



Centro Profesional
Universidad Europea Madrid
LAUREATE INTERNATIONAL UNIVERSITIES

UNIVERSIDAD EUROPEA



ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

**CICLO FORMATIVO DE GRADO SUPERIOR ADMINISTRACIÓN DE
SISTEMAS INFORMÁTICOS EN RED**

PROYECTO FIN DE CICLO

[Servicios de alta disponibilidad]

**Rodrigo Gálvez Alejandro Cuadrado Carlos Medina Manuel
Rodríguez**

CURSO 2023-24

TÍTULO: Servicios de alta disponibilidad

AUTOR: ALEJANDRO CUADRADO FERNANDEZ RODRIGO GALVEZ CERVANTES
CARLOS MEDINA FERNANDEZ MANUEL RODRIGUEZ PEREZ

TUTOR DEL PROYECTO: DAVID ALVAREZ BOYERO

FECHA DE LECTURA: de Junio de 2024

CALIFICACIÓN:

Fdo: ALEJANDRO CUADRADO FERNANDEZ
RODRIGO GALVEZ CERVANTES CARLOS MEDINA FERNANDEZ MANUEL
RODRIGUEZ PEREZ

DAVID ALVAREZ BOYERO

RESUMEN:

El trabajo de nuestro grupo compuesto por Rodrigo Gálvez, Manuel Pérez, Carlos Medina y yo Alejandro Cuadrado, se basa en la creación de un cluster de Kubernetes de alta disponibilidad y la posibilidad de ofrecérselo a distintas entidades, para que estas puedan implementar nuestros servicios de forma sencilla y rápida, asegurando la escalabilidad automática y alta disponibilidad de estos, para este trabajo hemos necesitado de un periodo de investigación en el que hemos ido recabando información sobre el campo relacionado con el trabajo que queremos hacer, y los campos que lo rodean, en busca de nuevas ideas para complementar a nuestro proyecto y obtener una base de conocimientos mayor para la facilitación de la realización del mismo.

Al inicio del desarrollo de nuestro trabajo, llevamos a cabo una planificación por la cual cada miembro del grupo tenía que ocuparse de un campo relacionado con el trabajo y cada cierto tiempo nos reuníamos y comparábamos nuestros avances y actualizábamos nuestras metas, a su vez hemos necesitado diferentes elementos para la finalización de este, así como máquinas virtuales y ordenadores medianamente potentes para poder “correr” las máquinas virtuales sin mayores problemas.

Hemos encontrado el resultado de nuestro trabajo satisfactorio, ya que hemos sido capaces de completar la mayoría de las metas que nos pusimos en un primer momento e incluso algunas de ellas superarlas, aun así, seguimos teniendo hambre de conocimientos por lo que seguimos buscando formas de innovar y poder hacer nuestro proyecto incluso mejor.

ABSTRACT:

Our group, composed of Rodrigo Gálvez, Manuel Pérez, Carlos Medina, and myself, Alejandro Cuadrado, is focused on creating a high-availability Kubernetes cluster. We aim to offer this to various entities so they can implement our services easily and quickly, ensuring automatic scalability and high availability. For this project, we have undertaken a period of research, gathering information related to the field of our work and surrounding areas, seeking new ideas to complement our project and build a greater knowledge base to facilitate its execution.

At the beginning of our project, we carried out a planning phase where each group member was responsible for a specific area related to the work. We met periodically to compare our progress and update our goals. Additionally, we required various resources for the completion of this project, such as virtual machines and moderately powerful computers to run the virtual machines without significant issues.

We found the outcome of our work satisfactory, as we were able to achieve most of the goals we initially set, and even surpass some of them. Nevertheless, we remain eager to acquire more knowledge, continually seeking ways to innovate and improve our project further.

AGRADECIMIENTOS

Queríamos dar gracias a todas las personas que nos han acompañado a lo largo del proyecto, como David Álvarez, nuestro tutor y una de las fuentes de información más importantes de nuestro proyecto, ya que sin él no se nos habrían ocurrido muchas ideas que nos han ayudado en la creación y resolución del mismo. También queríamos agradecer a Javier Manzanares, una de las principales fuentes de inspiración de nuestro proyecto, ya que gracias a los talleres que nos impartió se nos ocurrió la idea inicial de nuestro proyecto, y también durante el desarrollo del mismo nos ha ayudado en todo lo que ha podido y nos ha aportado todo tipo de ideas. También nos gustaría agradecer a Álvaro Flores, que nos ayudó durante la creación del clúster, al igual que siempre que lo necesitábamos nos resolvía todas las dudas que le comentábamos. Por último, pero no menos importante, nos gustaría agradecer a todos los profesores que hemos tenido durante la duración del grado de ASIR, ya que aunque no hayan tenido una influencia directa en nuestro proyecto, sí que nos han otorgado todo tipo de conocimientos a lo largo de estos dos años, que nos han ayudado en mayor o menor medida durante todo este proceso, y no menos importante, nos han dado la oportunidad de conocernos y poder trabajar en este proyecto juntos. Por eso y por mucho más,

Muchas gracias.



Esta obra se distribuye bajo una licencia Creative Commons.

Se permite la copia, distribución, uso y comunicación de la obra si se respetan las siguientes condiciones:

- Se debe reconocer explícitamente la autoría de la obra incluyendo esta nota y su enlace.
- La copia será literal y completa
- No se podrá hacer uso de los derechos permitidos con fines comerciales, salvo permiso expreso de los autores.

El texto precedente no es la licencia completa sino una nota orientativa de la licencia original completa(jurídicamente válida) que puede encontrarse en: <http://creativecommons.org/licenses/by-nc-nd/3.0/deed.es>



INDICE

1. INTRODUCCIÓN.....	1
1.1. OBJETIVOS	1
1.2. MOTIVACIÓN.....	2
1.3. ANTECEDENTES.....	2
2. DESARROLLO DE LA PRÁCTICA.....	2
2.1. MATERIAL.....	2
2.2. PLANIFICACIÓN	6
Gráfico planificación de tiempo.....	7
2.3. DESCRIPCIÓN DEL TRABAJO REALIZADO	8
Gráfico costes principales.....	27
Gráfico costes materiales.....	28
Gráfico uso de equipos	29
2.4. RESULTADOS Y VALIDACIÓN.....	30
3. CONCLUSIONES	32
3.1. APORTACIONES	33
3.2. TRABAJO FUTURO.....	34
4. BIBLIOGRAFÍA Y WEBGRAFÍA	35
5. ANEXOS	I
5.1. CREACIÓN DE IMÁGENES DOCKER	I
Listado 1: Imagen DHCP comandos	III
Listado 2: Imagen DNS comandos.....	IX
Listado 3: Imagen Nginx comandos.....	XII
5.2. CREACIÓN CLÚSTER DE KUBERNETES.....	XIII
Estructura del clúster.....	XIII
Listado 4: Configuración HAProxy y Keepalived comandos	XIV
Listado 5: Comandos creación máquina Master.....	XV
Listado 6: Comandos creación cluster máquina Master	XV
Listado 7: Comandos unión cluster máquina Master.....	XVI
Listado 8: Comandos unión cluster máquina Worker	XVI
Listado 9: Comandos iniciar clúster.....	XVII
5.3. CREACIÓN Y CONFIGURACIÓN DE VPN.....	XX
Listado 10: Comandos configuración Wireguard máquina Maestro	XXII
Listado 11: Comandos configuración Wireguard máquina cliente	XXII
5.4. CREACIÓN PÁGINA WEB	XXIII
Listado 12: Código Index.html.....	XXIV
Listado 13: Código Coming_soon.html	XXV
Listado 14: Código contáctanos.html	XXVI
Listado 15: Código login.html	XXVII
Listado 16: Código quienes_somos.html	XXVIII
Listado 17: Código registro.html.....	XXIX
Listado 18: Código style.css	XXXV



