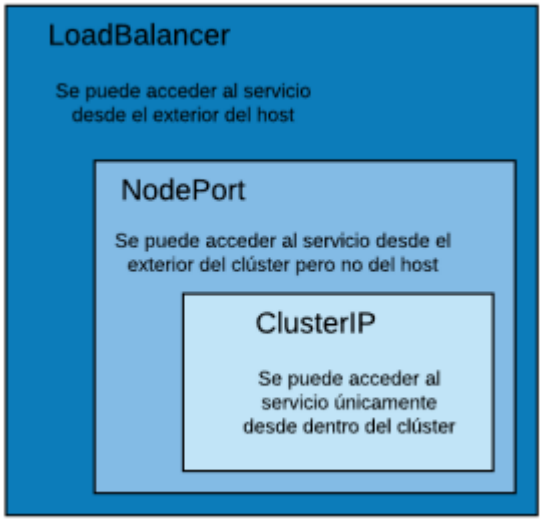

Figura 3.6 Tipos de servicios [3]

Namespace

Espacio de nombres o *namespaces* en inglés, es un modo de distribuir los recursos del clúster entre varios usuarios o *clústeres* virtuales. Además, los objetos creados en un espacio de nombre son transparentes para los usuarios, pero no entre espacios de nombres. Su utilización se enfoca en entornos que poseen un gran número de usuarios repartidos entre varios equipos. Kubernetes al momento de arrancar inicializa con tres espacios de nombres de forma predeterminada, los cuales son: *default*, kube-system y kube-public, sin embargo, tiene la capacidad de añadir otros con los objetos que se deseen [1], [3].

- Default. – Es un espacio de nombres por defecto son usados por los objetos que no especifican ningún espacio de nombres.
- Kube-system. – Es el espacio de nombre que es asignado para los objetos que son creados por kubernetes.
- Kube-public. – Es el espacio de nombres que es creado automáticamente y al cual pueden acceder todos los usuarios autenticados y no autenticados, la utilización se la efectúa cuando se requiere que un recurso pueda ser accesible por el clúster.

Controladores

Replicaset

Un replicaset es utilizado para mantener estable al momento de la ejecución un conjunto de réplicas de pods. Se lo define con varios campos sobre una plantilla en la que se especifica cuántas replicas se crearán o se eliminarán para que coincida el número de

ejecución y el número de réplicas según sea necesario. Además, le permiten seleccionar los pods para administrar y controlar [3], [10].

Deployment

Un despliegue o deployment en inglés, se encarga de proporcionar actualizaciones para un conjunto de replicaset y pods. Un despliegue es la mejor manera de gestionar y crear aplicaciones en kubernetes, por otro lado, kubernetes define las actualizaciones deseadas y el despliegue se encarga de interactuar con los elementos necesarias para efectuar de manera automática y progresiva, por lo general dichos elementos son pods y replicaset [3].

Al utilizar una aplicación con un gran número de pods ejecutándose dentro de un despliegue y uno de los pods se borra, posteriormente se creará de forma automática uno nuevo. Esto es provocado por el trabajo interno que hace el despliegue, ya que, al momento de no coincidir el número de pods con el número deseado de ellos se crea una replicaset para mantener la correlación [1], [3]. En la **Figura 3.7**, se muestra como un despliegue cuenta con los pods ejecutándose y asimismo con las replicasets en caso de que algún pod deje de funcionar.

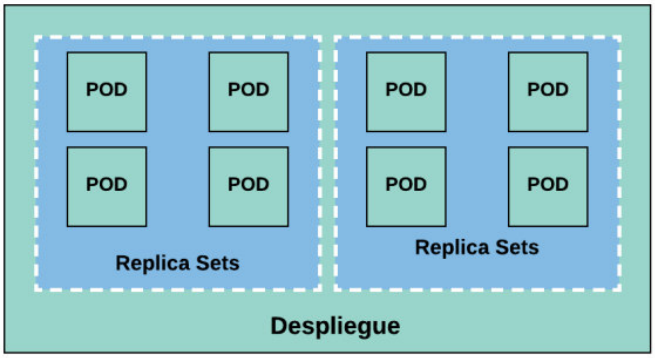

Figura 3.7 Relación entre despliegue, pods y replicasets [1]

Implementación de Kubernetes

Para la instalación de un clúster de kubernetes existen un gran número de caminos para lograr su obtención, los métodos que se mostrarán a continuación son los más simples y rápidos [8].

Jugar con Kubernetes

PWK (*Play With Kubernetes*), es una manera rápida para interactuar con el ambiente de kubernetes, por su forma sencilla de obtener un clúster. Los requisitos que se deben tener para su despliegue es una computadora y el acceso a internet, además, de contar con una cuenta en docker Hub o GitHub [8].

Cabe mencionar que existen limitaciones al utilizar PWK, se debe considerar que posee un límite de tiempo para su uso siendo aproximadamente de 4 horas, no tiene integraciones con servicios externos como balanceadores de carga y volúmenes basados en la nube [8].

Configuración de PWK

1. Colocar en el navegador https://https://labs.play-with-k8s.com/
2. Iniciar con su cuenta de docker hub o github.
3. Hacer clic en la sección de agregar nueva instancia que se ubica en el lado izquierdo del navegador, posteriormente se cargará un nuevo elemento, pues este es un nodo de kubernetes.
4. Ejecutar algunos comandos para visualizar los componentes preinstalados en el nodo, con los comandos de la **Figura 3.8** se visualizarán los elementos preinstalados.

```
$ docker version
Docker version 19.03.11-ce...

$ kubectl version --output=yaml
clientVersion:
...
  major: "1"
  minor: "18"
```

Figura 3.8 Comando para PWK [8]

5. Para iniciar el nuevo clúster se debe ejecutar el comando init proporcionado por kubeadm cuando se agrega la nueva instancia que consta en el paso 3. PWK proporciona una serie de comandos para correr el clúster y también el nodo, esto puede visualizarse en **Figura 3.9**.

```
$ kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr...
[kubeadm] WARNING: kubeadm is in beta, do not use it for prod...
[init] Using Kubernetes version: v1.18.8
[init] Using Authorization modes: [Node RBAC]
<Snip>
Your Kubernetes master has initialized successfully!
<Snip>
```

Figura 3.9 Inicialización del clúster [8]

6. Después, se debe verificar el funcionamiento del clúster con el siguiente comando como se muestra en la **Figura 3.10** obteniendo como resultado el clúster de un solo nodo, sin embargo, el estado de nodo está en *Notready* esto se debe a que aún no se ha configurado la red pod.

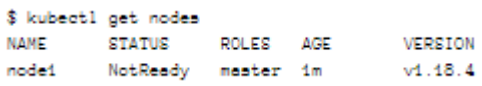

Figura 3.10 Verificación del nodo [8]

7. A continuación, se debe iniciar la red pod mediante el segundo comando proporcionado por PWK que se representa en la **Figura 3.11** en la línea de la *shell*.

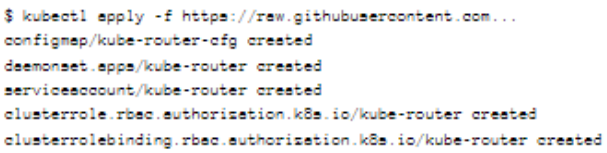

Figura 3.11 Inicialización de la red Pod [8]

8. Por último, se vuelve a comprobar el clúster para ver el cambio de estado del nodo consiguiendo un estado *Ready*, se lo evidencia en la **Figura 3.12**.

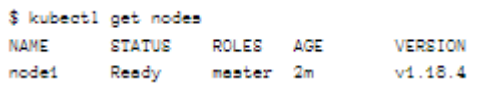

Figura 3.12 Verificación del funcionamiento [8]

Google Kubernetes Engine (GKE)

Corresponde a un servicio derivado de kubernetes alojado en la *Google Cloud Plataform* (GCP), de igual manera, esta extensión de kubernetes proporciona servicios como cualquier otro [8].

Entre los servicios que esta plataforma ofrece se puede mencionar que posee una manera rápida y fácil para la obtención de un clúster de kubernetes a nivel de producción, además, cuenta con un plan de control administrado y por último un sistema de facturación detallada. Sin embargo, GKE y otros servicios no son gratuitos y toca pagar por usarlos [8].

Configuración de GKE

1. Desde el apartado de consola de su proyecto de GCP, abrir el panel de navegación ubicado en el lado izquierdo y posteriormente seleccionar kubernetes Engine > clústeres.
2. Luego, hacer clic en el botón crear clúster lo que provocará la inicialización de asistente para la respectiva creación de un nuevo clúster de kubernetes.
3. A continuación, se asignará un nombre y una descripción al clúster.

4. Luego, debe elegir entre si desea un clúster regional o uno zonal. El clúster regional es más nuevo y potente, dado que, sus maestros y nodos se distribuyen en varias zonas.
5. Elegir la región o zona para su clúster.
6. Después, se seleccionará la versión master, la cuál será la versión que llevarán los maestros y nodos. Estas versiones disponibles se encuentran limitadas.
7. En este punto del despliegue se va a especificar unas opciones avanzadas situadas en el panel izquierdo. Estas opciones pueden incluir si se desea que los nodos se ejecuten en docker o simplemente containerrd, además, si habitarán la red de servicios o no. Para evitar configurar cada una de ellas se las puede dejar como predeterminadas.
8. Una vez configurado los parámetros con sus opciones, se da clic en crear.
9. En este paso se visualizará el clúster y la página del mismo con la descripción colocada. Se puede apreciar en la **Figura 3.13** se muestra el clúster creado con 3 nodos llamados GKE.