1. INTRODUCCIÓN

Creación de un sistema de servicios de alta disponibilidad mediante Kubernetes. Así se podría resumir nuestro proyecto en una simple frase. Tanto Carlos Medina como Rodrigo Gálvez, Manuel Pérez y yo, Alejandro Cuadrado, hemos estado trabajando e investigando para poder realizar este proyecto durante los últimos 3 meses, y en esta memoria les presentamos el proceso por el que hemos pasado durante nuestro proyecto, así como nuestras dificultades, cambios que hemos tenido que realizar, los distintos errores y problemas que nos hemos encontrado, junto con mejoras que no sabíamos que se podían implementar, y la finalización del proyecto.

1.1. Objetivos

Una vez que supimos que nuestro trabajo iba a consistir en el uso y la modificación de Kubernetes, y después de un poco de investigación, nos pusimos unos objetivos que seguir para no ir sin rumbo y tener un mapa de nuestros objetivos en relación con otras variables, como el tiempo que teníamos para trabajar en el proyecto o si necesitábamos algún elemento extra aparte de los que nosotros ya disponíamos.

Listado de los objetivos principales de nuestro proyecto:

- Base de Kubernetes
- Imágenes Docker
- Salto a Internet
- Alta disponibilidad
- Contacto con empresas

Para el primer objetivo, necesitábamos un sistema operativo basado en Linux, en el que nuestro objetivo era que cada vez que iniciara, nuestro servidor de Kubernetes con minikube, se hiciera una revisión de las imágenes Docker que habíamos configurado previamente en el mismo y estas se iniciaran automáticamente.

El segundo objetivo consistía en la implementación de imágenes Docker en nuestro clúster de Kubernetes, pero una vez que nos dimos cuenta de que necesitábamos crear nuestras propias imágenes, ese paso pasó a ser el nuevo objetivo. Por lo tanto, tuvimos que crear una imagen Docker para cada servicio que fuéramos a implementar en el clúster de Kubernetes.

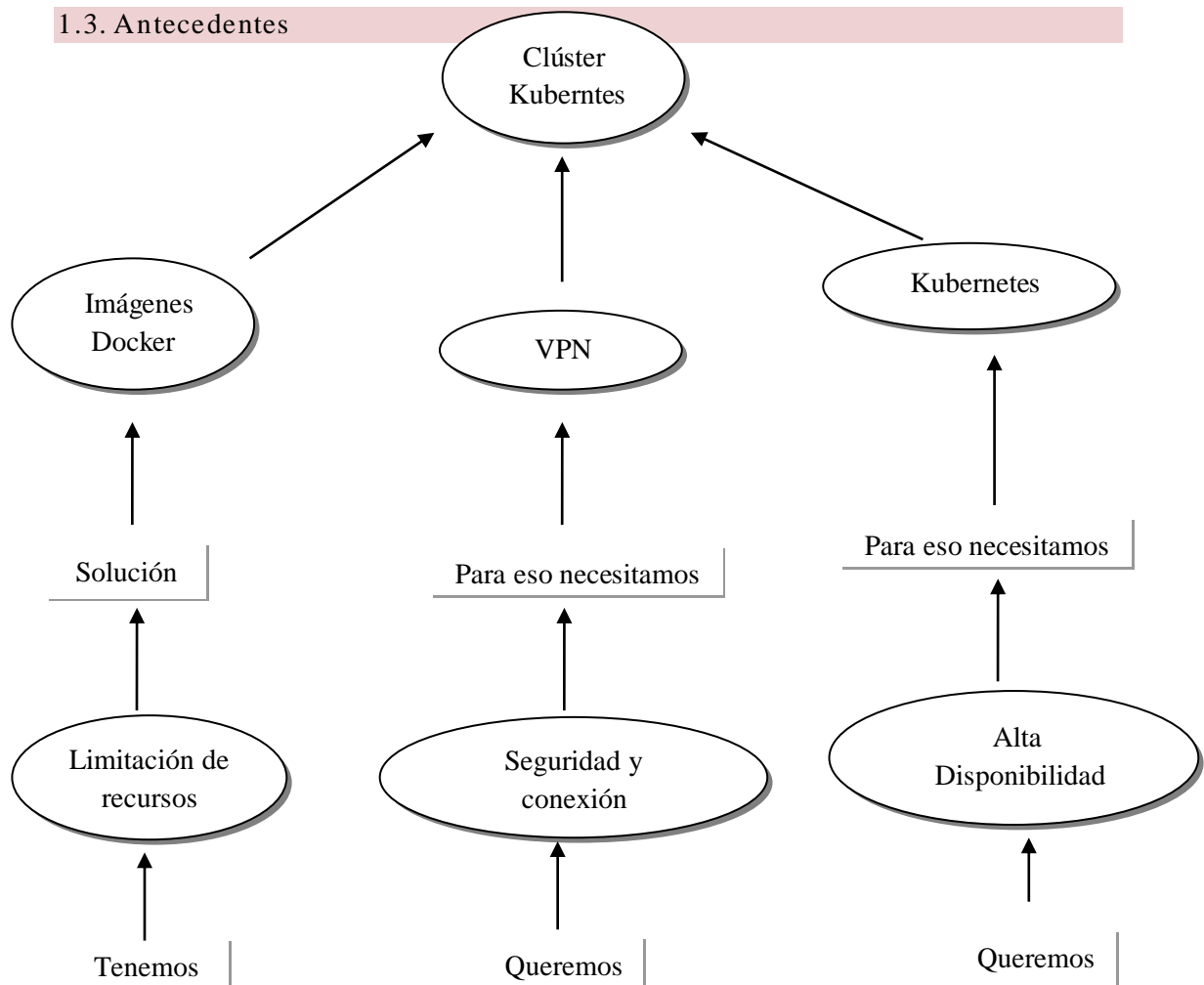
Para la conexión a Internet, lo que teníamos que hacer era, una vez que tuviéramos las imágenes Docker creadas junto con el clúster de Kubernetes totalmente operativo, en lugar de implementarlo todo en una red local, lo implementaríamos a nivel online.

Una vez que tengamos configurado y en funcionamiento el servidor, pasaremos a crear un sistema de alta disponibilidad añadiendo dos equipos o más que se ejecuten al mismo tiempo junto con el principal, y configurándolos de tal manera que si uno de ellos deja de funcionar, los otros equipos levantarían uno nuevo rápidamente y de manera eficaz.

1.2. Motivación

En cuanto nuestro grupo, compuesto por Carlos Medina, Rodrigo Gálvez, Manuel Pérez y yo, Alejandro Cuadrado, supimos que teníamos que hacer un proyecto de fin de grado o como también se le suele llamar PFG, nos adentramos en un trabajo de investigación sobre lo que queríamos hacer. En primera instancia, empezamos a desarrollar una idea sobre la creación de páginas web para ayudar a empresas con sus problemas informáticos, pero nos pareció directamente un trabajo con muy poca profundidad. También se nos ocurrió la idea de montar varios equipos con ordenadores que ya no se usaran para armar un superordenador y poder usarlo de manera eficiente, pero descartamos la idea porque no nos llamaba tanto la atención. Fue entonces, en los últimos días de clase antes del inicio de las prácticas, cuando tuvimos una serie de talleres, a cargo de Javier Manzanares Flores, en los cuales descubrimos Kubernetes y se nos dio una comprensión más profunda sobre los contenedores Docker. Fue en ese momento cuando se nos ocurrió la idea de crear, con los conocimientos que habíamos adquirido en los talleres, junto con un trabajo de investigación extra por nuestra parte, un clúster de Kubernetes de alta disponibilidad para poder ofrecerlo a distintos tipos de empresas y brindarles un servicio tanto seguro como rápido y eficiente.

1.3. Antecedentes





2. DESARROLLO DE LA PRÁCTICA

"Labor omnia vincit" es una frase del latín que se traduce como "El trabajo todo lo vence", y podemos decir con seguridad que es una frase que define muy bien nuestro trabajo en torno al proyecto. No importaba la cantidad de muros o piedras que encontráramos en el camino; trabajando duro y con esfuerzo, hemos sido capaces de crear un proyecto del que nos sentimos muy orgullosos.

2.1. Material

En la creación de un proyecto de esta envergadura, siempre es necesario usar un amplio abanico de materiales y herramientas. A continuación, enumeraremos tanto las herramientas utilizadas a lo largo del mismo, como distintos materiales usados como referencia, y proporcionaremos un breve resumen de los mismos para los lectores menos familiarizados con el campo en el que hemos trabajado.

- Oracle VM VirtualBox es un hipervisor alojado para virtualización x86 desarrollado por Oracle. Para comprender mejor esta definición, primero necesitamos entender el término hipervisor, también conocido como monitor de máquina virtual o virtualizador. Se trata de un tipo de software desarrollado para la creación y ejecución de máquinas virtuales. Además, existen dos tipos de hipervisores clasificados en la tesis de Robert P. Goldberg en 1973. El primer tipo de hipervisores son los bare-metal, que se alojan directamente en el hardware del host para controlarlo y administrar los sistemas operativos invitados. Mientras que el segundo tipo de hipervisores son los alojados, que se ejecutan en un sistema operativo convencional junto con otros programas informáticos más comunes. Como seguramente se pregunten, el x86 hace referencia a los microprocesadores compatibles con el juego de instrucciones Intel 8086. Ahora, sabiendo mejor de dónde viene la denominación de hipervisor, podemos continuar explicando el funcionamiento básico de VirtualBox. Como se ha podido entender en la explicación sobre los tipos de hipervisores, VirtualBox es un hipervisor de tipo 2, cuyo uso principal es el de virtualizar sistemas operativos en el interior de nuestro ordenador, dando lugar a lo que se conoce como máquina virtual. VirtualBox es capaz de instalarse en una amplia variedad de sistemas operativos, pero en nuestro caso, debido a los sistemas operativos de nuestros equipos, lo hemos instalado tanto en Windows 10, Windows 11 y Linux. A su vez, hemos virtualizado distintos sistemas operativos, los cuales han sido Ubuntu 22.04 y Ubuntu 22.04. Gracias a Oracle VM VirtualBox, hemos sido capaces de llevar a cabo nuestro proyecto, ya que sin la posibilidad de virtualizar sistemas operativos completos y crear distintas máquinas virtuales con ellos, nos habría resultado imposible continuar nuestro trabajo, debido al alto número de equipos necesarios, tanto durante el desarrollo del mismo, ya que hemos necesitado más de 10 sistemas operativos distintos para la creación de los elementos necesarios de nuestro trabajo, así como a la hora de la presentación de nuestro proyecto.

- VMware Workstation es una herramienta desarrollada por VMware, una filial de Broadcom Inc., una empresa enfocada principalmente en el desarrollo y diseño de productos de software de infraestructura y semiconductores, con sede en Estados Unidos y proveedores mundiales de sus productos. VMware Workstation es un programa cuya finalidad es proporcionar un software de virtualización para ordenadores x86, al igual que Oracle VM VirtualBox. Además de VMware Workstation, VMware ofrece otros programas como VMware Server y VMware Player, que funcionan como apoyo a la hora de utilizar VMware Workstation. A diferencia de Oracle VM VirtualBox y aunque es cierto que VMware ofrece pruebas gratuitas, en última instancia no se trata de un programa de código abierto, sino que es un programa de pago anual cuyo precio por suscripción ronda los 120 dólares. Dado que es una empresa estadounidense, el precio en euros sería de unos 110.40 euros. En nuestro caso, hemos usado la versión gratuita, ya que no necesitábamos la versión de pago o Pro, según la denomina VMware, para nuestras necesidades.
- Terminal de comandos de Ubuntu: Como ya hemos explicado, los sistemas operativos en los que hemos basado la mayoría de nuestro trabajo son de base Linux, más concretamente Ubuntu. En estos sistemas, la mayoría de los elementos que hemos programado han sido a través del terminal de comandos. A continuación, presentaré una pequeña demostración de los comandos más importantes y su explicación:
 - Sudo su: nos otorga acceso al modo administrador en la consola de comandos. Se obtiene el mismo resultado si agregamos sudo delante del comando que vamos a utilizar. Los comandos que enseñaré a continuación llevarán sudo delante, ya que prefiero usarlo así para evitar problemas de permisos.
 - sudo apt update: comando usado para actualizar la lista de paquetes disponibles en el sistema..
 - sudo apt upgrade: comando usado para actualizar la lista de paquetes instalados, que se actualizan mediante sudo apt update.
 - sudo mkdir <nombre del directorio>: comando usado para crear un nuevo directorio o carpeta.
 - sudo rm <nombre del directorio>: comando para eliminar un directorio vacío.
 - sudo nano <nombre del archivo>: comando usado para editar un archivo existente o crear uno nuevo y editarlo directamente. Todo dependerá de si el <nombre del archivo> introducido existe ya o no.
 - cd <nombre del directorio>: comando usado para moverse entre directorios. Si queremos movernos a un directorio a varios directorios de distancia, en lugar de introducir el comando continuamente, podemos usar la siguiente sintaxis: cd /<nombre del directorio en el que se encuentra el directorio destino>/<nombre del directorio>, lo que nos llevará directamente al directorio al que queremos ir.
 - ls: muestra los elementos del directorio en el que nos encontramos en ese momento, como archivos, carpetas u otros directorios.
 - sudo apt install <nombre del programa>: instala el programa que deseamos en nuestro sistema operativo.