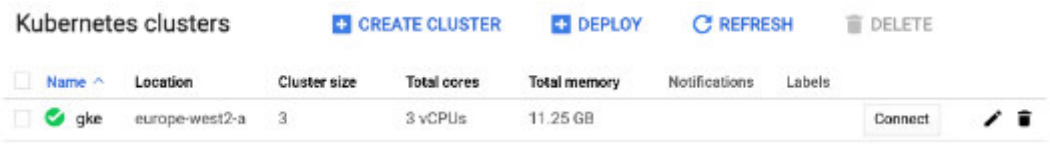

Figura 3.13 Verificación del funcionamiento del clúster con GKE [8]

Minikube

Minikube a diferencia de PWK y GKE es una herramienta que permite desplegar un clúster de kubernetes en una máquina local. Además, permite ejecutar un único nodo que cumple la función de maestro y nodo a la vez [11].

Para la instalación se siguen las instrucciones desde su documentación oficial colocada en su página web https://minikube.sigs.k8s.io/docs/start/, en donde nos detalla ciertos requisitos que se deben tener en cuenta.

Instalación de Minikube:

1. Seleccionar el tipo de sistema donde se realizará la instalación, en este caso se seleccionará un sistema libre Linux, donde se insertarán los comandos que obtendrán después de la elección del sistema operativo evidenciado en la **Figura 3.14**.

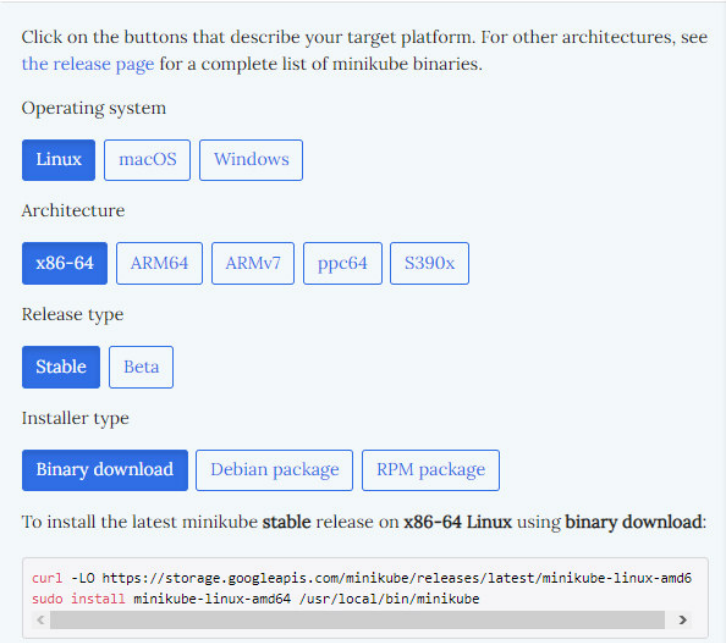

Figura 3.14 Página oficial para la instalación de minikube

2. Luego, desde la terminal de Linux se insertará el siguiente comando que se puede apreciar en la **Figura 3.14** que dará inicio o instalará los componentes necesarios para su ejecución mediante el comando que se observa en la **Figura 3.15.**

```
minikube start
```

Figura 3.15 Comando para inicialización de minikube [11]

3. Posteriormente se instalará la interfaz de minikube y se procederá a probar desde la CLI de comandos con comandos para ver su funcionamiento, se puede evidenciar en la **Figura 3.16**, **Figura 3.17**, **Figura 3.18** y **Figura 3.19** algunos comandos para dicha prueba de funcionamiento.

```
kubectl get po -A
```

Figura 3.16 Comando de kubernetes [11]

```
minikube kubectl -- get po -A
```

Figura 3.17 Comando de minikube [11]

```
minikube pause
```

Figura 3.18 Comando de minikube [11]

```
minikube addons list
```

Figura 3.19 Comando de minikube [11]

4. Su clúster de kubernetes se encuentra listo para ser utilizado se lo evidencia en la **Figura 3.20**.

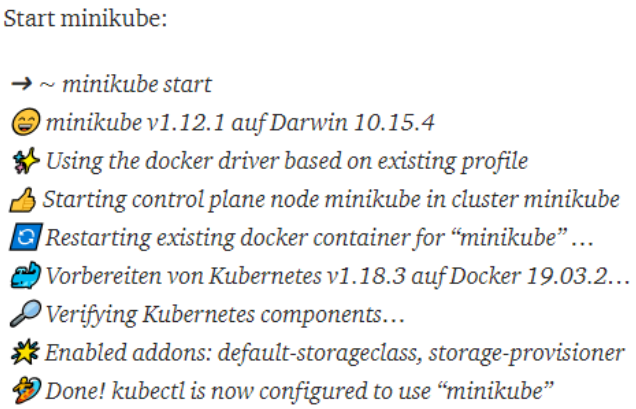

Figura 3.20 Minikube ejecutándose

Comparación de Ansible y Docker con Kubernetes

Tabla 1 Comparación entre herramientas de DevOps y Kubernetes

Características	Ansible	Docker	Kubernetes
Lenguaje de programación	Python, PowerShell, Ruby	GO	GO
Flexibilidad	Orquestación a nivel de aplicación sin importar el entorno.	Utilización de imágenes gratuitas y portables	Utilización portátil y en múltiples entornos
Costos	Plataforma gratuita de código abierto.	Plataforma gratuita de código abierto.	Plataforma gratuita de código abierto.
Proceso de automatización	Utiliza playbooks en formato YAML para la automatización de software, sistemas y funciones de TI.	No posee automatización, pero consta de todas las herramientas de desarrollador.	Diseñada para la orquestación de contenedores.
Mantenimiento	Mantenimiento fácil debido a su arquitectura simple.	Mantenimiento complicado, debido a que su	sencillo debido a la capacidad de

		arquitectura es propensa a ataques de virus.	auto reparación.
Interfaz de Usuario	No se encuentra muy bien desarrollada.	Amigable con el usuario.	Amigable con el usuario.

Servidor Web

Es un programa informático que forma parte de un servidor, diseñado para la gestión de aplicaciones del lado del servidor y transferencia de información de hipertexto (páginas *web*) cuando recibe peticiones por parte de los usuarios. Las peticiones de los usuarios a menudo utilizan el protocolo HTTP (*Hypertext Transfer Protocol*) o HTTPS (*Hypertext Transfer Protocol Secure*), luego son recopiladas y ejecutadas por un navegador *web* como Chrome, Firefox, entre otros, los cuales reciben una respuesta por parte del servidor web en forma de un fichero HTML o PHP [12], [13].

Servidores Web

Servidor HTTP Apache

Es un servidor de código abierto y gratuito desarrollado por *Apache Software Fundation,* es uno de los servidores más utilizados en el mundo, debido a que, soporta diferentes lenguajes de desarrollo web como PHP, Python y Perl. También, se incluyen características como conexiones seguras y la compresión de datos [12], [13].

Es uno de los servidores más antiguos y confiables, esto se debe por ser un servidor multiplataforma soportando sistemas operativos como Linux, Windows y entre otros.

Servidor Nginx

De igual forma que el servidor apache es un servidor de código abierto y compatible en muchas plataformas como: Windows, Linux, IOS y entre otros. Además, es conocido por ser ligero y con una gran eficiencia, por esa razón es utilizado como un proxy inverso y un balanceador de carga [12], [13].

Servidor Microsoft IIS

Es un servidor web desarrollado por Microsoft, el cual corresponde a un conjunto de servicios en un sistema Microsoft Windows y se ejecuta sobre el mismo gracias a su tecnología IIS (*Internet Information Services*). Una de las ventajas de su uso es la capacidad de atención de peticiones de los usuarios que las realiza de manera

concurrente y mejoras importantes en la entrada y salida asincrónica mejorando la eficiencia de la aplicación [12].

Virtual Hosting

Es una herramienta que permite que múltiples sitios webs puedan trabajar sobre una IP compartida, es decir, es capaz de publicar más de una página web en un mismo servidor web. Los recursos que son asociados a un *host* virtual no se pueden compartir con los recursos de otro virtual host, en pocas palabras, los *hosts* que se encuentren en un mismo servidor su gestión y ejecución de cada sitio web se la realiza de forma independiente [14], [15].

En una configuración de *hosts* virtuales cada uno de ellos posee su propio nombre lógico y uno o varios nombres de dominio para interconectarse, además, el número de puerto TCP/IP que se utiliza para solicitar el servicio [15].

Implementación de Virtual Host

Para la configuración de *hosts* virtuales, se lo puso a prueba mediante una máquina virtual con un sistema operativo Ubuntu 22.04 y levantado el servicio de apache 2.

Procedimiento:

1. Se crea los directorios que tendrá alojamiento las páginas webs donde se contendrán el contenido de los diferentes sitios web que luego utilizaremos como se lo aprecia en la **Figura 3.21**.

```
root@jeff:/home/jeff# mkdir /var/www/jefferson1
root@jeff:/home/jeff# mkdir -p /var/www/jefferson1
root@jeff:/home/jeff# mkdir -p /var/www/jefferson2
```

Figura 3.21 Instrucciones para la creación de archivos de alojamiento

2. Se crea el archivo de configuración de los nuevos *hosts* virtuales, para la creación del archivo se coloca dentro del directorio /etc/apache2/sites-available y se procede a copiar el archivo cp 000-default.conf dentro de otro archivo denominado jefferson1.conf y jefferson2.conf respectivamente, dicho proceso se lo evidencia en la **Figura 3.22**.

```
root@jeff:/var/www# cd /etc/apache2/sites-available/
root@jeff:/etc/apache2/sites-available# cp 000-default.conf jefferson1.conf
root@jeff:/etc/apache2/sites-available# nano jefferson1.conf
root@jeff:/etc/apache2/sites-available# cp 000-default.conf jefferson2.conf
root@jeff:/etc/apache2/sites-available# nano jefferson2.conf
```

Figura 3.22 Archivo de configuración del nuevo host

3. Posteriormente, se realiza el proceso de edición de los archivos jefferson1.conf y jefferson2.conf mediante el editor de texto “nano” que se presenta en la **Figura**

3.23, para el host virtual número 2 se lo puede apreciar en la **Figura 3.24**, en donde se insertó el nombre de dominio y la ruta del directorio creado.