- DHCP o Protocolo de Configuración Dinámica de Host es un protocolo cliente-servidor que automáticamente asigna direcciones IP a los dispositivos en una red, junto con la información asociada como la máscara de subred y la puerta de enlace predeterminada. Esto facilita la creación de redes al permitir la comunicación entre dispositivos sin necesidad de configurar manualmente las direcciones IP, simplificando así la comunicación.
- DNS, que significa Sistema de Nombres de Dominio, nos libera de la tarea de tener que memorizar las direcciones IP de todos los sitios web que queremos visitar. En lugar de eso, traduce automáticamente esas direcciones a un lenguaje que podemos entender. Para aquellos menos familiarizados con las redes, toda la información se almacena en forma numérica, y cuando nos conectamos a una red con un dispositivo, se nos asigna una dirección IP para identificarnos. Lo mismo ocurre al buscar páginas web, pero estas direcciones IP suelen permanecer iguales hasta que el sitio se cierra, para evitar cambios constantes de nombres. Así que la labor del DNS es permitirnos buscar cualquier página web, como por ejemplo Google, sin necesidad de conocer su dirección IP.
- Nginx es uno de los software de servidor web de código abierto más famosos. Pero, ¿qué es un servidor web? Un servidor web es aquel cuya principal misión es devolver información, como páginas web, a un usuario que ha solicitado esa información. Es decir, son los responsables de que si un usuario quiere ver una página web específica en su equipo, pueda hacerlo.
- Docker es una plataforma de software que permite la creación e implementación eficiente y rápida de aplicaciones. En nuestro caso, lo hemos utilizado para almacenar los servidores DHCP, DNS y Nginx, permitiendo así una instalación rápida y eficiente en distintos dispositivos. ¿Cómo almacena Docker el software de los distintos servidores? Utiliza una serie de programas de empaquetamiento llamados contenedores, que incluyen todo lo necesario para ejecutar el software. Estos contenedores contienen bibliotecas, código, herramientas del sistema y tiempo de ejecución. A su vez, se deben utilizar distintos comandos en el terminal de Linux al usar Docker, y debido a su importancia, vamos a enseñarlos y explicar los más importantes:
 - `sudo apt install docker-ce`: Comando utilizado para instalar Docker.
 - `sudo systemctl status docker`: Comando utilizado para ver el estado de Docker. También se pueden usar los términos ``stop`` para detenerlo, ``start`` para iniciarlo y ``restart`` para reiniciarlo, todo dentro del mismo comando.
- Kubernetes: Es un software de administración centrado en la operación de contenedores. Sus principales utilidades son la implementación, escalado y administración de aplicaciones alojadas en contenedores. Se trata de un sistema de código abierto, por lo que es totalmente gratuito. Es uno de los software de administración de contenedores de código abierto más populares en los últimos días debido a todas las funciones que incluye, así como a los comandos ya integrados para la implementación, actualización y escalado de aplicaciones según las necesidades de cada usuario, y a la automatización de tareas operativas para la administración de los contenedores.

- Visual Studio Code: Es un editor de código multiplataforma y de código abierto. Visual Studio Code es una de las herramientas más utilizadas por los programadores, tanto nuevos como experimentados, debido a sus numerosas ventajas sobre otros editores de código abierto. Entre estas ventajas se incluye una amplia variedad de extensiones creadas por la comunidad para un uso más eficiente y rápido de Visual Studio Code. Además, ofrece mejoras como un terminal integrado en el propio programa y, no menos importante, la capacidad de personalizar la interfaz del mismo según tus preferencias. Estas son algunas de las razones, junto con su facilidad de uso, que han llevado a Visual Studio Code a ser uno de los editores de código más famosos en la actualidad.
- Amazon Web Services: Es una parte de la empresa mundialmente conocida Amazon que ofrece una serie de servicios en la nube. Proporciona una amplia gama de servicios en la nube que permiten tanto a desarrolladores individuales como a grandes empresas construir y gestionar servicios y aplicaciones a través de centros de datos dirigidos por Amazon en todo el mundo. AWS ofrece una amplia gama de utilidades, como alta disponibilidad, gran fiabilidad y seguridad. Uno de los servicios más importantes es OpenVPN, que se trata de una solución de software que implementa técnicas de VPN para crear conexiones seguras de punto a punto en configuraciones enrutadas de acceso remoto. Suele ser utilizada de manera empresarial debido a su gran facilidad de implementación y uso.
- - VPN (Red Privada Virtual): Una VPN, también conocida como red privada virtual, es una tecnología que permite establecer una conexión segura y cifrada a través de una red normalmente menos segura, como suele ser el caso de internet. Para comprender de manera sencilla para qué sirve una VPN, esta crea un túnel entre el equipo del usuario y el servidor VPN, por lo que toda la información que se pasa a través de este túnel se cifra y está protegida de usuarios externos. Las VPN suelen ser usadas en gran medida en empresas debido a toda la seguridad que ofrecen. Sin embargo, también son utilizadas por gran parte de la población debido a esta misma razón, junto con diferentes tipos de usos que pueden tener, como el acceso remoto a diferentes equipos, la privacidad adicional que ofrecen e incluso, en muchos casos, son utilizadas con el único propósito de saltarse las restricciones geográficas de distintas páginas web, que pueden ser desde pedir un paquete desde otro país en Amazon, hasta ver series que ya no están disponibles en tu región en Netflix.

2.2. Planificación

El 10/03/2024 comenzamos con el trabajo de investigación que serviría como base de nuestro trabajo. La primera reunión informativa la tuvimos el 25/03/2024. En esta reunión, presentamos los resultados de nuestro trabajo de investigación realizado en las 2 semanas anteriores y presentamos nuestras primeras ideas para nuestro proyecto. Sin embargo, debido al poco trabajo de investigación realizado y a que no teníamos una idea clara, en ese momento aún no teníamos completamente definida la base de nuestro proyecto. Sabíamos el rumbo que nos gustaría tomar, pero sin tener aún una idea clara, nos dimos otra semana dedicada a trabajo de investigación.

El 03/04/2024 tuvimos nuestra segunda reunión informativa. En esta reunión, junto con la investigación realizada en las 3 últimas semanas, pudimos obtener una base sólida para nuestro proyecto, con la que estábamos bastante satisfechos. Por lo tanto, procedimos a iniciar la planificación para la realización del mismo.

El 15/04/2024 tuvimos nuestra tercera reunión informativa para exponer las herramientas y materiales que encontramos cada uno durante la semana y media pasada, y así poder empezar con la realización del proyecto lo antes posible. Durante esta reunión, destacamos las partes del trabajo que pensamos que nos llevarían más tiempo completar y asignamos a cada miembro del grupo un trabajo específico para poder comenzar así con el proyecto.

El 23/04/2024 se llevó a cabo nuestra cuarta reunión informativa, en la que discutimos las dificultades encontradas por cada uno en sus campos. En esta reunión, pudimos llegar a una idea más clara de cómo queríamos que fuera el resultado final de nuestro proyecto, gracias al continuo trabajo de investigación realizado en las últimas 4 semanas y a los distintos errores o problemas que nos encontrábamos en el camino, tanto por parte de la información recopilada como por parte de las herramientas utilizadas.

Después de otra semana de trabajo, el 30/04/2024 llevamos a cabo nuestra quinta reunión informativa, en la que presentamos los avances hechos por cada miembro del grupo, así como las dificultades que habíamos encontrado, las partes del trabajo que nos dimos cuenta que tuvimos que rehacer, así como mejoras que pudimos encontrar durante el tiempo dedicado a completar el proyecto.

El 7/5/2024 se llevó a cabo nuestra sexta reunión informativa, en la que pudimos mostrar nuestros avances en los distintos campos que nos asignamos, y a su vez pudimos entregarnos nuevos trabajos para poder avanzar en el proyecto.