Figura 3.23 Archivo de configuración jefferson1.conf

Figura 3.24 Archivo de configuración jefferson2.conf

4. Luego, se procede a habilitar los nuevos *hosts* virtuales, el comando para estos procesos se los puede visualizar en la **Figura 3.25** y **Figura 3.26** para cada uno de los sitios.

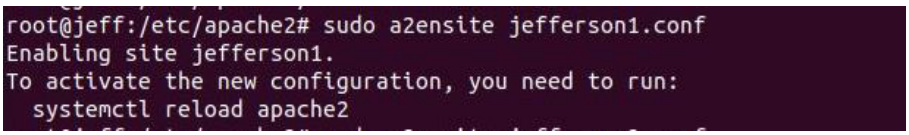

Figura 3.25 Habilitación de archivos de configuración jefferson1.conf

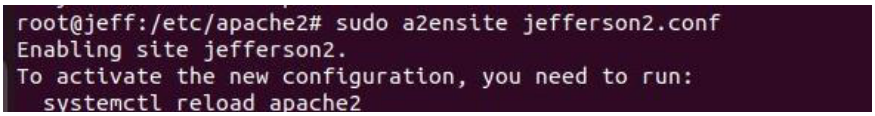

Figura 3.26 Habilitación de archivos de configuración jefferson2.conf

5. Posteriormente, se realizó la creación de un fichero index.html para cada sitio web a implementar, lo cual se realizó mediante el editor de texto “nano” desde el directorio de cada carpeta de dicho sitio creado, se lo evidencia dicho proceso en la **Figura 3.27**.

```
root@jeff:/etc/apache2# cd /var/www/
root@jeff:/var/www# cd jefferson1
root@jeff:/var/www/jefferson1# nano index.html
root@jeff:/var/www/jefferson1# cd /var/www/
root@jeff:/var/www# jefferson2
jefferson2: orden no encontrada
root@jeff:/var/www# cd jefferson2
root@jeff:/var/www/jefferson2# nano index.html
root@jeff:/var/www/jefferson2#
```

Figura 3.27 Creación de ficheros index.html jefferson1 y jefferson2

6. En la **Figura 3.28** y **Figura 3.29** se presenta el contenido de los archivos index.html para cada uno de los sitios webs que van a poseer al momento de acceder a cada nombre de dominio.

Figura 3.28 Contenido del fichero index.html jefferson1

Figura 3.29 Contenido del fichero index.html jefferson2

7. Luego, para tener los dominios en local y poder probar las páginas webs se procede a editar el archivo /etc/hosts/ con el editor de texto “nano” donde se agrega la dirección IP de la máquina y los nombres de dominios colocados en pasos anteriores consiguiendo lo que se presenta en la **Figura 3.30.**

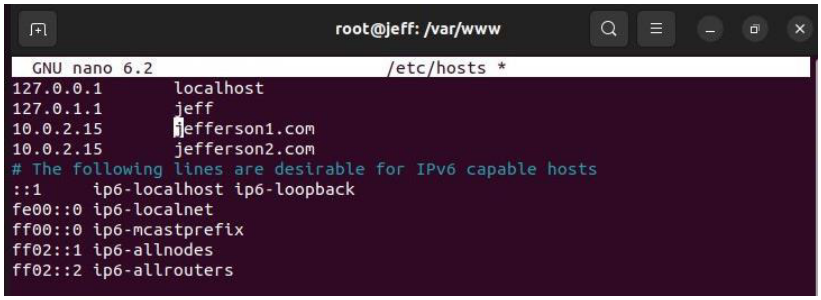

Figura 3.30 Asignación de los nombres de dominio para ser accedidos

8. Finalmente, configurado los aspectos necesarios para cada uno de los sitios webs se procede a recargar el servicio de apache 2 mediante el comando systemctl reload apache2, para posteriormente probar los sitios creados como se presenta en la **Figura 3.31** y **Figura 3.32.**

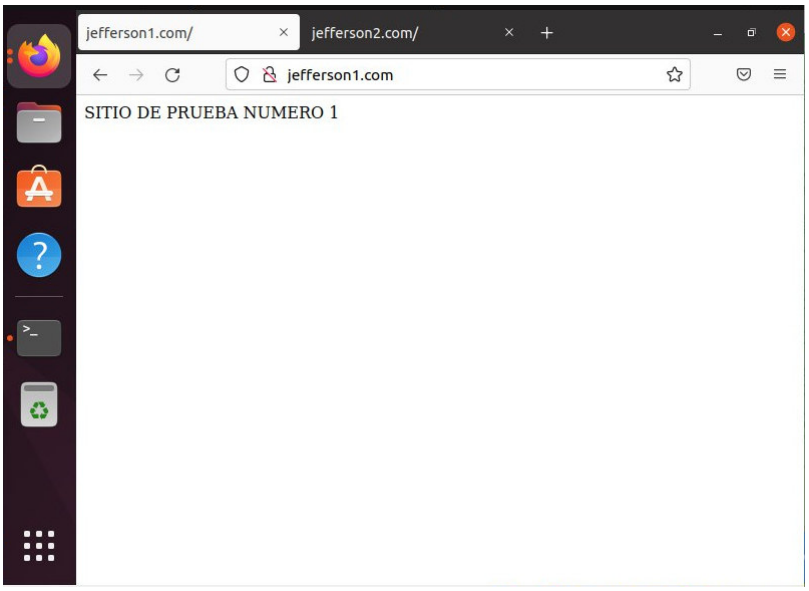

Figura 3.31 Ejecución de sitio web jefferson1.com

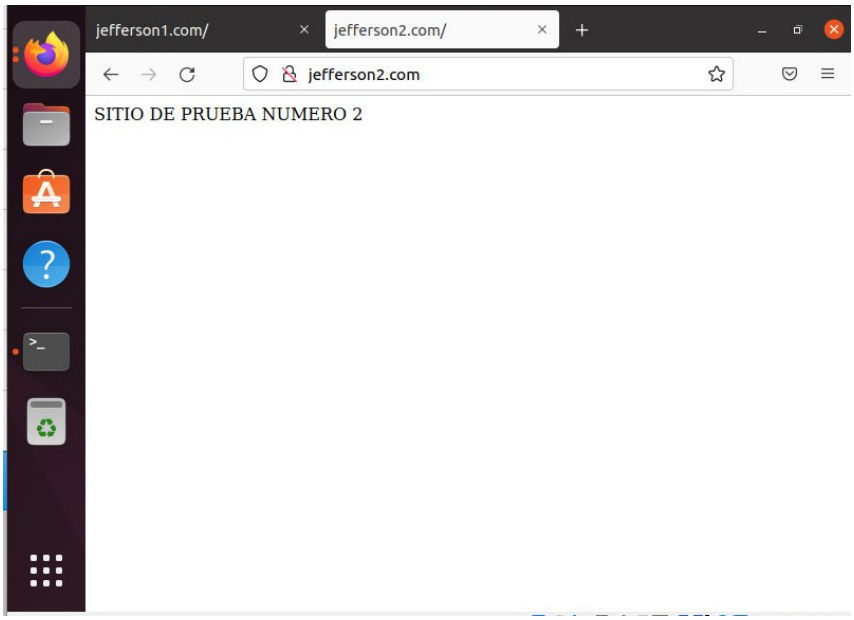

Figura 3.32 Ejecución de sitio web jefferson2.com

3.2 Diseñar una aplicación en Kubernetes que permita la implementación de servidores web virtuales.

La manera más sencilla de empezar a manipular un clúster de kubernetes es por medio de minikube. Antes de instalar la herramienta minikube se debe tener en cuenta los requisitos con los que debe contar el sistema para su despliegue.

Requisitos:

- 2 o más procesadores

- 2 GB o más de memoria RAM
- 20 GB o más de almacenamiento libre
- Conexión a internet
- Administrador de contenedores o máquinas virtuales (Docker, Hyper V, VirtualBox, Podman)

En la **Figura 3.33** se puede comprobar que se cumplen los 4 requisitos previos, excluyendo al último de ellos para el funcionamiento de minikube, esto se lo realiza por medio de una virtualización de una máquina con un sistema operativo Ubuntu 20.04 y en la que se asignan los parámetros de almacenamiento, procesadores y capacidad de la memoria RAM requeridos para su óptimo funcionamiento.

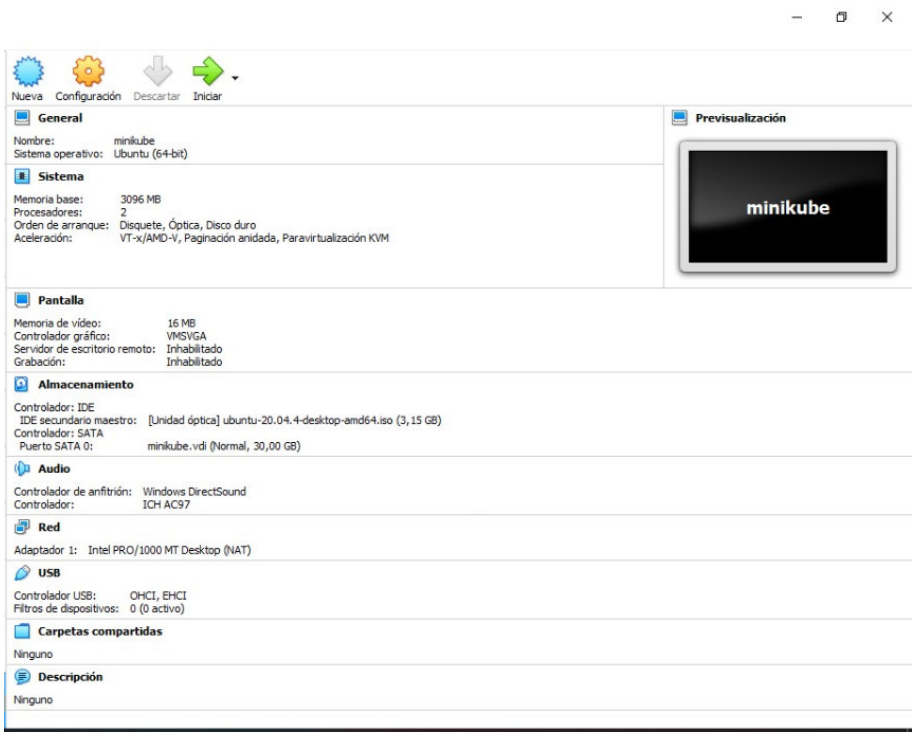

Figura 3.33 Configuración de la máquina virtual

Para cumplir con el último requisito del despliegue de minikube se ejecuta el comando en modo superusuario o *root* para la instalación de docker, en la **Figura 3.34** se muestra la ejecución del comando para la instalación de docker mediante la *shell*.