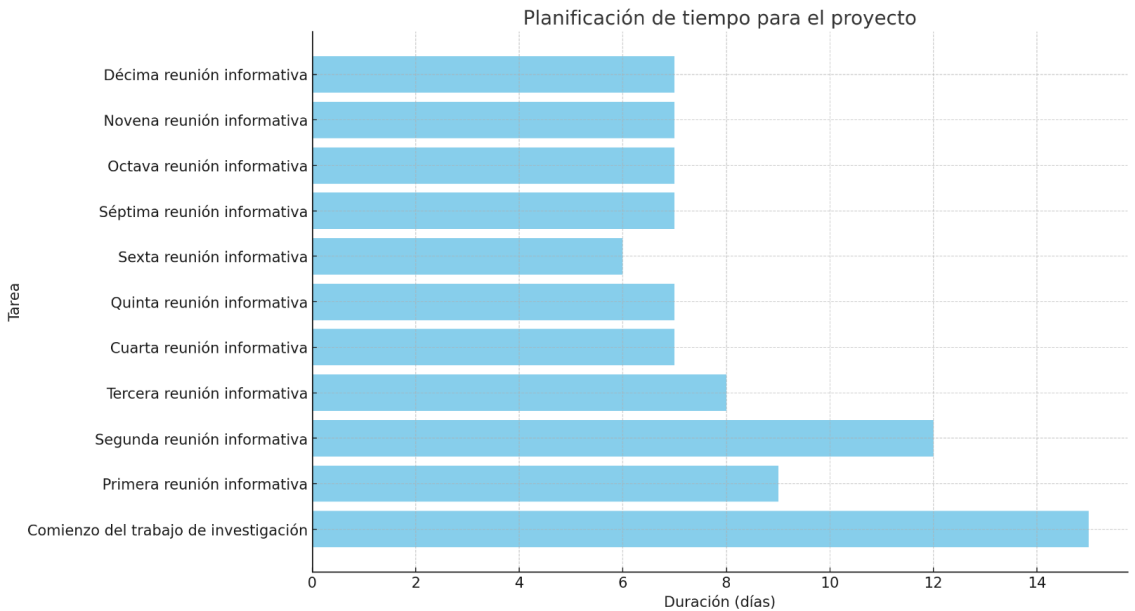
El 13/05/2024 se llevó a cabo nuestra séptima tutoría, en la que, junto con ayuda de personas ajenas a nuestro proyecto, se nos ofrecieron nuevas soluciones a distintos problemas que teníamos sobre nuestro trabajo, así como distintas formas de completar las siguientes partes de nuestro trabajo y a su vez brindarnos un poco más de inspiración.

El 20/05/2024, después de seguir trabajando una semana más, tuvimos nuestra octava tutoría, en la que cada uno de nosotros presentamos los logros que habíamos conseguido hasta ahora y compartimos distintos puntos de vista sobre cómo veíamos el futuro del trabajo, así como las ideas que debíamos descartar y en las que nos debíamos centrar más.

El 27/05/2024, nuestra novena reunión informativa se llevó a cabo y empezamos a dar los retoques finales a nuestro proyecto. Hicimos otra presentación sobre los logros finales alcanzados por cada uno, así como partes del trabajo que, junto con el poco tiempo que nos quedaba, veíamos imposibles de completar y tuvimos que descartar.

El 03/06/2024 se llevó a cabo nuestra décima y última reunión informativa, en la que dimos los retoques finales a nuestro proyecto y comenzamos a preparar tareas extra del mismo. Cabe destacar que, aunque es verdad que las reuniones más importantes se efectuaron en los días presentados anteriormente, el grupo estuvo en contacto diariamente usando distintas herramientas de comunicación como fueron WhatsApp, Gmail y Discord.

Gráfico planificación de tiempo



2.3. Descripción del trabajo realizado

Comenzamos el trabajo de investigación el 10/03/2024. Ese mismo día hicimos una pequeña reunión sobre el campo en el que queríamos desarrollar nuestro proyecto y el rumbo que deseábamos que tuviera a lo largo de su realización. En esta primera reunión, decidimos que nuestro trabajo estaría centrado en temas relacionados con Docker y Kubernetes. Esta decisión fue influenciada por una serie de talleres impartidos por Javier Manzanares Flores antes de las prácticas universitarias, que trataban precisamente sobre la creación de clústeres de Kubernetes y su uso junto con contenedores Docker. Durante las dos primeras semanas, cada miembro del grupo se dedicó a buscar información relacionada con estos temas.

Alejandro Cuadrado se centró principalmente en Docker y encontró una gran cantidad de información sobre la creación de contenedores Docker y la gestión de imágenes.

Carlos Medina y Rodrigo Gálvez se enfocaron en Kubernetes y recopilamos una gran cantidad de información sobre su principal utilidad, así como abundante documentación sobre el tema.

Manuel Rodríguez dedicó estas dos primeras semanas a investigar sobre VPN y su creación e implementación.

Después de esta primera reunión y tener claro hacia dónde dirigiríamos el proyecto, establecimos los primeros objetivos que servirían como los cimientos de nuestro trabajo. Para mantener un registro de nuestros avances y de las tareas que no pudimos realizar debido a la imposibilidad o falta de conocimientos, decidimos crear una serie de tablas donde detallaríamos nuestros siguientes objetivos y el trabajo realizado hasta el momento. Nuestros objetivos iniciales fueron centrar definitivamente la dirección del proyecto, identificar las herramientas necesarias y darle un sentido y una finalidad clara a nuestro trabajo.

Objetivos iniciales	Dirección del proyecto	
	Herramientas necesarias	
	Utilidad del proyecto	
	Finalidad del proyecto	

El 25/03/2024 tuvimos nuestra primera reunión, durante la cual intercambiamos la información recopilada en las últimas dos semanas. Durante esta reunión, descartamos la información que no era relevante para nuestros objetivos y comenzamos a definir nuestras metas iniciales. Durante esta discusión, quedó claro que queríamos centrarnos en Kubernetes y en la implementación de contenedores Docker. Decidimos que nuestro objetivo era crear un servicio dirigido a todo tipo de empresas, desde tecnológicas hasta aquellas especializadas en otros campos como deportes o legal. Como objetivos para la próxima semana, nos enfocamos en cómo crear este servicio.

Después de esta reunión inicial, establecimos los primeros objetivos que servirían como base para nuestro trabajo futuro. Durante las siguientes dos semanas de investigación, Carlos Medina y Rodrigo Gálvez se centraron en la documentación proporcionada por Kubernetes, pero esta vez, con una orientación más específica hacia la creación de distintos clústeres de Kubernetes y sus posibles implementaciones. Comenzaron a realizar cursos relacionados con la creación de clústeres ofrecidos gratuitamente por Kubernetes.

Alejandro Cuadrado se centró en la descarga e implementación de servicios como DHCP, DNS, Apache y NAS mediante contenedores Docker en máquinas virtuales, con el objetivo de crear nuestro propio servidor para ofrecerlo a las empresas.

Manuel Rodríguez continuó su búsqueda de documentación sobre VPN, centrándose en la creación e implementación de estas en un clúster de Kubernetes.

Durante esta etapa, logramos completar nuestros objetivos iniciales, que incluían la búsqueda de inspiración, definir la dirección de nuestro proyecto hacia la creación de un clúster de Kubernetes con el uso de imágenes Docker, identificar las herramientas necesarias como los virtualizadores para utilizar imágenes Docker de DHCP, DNS, Apache y NAS, comprender el uso potencial de nuestro proyecto como un servicio de suscripción mensual o anual para empresas y usuarios, y definir la finalidad de nuestro proyecto: ofrecer a empresas y usuarios la posibilidad de instalar una base de Minikube con servicios principales para empresas, facilitando la implementación de servicios y garantizando la escalabilidad automática y la alta disponibilidad.

Objetivos iniciales	Dirección del proyecto	X
	Herramientas necesarias	X
	Utilidad del proyecto	X
	Finalidad del proyecto	X

El 15/04/2024 nos reunimos nuevamente, habiendo completado los objetivos iniciales que nos propusimos. Durante esta reunión, cada miembro del grupo comenzó con su parte del proyecto. Alejandro Cuadrado inició la instalación de las imágenes Docker de DHCP, DNS, Apache y NAS en una misma máquina virtual. Para ello, primero creó la máquina virtual utilizando una imagen de Ubuntu 22.04 para una instalación más cómoda y segura. Luego, ejecutó los comandos más importantes en el terminal de Ubuntu, como `sudo apt update` para actualizar los paquetes del equipo, y `sudo apt upgrade` para instalar las últimas versiones de los paquetes. A continuación, instaló Docker en el equipo con `sudo apt install docker-ce`. Luego, creó un nuevo directorio llamado `Docker_DHCP` con `sudo mkdir Docker_DHCP`. Después, descargó la imagen Docker con el comando `sudo docker pull networkboot/dhcpd`. Una vez descargada la imagen, modificó los archivos de configuración con `sudo nano dhcpd.conf`, configurando los parámetros necesarios. Luego, ejecutó el contenedor Docker con el comando `sudo docker run -v /home/alumno/docker_DHCP:/data --network host --name dhcp_GAS --restart unless-stopped networkboot/dhcpd enp0s3`.

Una vez finalizado el servicio DHCP, Alejandro comenzó a trabajar en el servicio DNS utilizando la misma máquina virtual. Para este servicio, ejecutó nuevamente los comandos `sudo apt update` y `sudo apt upgrade` para actualizar completamente la máquina. Luego, instaló `bind9` con sus dependencias con el comando `sudo apt install bind9 bind9utils bind9-doc -y`. Creó un nuevo directorio llamado `bind9-dns-docker`, y dentro de este, creó cuatro archivos que servirían como base para la imagen Docker que descargarán a continuación. Utilizó el comando `sudo nano` para crear los archivos `forward.Goldenarrow.com`, `reverse.Goldenarrow.com`, `named.conf.local`, y `named.conf.options`. Una vez creados los archivos, descargarán la imagen Docker de `bind9` desde Docker Hub con `sudo docker pull ubuntu/bind9`, y ejecutaron el contenedor con `sudo docker run --name bind9-dns-server --rm -it --net host ubuntu/bind9`. Después de completar los servicios de DHCP y DNS, Alejandro comenzó con la instalación del servicio Apache. Nuevamente, ejecutó los comandos `sudo apt update` y `sudo apt upgrade` para actualizar la máquina virtual. Luego, descargó la imagen Docker del servicio Apache desde Docker Hub con `sudo docker pull httpd`, y ejecutó la imagen con `sudo docker run -dit --name myapache -p 8080:80 httpd`.

Durante este período, Carlos Medina y Rodrigo Gálvez continuaron recopilando documentación sobre Kubernetes, pero también buscaron información sobre Docker para ajustar algunas partes

del trabajo. Manuel Rodríguez continuó buscando documentación sobre cómo crear una VPN. Todos estos avances se compartieron en una reunión de grupo intermedia.

El 30/04/2024 tuvimos una reunión importante en la que decidimos crear nuestras propias imágenes Docker y montarlas en la misma máquina virtual y contenedor, siguiendo el consejo de nuestro profesor David Alvarez. Rodrigo Gálvez y Alejandro Cuadrado se encargarían de buscar documentación sobre cómo crear imágenes Docker desde cero y cómo subirlas a Docker Hub, mientras que Carlos Medina comenzó a crear el clúster de Kubernetes en una máquina virtual y Manuel Pérez empezó a trabajar en la creación de una página web para la empresa. Con estos nuevos objetivos, continuamos avanzando en nuestro proyecto.

Objetivos finales	Creación de imágenes Docker desde 0	
	Creación de clúster de Kuberntes	
	Implantación de VPN	
	Creación de Pagina Web	

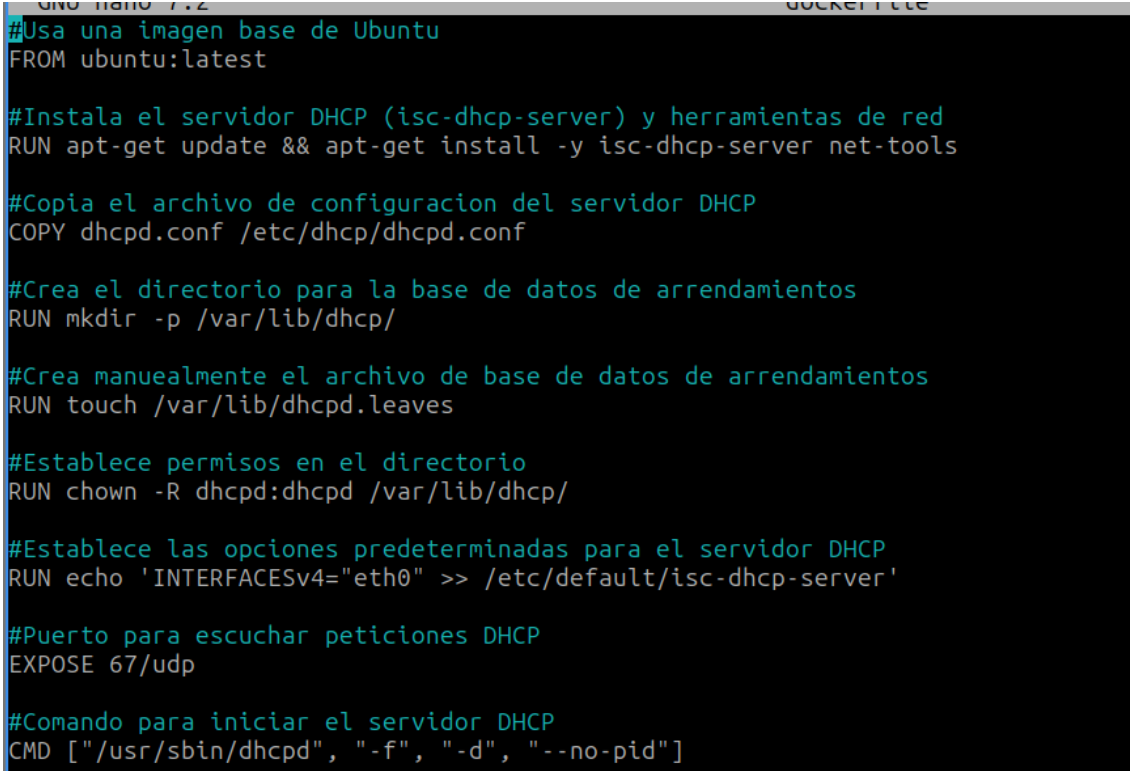
La reunión del 07/05/2024 fue otra de las más importantes para nuestro grupo, ya que coincidió con nuestra reunión programada con nuestro tutor, David Alvarez, y además asistieron antiguos alumnos suyos, entre los cuales estaba Javier Manzanares, quien nos había impartido los cursos tanto de Kubernetes como de Docker. Aprovechamos esta oportunidad para presentarles nuestro trabajo hasta la fecha y plantearles diversas dudas. Uno de los puntos clave que nos aclararon fue que era prácticamente imposible alojar las tres imágenes en un mismo docker. Con esta información, modificamos nuestro enfoque y decidimos crear tres imágenes distintas en tres máquinas virtuales. Además, recibimos consejos sobre la forma de crear un clúster de Kubernetes. Gracias a esta reunión, pudimos iniciar la recta final de nuestro proyecto.