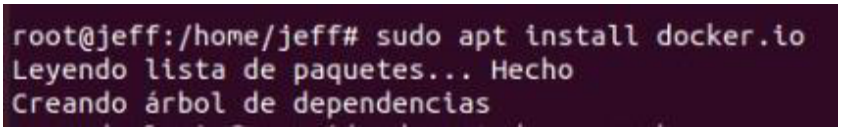

Figura 3.34 Instalación de Docker

Una vez que se tiene una máquina virtual que cumple con los requisitos para desplegar la herramienta minikube, se sitúa dentro de la documentación oficial que proporciona la plataforma de minikube en su página web, donde se selecciona el tipo de sistema

operativo. Además, la arquitectura y el tipo de instalador para que de esta manera facilite los respectivos comandos para la descarga e instalación de dicha herramienta, dicho proceso se lo puede observar en la **Figura 3.35** donde se nos proporciona los comandos requeridos para su despliegue.

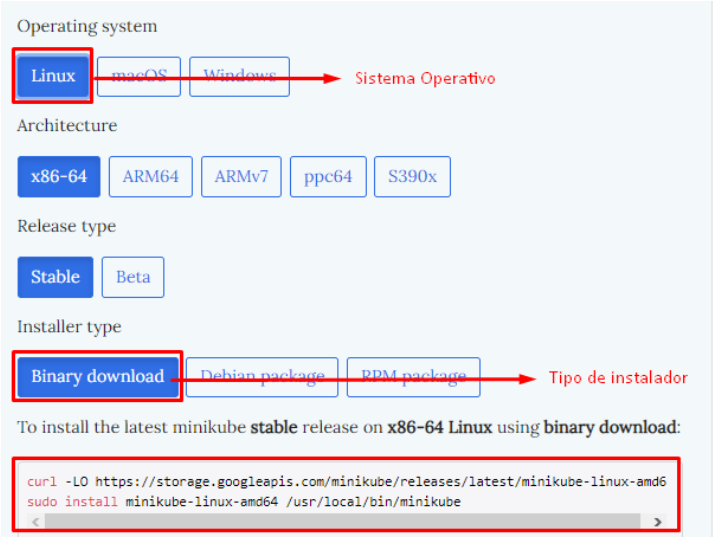

Figura 3.35 Página oficial para obtención del instalador binario de minikube

Una vez colocado y ejecutado los comandos de la ruta de descarga e instalación dentro de la terminal de comandos de Ubuntu se obtendrán como resultado algo similar a la **Figura 3.36**, donde se aprecia la descarga de la herramienta de minikube y posteriormente su instalación.

```
root@jeff:/home/jeff# curl -LO https://storage.googleapis.com/minikube/releases
/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 71.4M  100 71.4M    0     0  3877k      0  0:00:18  0:00:18 --:--:-- 5856k
root@jeff:/home/jeff# sudo install minikube-linux-amd64 /usr/local/bin/minikube
root@jeff:/home/jeff#
```

Figura 3.36 Descarga del instalador binario de minikube

Desde la terminal de comandos del sistema operativo Ubuntu se coloca el comando que se aprecia en la **Figura 3.37** para iniciar el clúster de kubernetes, cabe destacar que se está ejecutando dentro de una máquina virtual en donde se debe especificar el tipo de ambiente que se utilizará, el tipo de ambiente en este caso será docker.

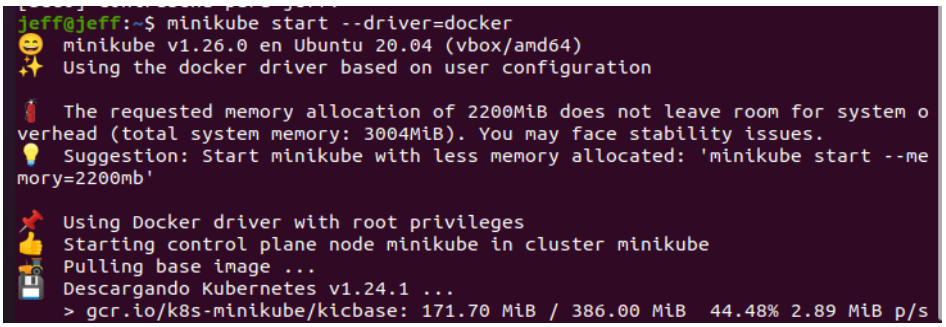

Figura 3.37 Minikube Ejecutándose

3.3 Implementar el diseño de kubernetes de servidores web virtuales.

Una vez conocidos todos los detalles referentes a kubernetes como los objetos, controladores y de igual manera el despliegue de su clúster como se lo aprecia en la **Figura 3.37**. Es momento de pasar a la siguiente fase que hace énfasis al despliegue de los servidores web virtuales para la implementación de *hosts* virtuales.

Con todo lo aprendido anteriormente, se inicia el clúster de kubernetes con la ayuda de la herramienta minikube, mediante el comando presentado en la **Figura 3.38** para poder trabajar en un ambiente tipo docker cómo se lo aprecia en la figura antes mencionada.

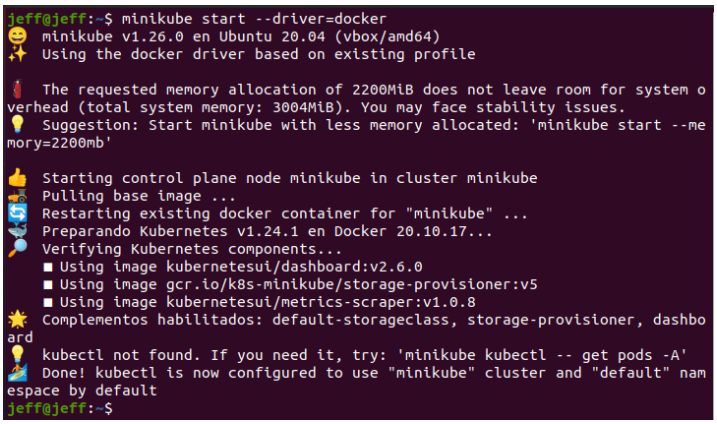

Figura 3.38 Inicialización de clúster de kubernetes-minikube

Con el clúster de kubernetes levantado y ejecutándose, se tiene que colocar el comando que se presenta en la **Figura 3.39**, el cual se lo utiliza para la interacción con el clúster y la verificación del estado de los servicios, pods y despliegues que en él existan mediante minikube. La visualización y verificación de esta herramienta se realiza mediante una página web, tal cual se muestra en la **Figura 3.40**.

```
jeff@jeff:~$ minikube dashboard
🤔  Verifying dashboard health ...
🚀  Launching proxy ...
🤔  Verifying proxy health ...
🎉  Opening http://127.0.0.1:37619/api/v1/namespaces/kubernetes-dashboard/servi
ces/http:kubernetes-dashboard:/proxy/ in your default browser...
```

Figura 3.39 Comando para la cooperación entre el clúster y sus componentes

La **Figura 3.40** corresponde al plano de control de minikube en el cual se aprecia que dentro del clúster no se encuentra ejecutándose ningún servicio, ni mucho menos un pod o un despliegue.

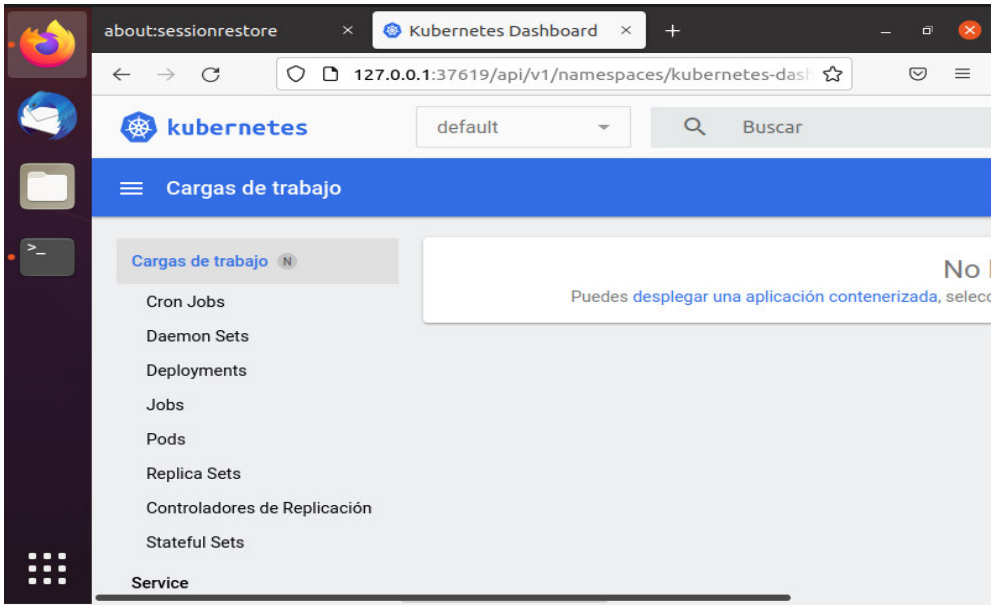

Figura 3.40 Plano de control de minikube

Para levantar un servicio, un pod o un despliegue, es necesario utilizar un archivo de tipo YML, el cual es utilizado para la creación y configuración de los objetos de minikube y se lo crea con editores de archivos convencionales como lo son “vim” o “nano” con un lenguaje de programación YAML.

Para este caso se opta por utilizar el editor “nano”, después, se procede a crear un fichero YML (app1) para levantar el primer servicio web y a su vez dentro de dicho archivo se incluirá del mismo modo los objetos del despliegue y el pod respectivamente. En la **Figura 3.41** se puede apreciar la creación del archivo YML.

```
jeff@jeff:~$ nano app1.yml
```

Figura 3.41 Comando para la creación del archivo del servicio web

En la **Figura 3.42** se detalla las instrucciones que permite el levantamiento del servicio uno (app1) y a su vez el respectivo pod y despliegue utilizando una imagen de google

para el servidor web dentro del pod, además, se coloca la estructura y sintaxis que deberá tener dicho archivo en un lenguaje de programación YAML.

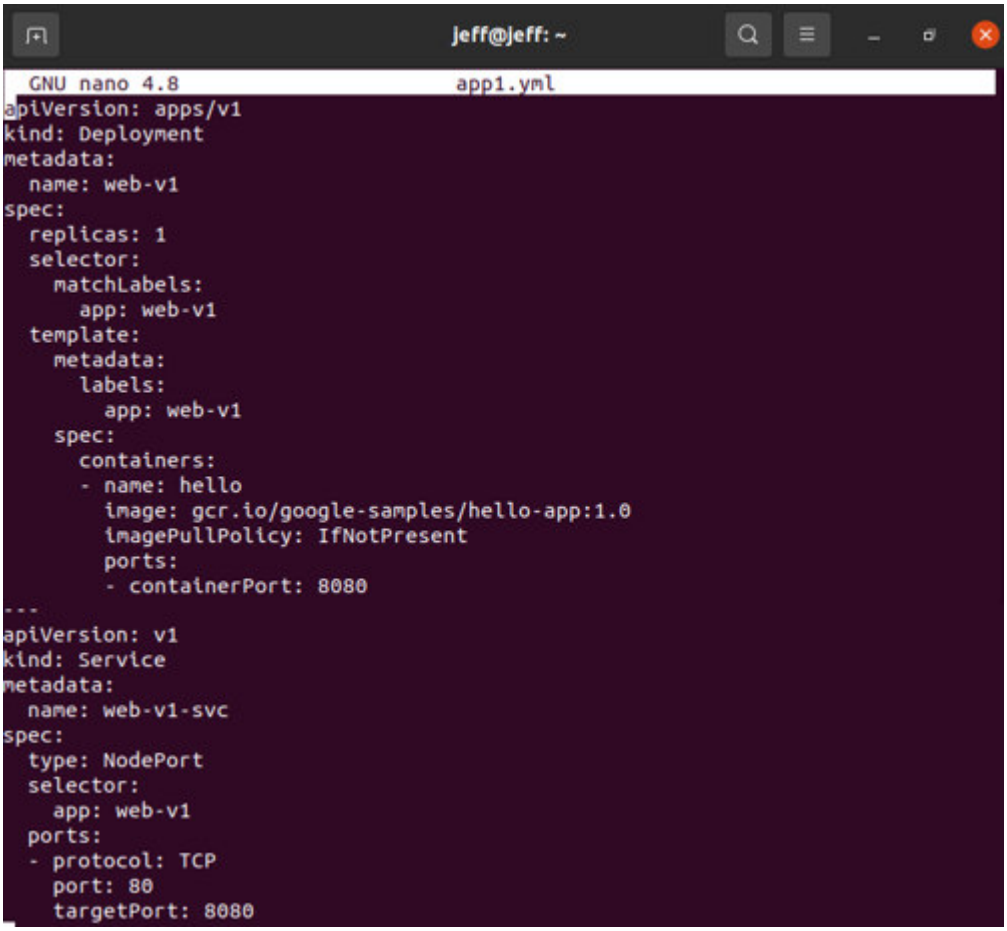

Figura 3.42 Instrucciones para desplegar la aplicación web 1

Para el servicio web número 2 se procede a hacerlo de forma similar al primero, se crea el fichero de tipo YML (app2), posteriormente se ingresa al fichero para la elaboración de las instrucciones del levantamiento del servicio, y a su vez se configura el objeto despliegue y el pod del mismo. Este procedimiento se evidencia en la **Figura 3.43** y **Figura 3.44**.

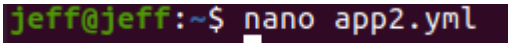

Figura 3.43 Creación del fichero YML

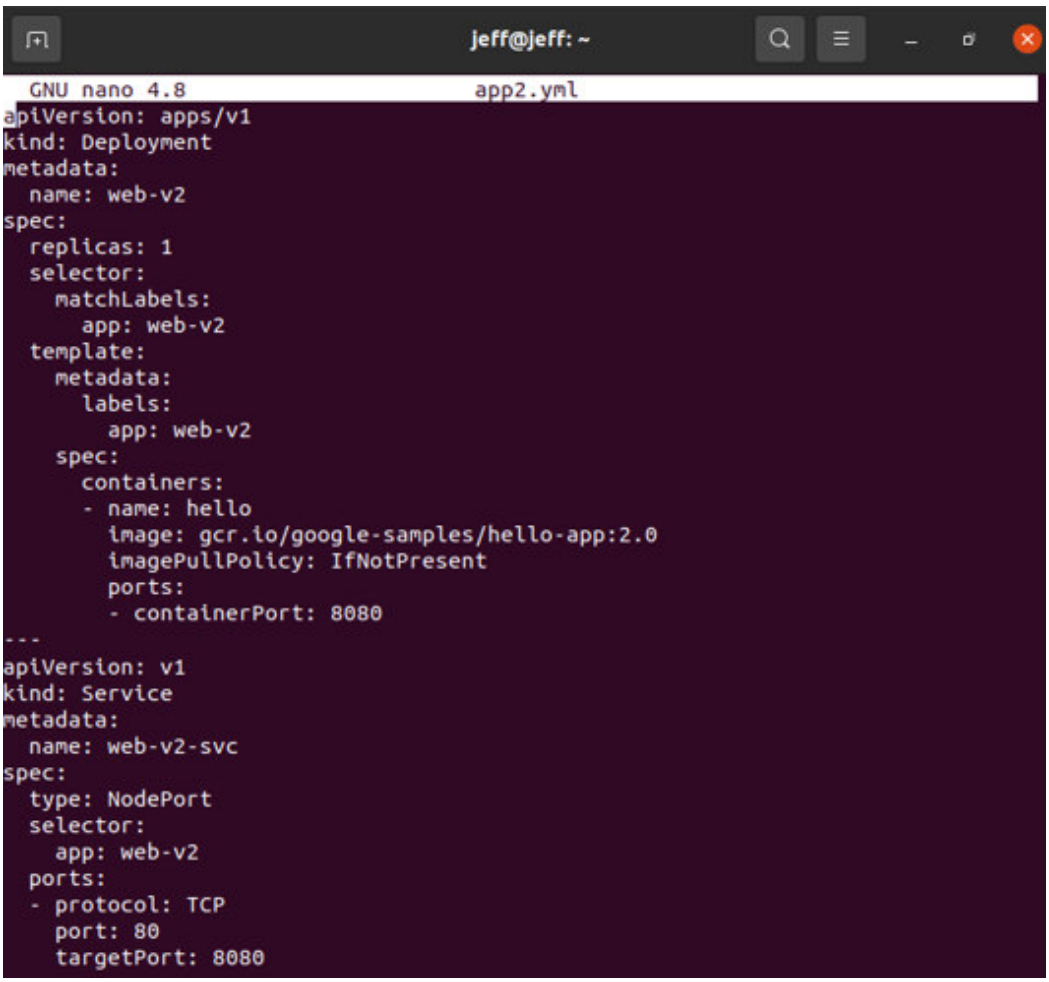

Figura 3.44 Instrucciones para desplegar la aplicación web 2

Una vez creado los manifiestos de las aplicaciones web1 y web2 se requiere una forma para poder direccionar dichos elementos a los pods correspondientes, por esa razón se necesita una herramienta extra denominada *Ingress*, la cual permite un correcto direccionamiento del servicio.

Al existir varios controladores de tipo de ingreso, se selecciona el más empleado en la industria siendo *Nginx Ingress Controller*, debido a su fácil inclusión en la herramienta minikube, en la **Figura 3.45** se evidencia la implementación de este tipo de enrutador.

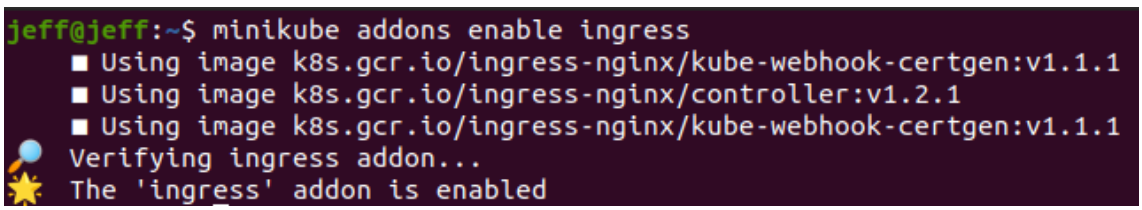

Figura 3.45 Implementación del controlador del ingreso nginx

Para verificar la instalación del controlador de ingreso nginx, se lo realiza mediante comando de la **Figura 3.46**, donde se comprueba el respectivo funcionamiento.

```
jeff@jeff:~$ kubectl get pods -n ingress-nginx
NAME                                        READY   STATUS      RESTARTS      A
GE
ingress-nginx-admission-create-5rv6p        0/1     Completed   0             3
8m
ingress-nginx-admission-patch-xdvr8         0/1     Completed   1             3
8m
ingress-nginx-controller-755dfbfc65-nhpgr   1/1     Running     2 (22m ago)   3
8m
```

Figura 3.46 Comprobación del Controlador de ingreso nginx

Para la configuración del controlador de ingreso de nginx, se lo realiza mediante la creación de un fichero de tipo YML, al igual que, los manifiestos para levantamiento de los servicios web, en el cual se especifica los nombres de dominio que se van a direccionar con las peticiones de los *hosts* virtuales y el servicio respectivo, el procedimiento se lo evidencia en la **Figura 3.47**.