Durante este período, continuamos reuniéndonos periódicamente y mantuvimos comunicación casi diaria hasta la fecha límite para el proyecto. Logramos avanzar significativamente en todos los aspectos en los que nos enfocamos.

La creación de imágenes Docker desde cero en distintas máquinas virtuales comenzó de nuevo con la creación de la imagen DHCP. Al tener una nueva máquina, lo primero que hicimos fue actualizar los paquetes de la máquina con `sudo apt update` y `sudo apt upgrade`. Luego, instalamos las dependencias de Docker para asegurar su correcto funcionamiento con `sudo apt install apt-transport-https ca-certificates curl software-properties-common`. A continuación, añadimos la clave GPG con `curl -fsSL https://download.docker.co/linux/ubuntu/gpg | sudo apt-key add -` y agregamos el repositorio Docker con `sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu \$(lsb_release -cs) stable"`. Después de actualizar los nuevos paquetes de la máquina virtual con `sudo apt update`, instalamos Docker con `sudo apt install docker-ce`. Finalmente, verificamos que el servicio Docker funcionara correctamente con `sudo systemctl status docker`.

```
alejandro@alejandro-virtualbox:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; >
   Active: active (running) since Sun 2024-06-02 20:00:43 CEST; 12s>
   TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
   Main PID: 3183 (dockerd)
   Tasks: 9
   Memory: 29.8M (peak: 30.1M)
   CPU: 334ms
   CGroup: /system.slice/docker.service
           └─3183 /usr/bin/dockerd -H fd:// --containerd=/run/conta>
```

Ahora, ya habiendo comprobado que el servicio Docker está correctamente instalado en nuestra máquina, podemos comenzar con la instalación del servidor DHCP. Lo primero que haremos será crear, con el comando ``sudo mkdir``, un directorio para este servidor dándole el nombre que nosotros queramos. Por lo que el comando que usaremos en este caso será ``sudo mkdir Docker-dhcp``. Nos movemos a este directorio con el comando ``cd Docker-dhcp/``, y dentro de este directorio crearemos un archivo llamado ``Dockerfile`` que contendrá toda la información de nuestra imagen Docker DHCP, por lo que usando el comando ``sudo nano Dockerfile``.



```

#Usa una imagen base de Ubuntu
FROM ubuntu:latest

#Instala el servidor DHCP (isc-dhcp-server) y herramientas de red
RUN apt-get update && apt-get install -y isc-dhcp-server net-tools

#Copia el archivo de configuracion del servidor DHCP
COPY dhcpd.conf /etc/dhcp/dhcpd.conf

#Crea el directorio para la base de datos de arrendamientos
RUN mkdir -p /var/lib/dhcp/

#Crea manualmente el archivo de base de datos de arrendamientos
RUN touch /var/lib/dhcpd.leaves

#Establece permisos en el directorio
RUN chown -R dhcpd:dhcpd /var/lib/dhcp/

#Establece las opciones predeterminadas para el servidor DHCP
RUN echo 'INTERFACESv4="eth0" >> /etc/default/isc-dhcp-server'

#Puerto para escuchar peticiones DHCP
EXPOSE 67/udp

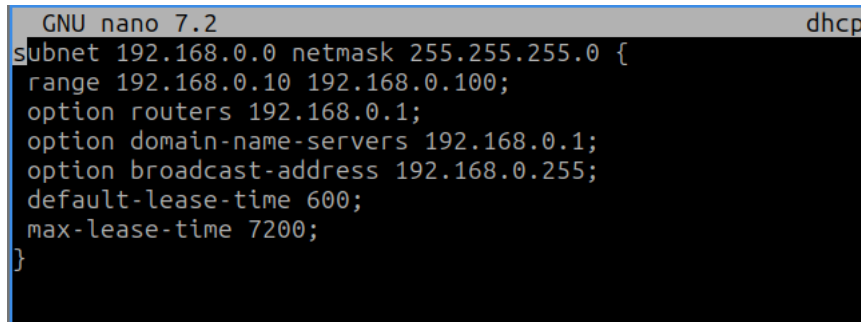
#Comando para iniciar el servidor DHCP
CMD ["/usr/sbin/dhcpd", "-f", "-d", "--no-pid"]

```

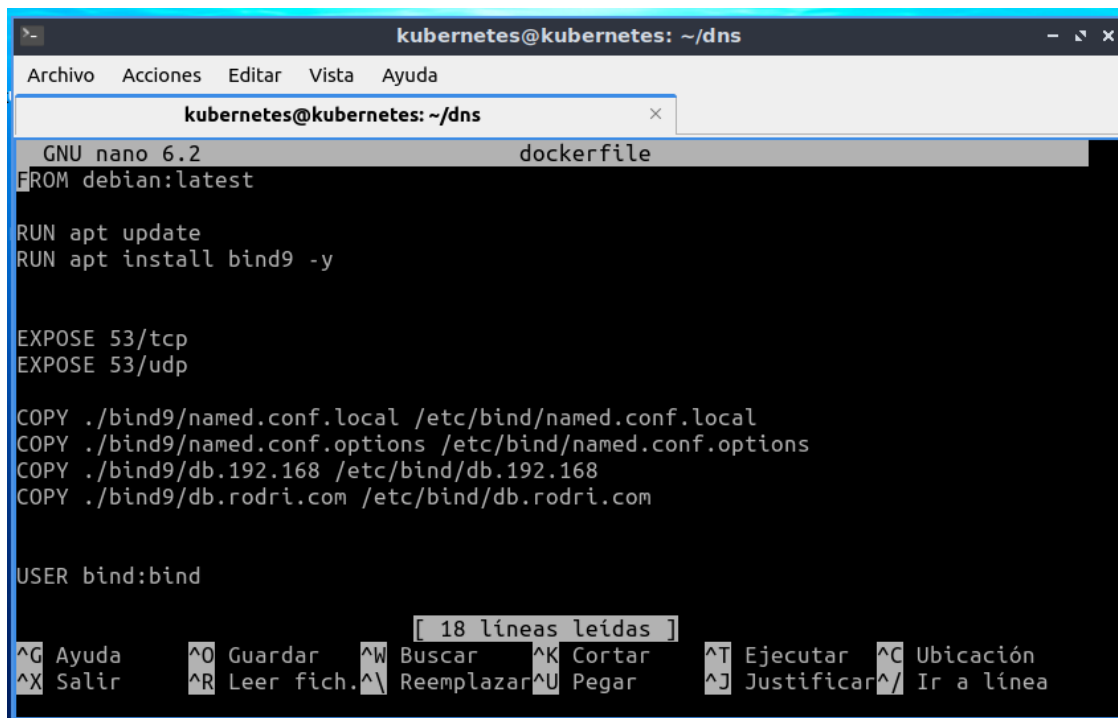
Como se puede ver en la imagen, dentro del archivo le indicamos que use una imagen base de Ubuntu. En la siguiente línea especificamos con el comando ``RUN apt-get update && apt-get install -y isc-dhcp-server net-tools``, que instale el servidor DHCP y las herramientas de red. A continuación, copiamos el archivo de configuración del servidor DHCP y creamos un directorio para la base de datos de los arrendamientos. Después de la creación de la base de datos, manualmente crearemos un archivo de base de datos para los arrendamientos, estableceremos los permisos en el directorio y configuraremos las opciones predeterminadas para nuestro servidor DHCP. De los últimos parámetros que introduciremos, será que exponga el puerto 67, ya que será el que usará el servidor DHCP para escuchar las peticiones. Por último, pero no menos importante, indicaremos que inicie nuestro servidor DHCP con el comando ``CMD ["/usr/sbin/dhcpd", "-4", "-f", "-d", "--no-pid"]``. Habiendo introducido este último comando, habremos terminado de editar nuestro archivo Dockerfile.