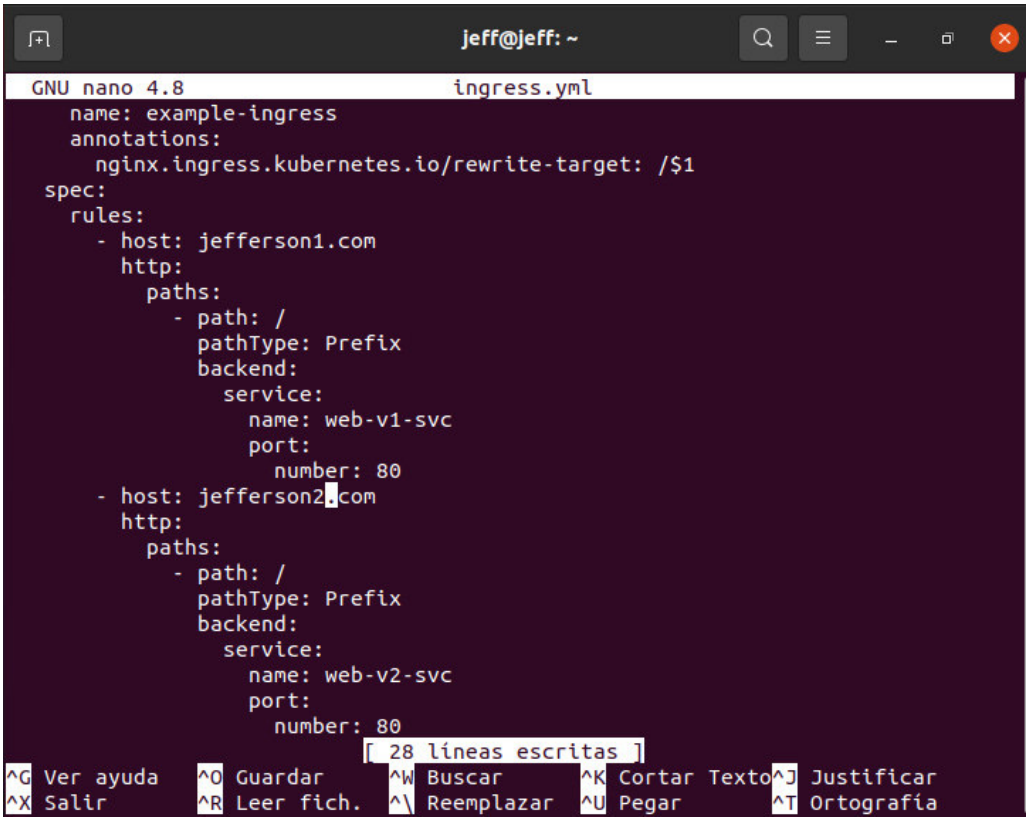

Figura 3.47 Configuración del controlador del ingreso nginx

En la **Figura 3.48**, se puede visualizar los ficheros que fueron creados para el levantamiento y creación de los servicios web de los diferentes *hosts* virtuales. La ejecución de estos manifiestos se lo realiza mediante la *shell* con ayuda del comando "kubectl apply –f < nombre del fichero >", el cual creará un pod y desplegará el servicio o despliegue correspondientemente.

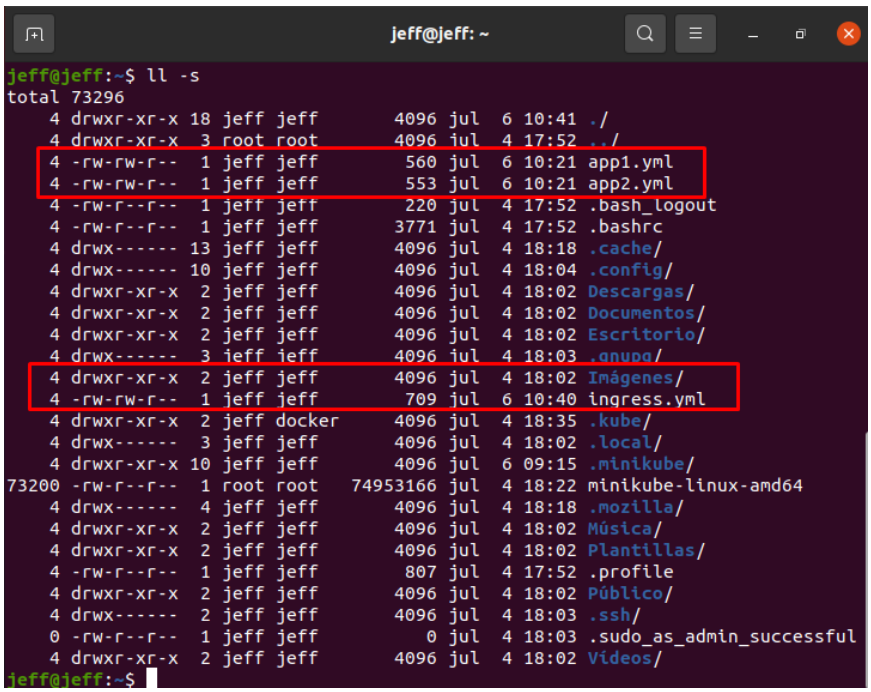

Figura 3.48 Verificación de ficheros YML creados

A continuación, mediante la **Figura 3.49** se observa la ejecución del comando que permite comprobar el correcto funcionamiento de los servicios levantados, además, características adicionales de cada uno de ellos entre las cuales se aprecia: el tiempo de creación, el puerto estático y el puerto que brinda el servicio, adicionalmente, la dirección IP.

```
jeff@jeff:~$ kubectl get services
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP        40h
web-v1-svc   NodePort    10.107.187.238   <none>        80:31816/TCP   4m37s
web-v2-svc   NodePort    10.103.38.181    <none>        80:30278/TCP   4m26s
jeff@jeff:~$ kubectl get deploy
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
web-v1   1/1     1            1           5m6s
web-v2   1/1     1            1           4m56s
```

Figura 3.49 Verificación de los servicios ejecutándose

Luego de la ejecución de los ficheros para los servicios web, se realiza el mismo procedimiento para el fichero del controlador de ingreso nginx (ingress), con el fin de conseguir un direccionamiento del servicio a los respectivos pods. Este proceso se evidencia en la **Figura 3.50** la cual anexa la ejecución del manifiesto de controlador de ingreso.

```
jeff@jeff:~$ kubectl apply -f ingress.yml
ingress.networking.k8s.io/example-ingress created
```

Figura 3.50 ejecución del fichero de ingreso del controlador nginx

En la **Figura 3.51** se puede observar el controlador de ingreso nginx ejecutándose, en donde corrobora la correcta configuración del controlador nginx, además, se presenta la dirección IP, los *hosts* virtuales y puerto que se va a dar el servicio.

```
jeff@jeff:~$ kubectl get ingress
NAME              CLASS   HOSTS                          ADDRESS        PORTS
  AGE
example-ingress   nginx   jefferson1.com,jefferson2.com   192.168.49.2   80
  77s
```

Figura 3.51 Verificación de funcionamiento del controlador de ingreso nginx

A continuación, para el direccionamiento de los *hosts* virtuales se procede a modificar el fichero con ayuda del editor de texto “nano” que se sitúa en el directorio /etc/hosts/ el cual se accede desde una nueva terminal. La **Figura 3.52** evidencia la inclusión de los *hosts* virtuales en el fichero /etc/hosts/ donde se agrega la dirección IP que proporcionó el controlador de ingreso instalado, además, de los nombres del dominio para acceder a cada servicio.

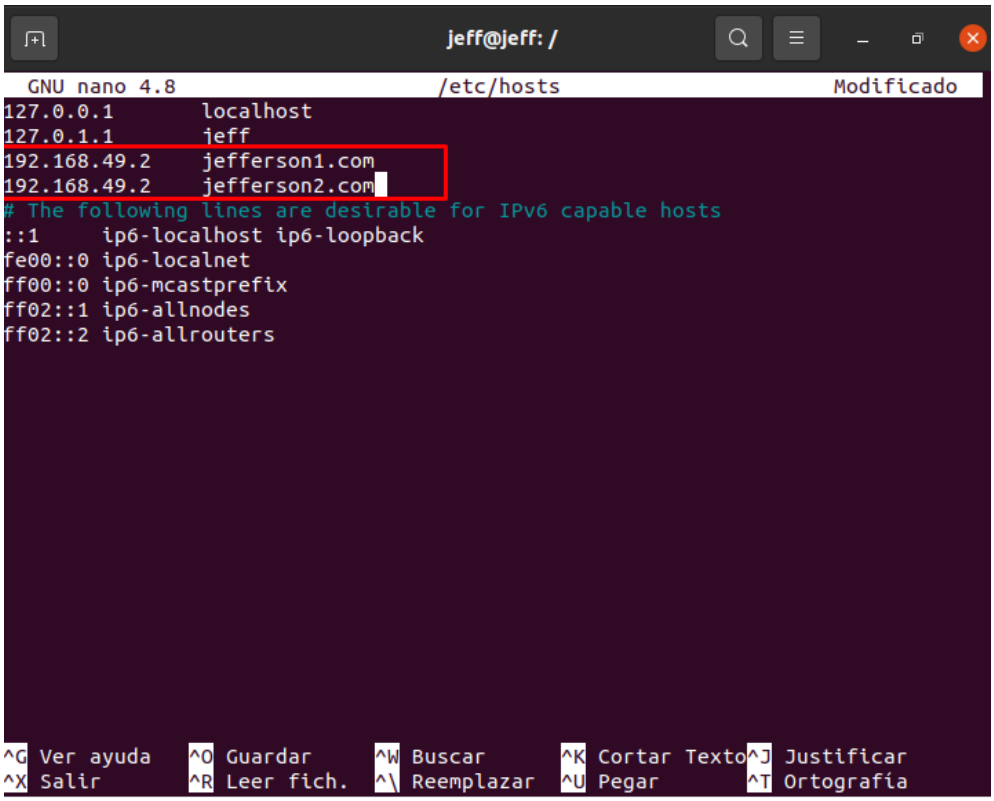

Figura 3.52 Agregación de los *hosts* virtuales en el fichero etc/hosts

Finalmente, se puede acceder al servicio proporcionado para cada *host* virtual por medio de la *shell* con ayuda del comando CURL presentado en la **Figura 3.53**, aquí se visualiza el contenido de cada servicio web al acceder al nombre de dominio.

```
jeff@jeff:~$ curl jefferson1.com
Hello, world!
Version: 1.0.0
Hostname: web-v1-6986bdbc47-qmfkh
jeff@jeff:~$ curl jefferson2.com
Hello, world!
Version: 2.0.0
Hostname: web-v2-5c5d5d4c77-npb9r
```

Figura 3.53 Contenido de cada servicio en su respectivo DNS

3.4 Verificar el funcionamiento de la aplicación desarrollada en Kubernetes.

Para la ejecución de las pruebas del funcionamiento en el clúster de kubernetes se lo realiza mediante el acceso al servicio con el nombre del dominio asignado para cada uno de los *hosts* virtuales, para cualquiera de los casos será jefferson1.com o jefferson2.com según sea el caso, se evidencia en la **Figura 3.53**.

Antes de acceder al servidor web para verificar el funcionamiento del mismo, se debe visualizar las cargas de trabajo para verificar que ningún servicio se encuentre fallando como se presenta en la **Figura 3.54** donde consta las cargas de trabajo de cada servicio desplegado. También, la **Figura 3.55** hace énfasis a los servicios en un plano minimalista los cuales fueron levantados en los pasos anteriores.

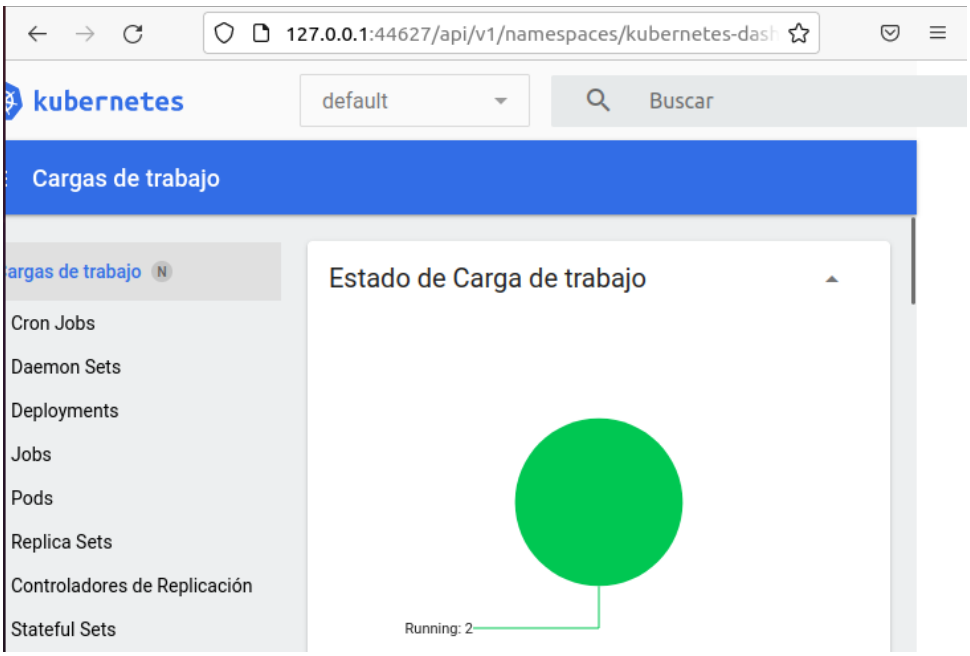

Figura 3.54 Cargas de trabajos para cada servicio

Figura 3.55 Servicios ejecutándose

Posteriormente de revisar los servicios, se realiza la verificación de manera similar a los despliegues los cuales permitirán obtener una mayor seguridad, en cuanto a, si un pod deja de funcionar, de esta forma no se obtendrán problemas con caídas de los pods de cada servicio. En la **Figura 3.56** se encuentra de forma gráfica los despliegues ejecutándose, por otro lado, en la **Figura 3.57** se encuentra en forma de lista los despliegues ejecutándose.

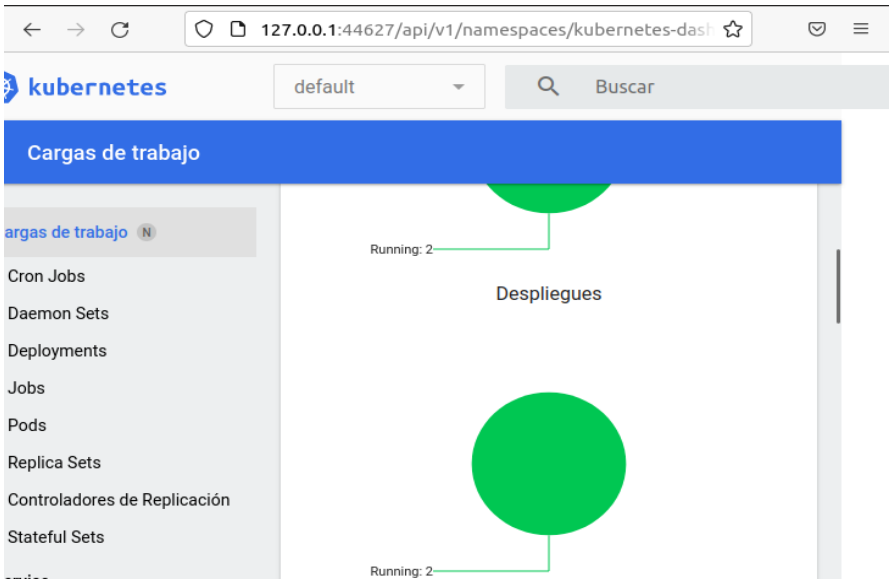

Figura 3.56 Funcionamiento de los despliegues de cada servicio

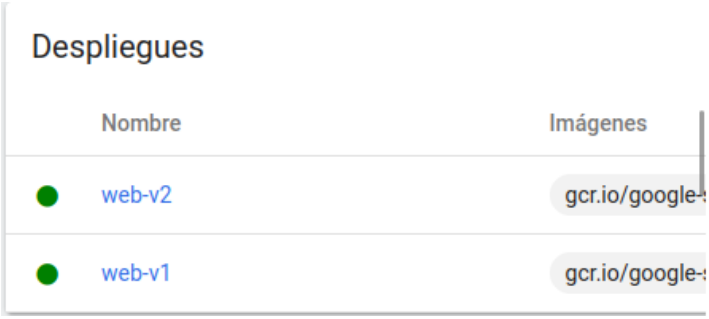

Figura 3.57 Despliegues de cada servicio web

Finalmente, para revisar el direccionamiento de cada página web de cada uno de los *hosts* virtuales se procede a colocar en el navegador del sistema operativo, se ingresa el nombre de dominio de cada servicio para verificar el funcionamiento y comprobar que los servicios web son accesibles. En la **Figura 3.58** y **Figura 3.59** se aprecia el correcto funcionamiento del prototipo de servidor web en un clúster de kubernetes con *hosts* virtuales.

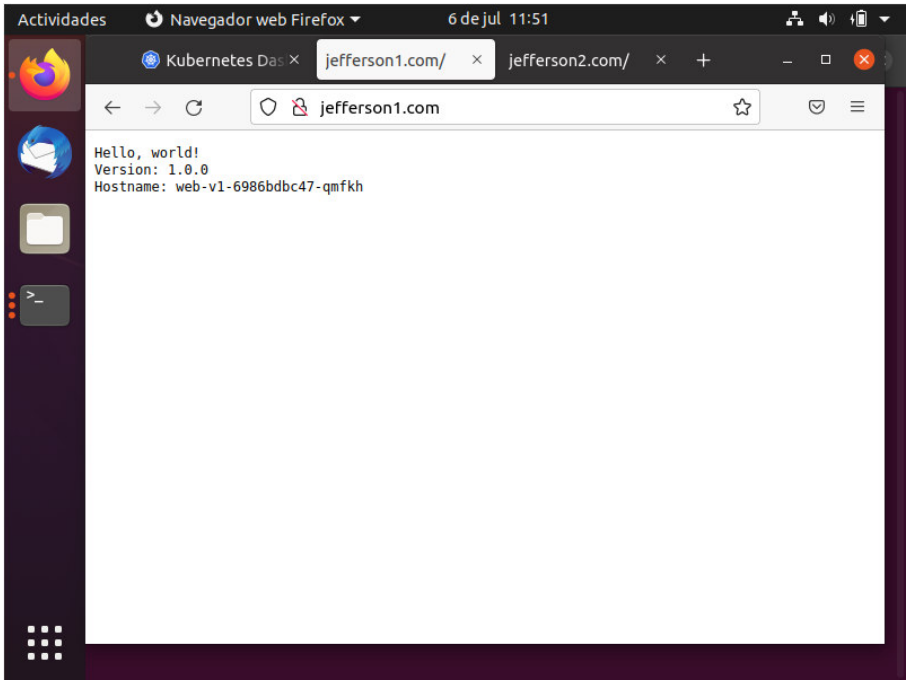

Figura 3.58 Funcionamiento del servidor web 1 con virtual host

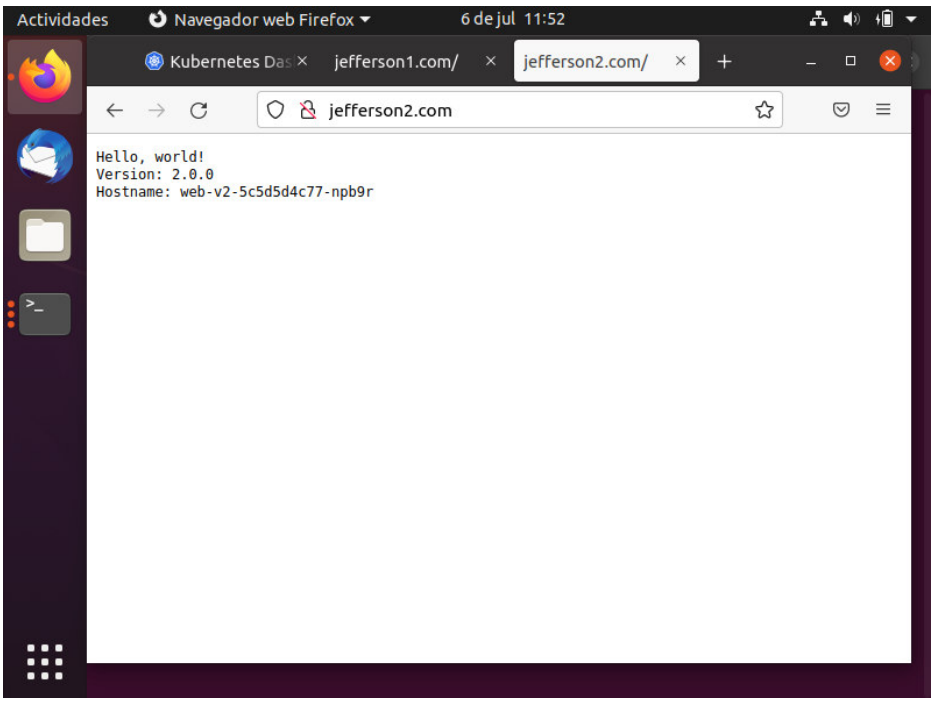

Figura 3.59 Funcionamiento del servidor web 2 con virtual host

4 CONCLUSIONES

- Con el estudio y manejo de Kubernetes se ha conseguido un mejor entendimiento acerca de la ejecución de contenedores, de igual manera, sobre la orquestación y manejo de servicios dentro de ellos.
- Con la utilización de kubernetes permite conseguir una mayor capacidad de respuesta al momento de implementar un servicio o aplicación a un gran número de *hosts* sin disminuir el rendimiento computacional de nuestro equipo.
- La herramienta minikube cumple con los aspectos necesarios para un despliegue eficaz de un clúster de kubernetes, sin embargo, no es la única herramienta, pero si es la más fácil de manejar cuando se empieza con la gestión de contenedores.
- Minikube se destaca por su entrega de software rápida y eficiente, además, de proporcionar un entorno adecuado a las necesidades de un aprendiz en Kubernetes proporcionando un entorno de pruebas para las aplicaciones.
- Con la implementación de un clúster de kubernetes mediante minikube se proporciona una interfaz de control muy intuitiva que permite llevar una mejor gestión de los conjuntos de contenedores y servicios, consiguiendo subsanar el problema que la mayoría de los departamentos de TI se suelen enfrentar cuando una aplicación se encuentra fuera de servicio.
- La incorporación de kubernetes proporciona un escalonado de aplicaciones o servicios, de esta manera, garantiza que siempre va a existir procesos mínimos ejecutándose y si se produce una falla en la aplicación o en el servicio seguirá operativa proporcionando tranquilidad y confiabilidad al departamento de TI.
- Con el desarrollo del presente proyecto de titulación de ello resulta necesario decir que se ha creado un espacio para brindar microservicios mediante peticiones y de igual manera un ambiente de pruebas proporcionando una protección a equipos con ayuda del aislamiento de procesos y aplicaciones.
- Kubernetes proporciona una configuración automatizada para los despliegues de servicios mediante la creación de manifiestos YML asegurando la eficiencia y consiguiendo mejoras en la monitorización y automatización con la ayuda del escalonado de contenedores haciendo frente a grandes demandas de ejecución de un servicio.

5 RECOMENDACIONES

- Se recomienda tener en cuenta los requisitos solicitados por la plataforma de minikube acerca del número de procesadores necesarios, caso contrario, no se instalará.
- En base al proyecto integrador realizado se recomienda colocar el tipo de driver como docker antes de inicializar el clúster para proporcionar un ambiente parecido a un contenedor.
- Instalar con antelación las herramientas necesarias como kubelet, Docker y CURL para tener el organizador y la herramienta necesaria para el instalador binario respectivamente, de esta forma, la instalación del minikube será rápida y sin problemas.
- Para el desarrollo de los archivos YML se recomienda seguir la guía de una plantilla o un ejemplo de un servicio o aplicaciones similares antes desplegadas, de esta forma se disminuirán los errores.
- Verificar el espaciado de los archivos YML de cada servicio, despliegue y aplicación dado que puede ocasionar errores al momento de ejecutarlo.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] R. Z. J. Francisco, «Repositorio Universidad de Almeria,» UNIVERSIDAD DE ALMERIA ESCUELA DE INGENIERIA, 2019. [En línea]. Available: http://repositorio.ual.es/bitstream/handle/10835/8018/TFG_RUIZ%20ZAMORA%2C%20JOSE%20FRANCISCO.pdf?sequence=1&isAllowed=y. [Último acceso: 27 julio 2022].

[2] AMAZON, «¿Qué es Docker?,» AWS, [En línea]. Available: https://aws.amazon.com/es/docker/ . [Último acceso: 27 julio 2022].

[3] «Documentación de Kubernetes,» Kubernetes.io, [En línea]. Available: https://kubernetes.io/es/docs/home/. [Último acceso: 27 julio 2022].

[4] E. Tomás, «¿Qué es Kubernetes y cómo funciona?,» Campus MVP, 14 marzo 2019. [En línea]. Available: https://www.campusmvp.es/recursos/post/que-es-kubernetes-y-como-funciona.aspx. [Último acceso: 27 julio 2022].

[5] «¿Qué es DevOps?,» Azure Microsoft, [En línea]. Available: https://azure.microsoft.com/es-es/overview/what-is-devops/ . [Último acceso: 27 julio 2022].

[6] «El concepto de DevOps,» Red Hat, 19 abril 2018. [En línea]. Available: https://www.redhat.com/es/topics/devops#:~:text=DevOps%20es%20un%20modo%20de,constante%20de%20servicios%20de%20TI. . [Último acceso: 27 julio 2022].

[7] K. Mikael, Learnig DevOps, segunda ed., S. Iqbal, Ed., Birmingham: Packt Publishing Ltd, 2022.

[8] N. Poulton, The Kubernetes, septima ed., LeanPub, 2020.

[9] «¿Qué es Kubernetes?,» RedHat, 27 marzo 2020. [En línea]. Available: https://www.redhat.com/es/topics/containers/what-is-kubernetes. [Último acceso: 27 julio 2022].

[10] «¿Qué es Kubernetes?,» vmware, [En línea]. Available: https://www.vmware.com/latam/topics/glossary/content/kubernetes.html . [Último acceso: 27 julio 2022].

[11] «minikube start,» Minikube, [En línea]. Available: https://minikube.sigs.k8s.io/docs/start/ . [Último acceso: 27 julio 2022].

[12] R. A. M. GABRIELA, «ANÁLISIS COMPARATIVO DE RENDIMIENTO A SERVIDORES WEB DE DISTRIBUCIÓN LIBRE UTILIZANDO APACHE BENCHMARK,» Repositorio Universidad Tecnica de Machala, 22 agosto 2019. [En línea]. Available: http://repositorio.utmachala.edu.ec/bitstream/48000/14567/1/ECFIC-2019-ISIS-DE00034.pdf. [Último acceso: 27 julio 2022].

[13] C. N. Bruno y G. d. l. a. Edisson, «Implementación de un servidor web y un diseño de una página utilizando herramientas de software libre para el dispensario "sagrada familia" de la ciudad de guayaquil,» Repositorio de la universidad Pollitecnica Salesiana, 2017. [En línea]. Available: https://dspace.ups.edu.ec/bitstream/123456789/14162/1/GT001840.pdf. [Último acceso: 27 julio 2022].

[14] «Hosts virtuales,» IBM, 19 junio 2021. [En línea]. Available: https://www.ibm.com/docs/es/was/9.0.5?topic=hosts-virtual . [Último acceso: 27 julio 2022].

[15] «Apache Core Features,» Apache, [En línea]. Available: https://httpd.apache.org/docs/2.4/mod/core.html#virtualhost . [Último acceso: 27 julio 2022].

7 ANEXOS

ANEXO I: Certificado de Originalidad

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 26 de agosto de 2022

De mi consideración:

Yo, FERNANDO VINICIO BECERRA CAMACHO, en calidad de Director del Trabajo de Integración Curricular titulado IMPLEMENTACIÓN DE UN PROTOTIPO DE UN SERVIDOR WEB CON HOSTS VIRTUALES MEDIANTE KUBERNETES asociado al IMPLEMENTACIÓN DEL SERVIDORES CON HERRAMIENTAS DE DEVOPS elaborado por el estudiante JEFFERSON DARIO MONTES BRAVO de la carrera en TECNOLOGÍA SUPERIOR EN REDES Y TELECOMUNICACIONES, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 10%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el link del informe generado por la herramienta Turnitin.

LINK

https://epnecuador-my.sharepoint.com/:b:/g/personal/fernando_becerrac_epn_edu_ec/ESbfL4i5JLBEv2b3nXqDYUEB-vD-zrNwWf4LbG2kd3DrHw?e=n8o8L2

Atentamente,

FERNANDO VINICIO BECERRA CAMACHO

Docente

Escuela de Formación de Tecnólogos

ANEXO II: Enlaces

Anexo II.I Código QR de la implementación y pruebas de funcionamiento