El siguiente paso en la creación de nuestra imagen Docker de DHCP será la creación del archivo ``dhcpd.conf``. Por lo que, haciendo uso del comando ``sudo nano dhcpd.conf``, crearemos el archivo y comenzaremos a editarlo. Dentro de este, tendremos que especificar las IPs que usará nuestro servidor DHCP. En nuestro caso, usamos las IP 192.168.0.0 como subnet y la 255.255.255.0 como máscara. Introduciremos que tenga un rango de IPs desde la 192.168.0.10 hasta la 192.168.0.100. La IP que tiene configurada nuestro router es la 192.168.0.1, junto con la de nuestro dominio que es la 192.168.0.1, es decir, la misma. Por último, la IP de nuestro

broadcast es la 192.168.0.255. Aunque hayamos de configurar las IPs, todavía nos queda por configurar los tiempos de cesión de nuestro servidor, que en nuestro caso, el predeterminado son 600 segundos, y el máximo son 7200 segundos. Por lo que nuestro archivo ``dhcpd.conf`` nos quedó de esta manera.

A screenshot of a terminal window with a dark background. At the top, it shows 'GNU nano 7.2' on the left and 'dhcpd.conf' on the right. The main content is a configuration block for a subnet: 'subnet 192.168.0.0 netmask 255.255.255.0 {' followed by several lines of options: 'range 192.168.0.10 192.168.0.100;', 'option routers 192.168.0.1;', 'option domain-name-servers 192.168.0.1;', 'option broadcast-address 192.168.0.255;', 'default-lease-time 600;', and 'max-lease-time 7200;'. The block ends with a closing brace '}'.

Después de haber creado tanto el archivo `Dockerfile` como el archivo `dhcpd.conf`, lo siguiente que tendremos que hacer será crear una red para nuestra IP. Por lo que tuvimos que introducir el siguiente comando: `sudo docker network create --subnet=192.168.0.0/24 Golden_Arrow_network`, este comando especifica la IP y máscara de nuestra subnet y el nombre de nuestra red. Una vez que hemos creado una red para nuestra IP, solo nos quedará montar el docker con el siguiente comando: `sudo docker build -t docker-dhcp .`, dado que `docker-dhcp` es el nombre de nuestra carpeta. Con el siguiente comando, podremos poner en marcha nuestra imagen docker: `sudo docker run --name servidor --net Golden_Arrow_network --ip 192.168.0.11 -d docker-dhcp`, y ahora solo nos queda subir la imagen docker. El primer comando que tendremos que usar con Docker ya instalado en la máquina virtual será `docker login` para conectarnos desde nuestra máquina virtual a nuestra cuenta de Docker Hub, si no tenemos una, es necesario crearla. Como ya hemos construido nuestra propia imagen Docker, tendremos que asegurarnos de que está construida correctamente usando el siguiente comando: `docker images`, con este comando, nos saldrán todas las imágenes construidas en nuestra máquina. Y, por último, lo que tendremos que hacer ya conectados a Docker Hub desde nuestra máquina virtual será, con el siguiente comando: `sudo docker tag docker-dhcp docker-dhcp:latest`, para etiquetar nuestra imagen. Luego, usaremos el siguiente comando para subir nuestra imagen a Docker Hub: `sudo docker push alejandro/docker-dhcp:latest`. Para nuestra siguiente imagen Docker, necesitamos seguir los pasos iniciales seguidos anteriormente para la creación de la imagen DHCP. Por lo que lo primero que tendremos que hacer será introducir los comandos `sudo apt update` y `sudo apt upgrade` para actualizar los nuevos paquetes e instalar las actualizaciones de esos paquetes en nuestra nueva máquina virtual. A continuación, volveremos a instalar todos los paquetes relacionados con Docker como lo hemos hecho anteriormente para nuestro servicio DHCP.



```
kubernetes@kubernetes: ~/dns
GNU nano 6.2 dockerfile
FROM debian:latest

RUN apt update
RUN apt install bind9 -y

EXPOSE 53/tcp
EXPOSE 53/udp

COPY ./bind9/named.conf.local /etc/bind/named.conf.local
COPY ./bind9/named.conf.options /etc/bind/named.conf.options
COPY ./bind9/db.192.168 /etc/bind/db.192.168
COPY ./bind9/db.rodri.com /etc/bind/db.rodri.com

USER bind:bind

[ 18 líneas leídas ]
^G Ayuda  ^O Guardar  ^W Buscar  ^K Cortar  ^T Ejecutar  ^C Ubicación
^X Salir   ^R Leer fich. ^_ Reemplazar ^U Pegar    ^J Justificar ^/ Ir a línea
```

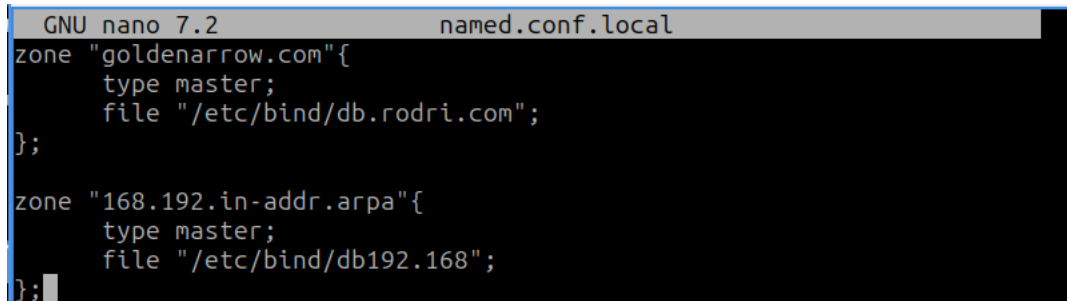
Le especificaremos la imagen base de nuestro servicio. En este caso, se trata de una imagen Debian. A continuación, ejecutará los comandos `apt update` para actualizar los programas del equipo y `apt install bind9 -y`, con el que se descargará Bind9 en el equipo sin necesidad de confirmación debido al `-y` al final de la línea de comando, ya que ese es su propósito. A continuación, especificaremos con los comandos `EXPOSE 53/tcp` y `EXPOSE 53/udp` que nuestro servicio, por los protocolos TCP y UDP, escuche por el puerto 53.

Siguiendo con la creación de nuestro archivo Dockerfile, introducimos los siguientes 4 comandos: Sudo nano dockerfile. Seguidio de los comandos: COPY ./bind9/named.conf.local /etc/bind/named.conf.local, COPY ./bind9/named.conf.options /etc/bind/named.conf.options, COPY ./bind9/db.192.168 /etc/bind/db.192.168, COPY ./bind9/db.rodri.com /etc/bind/db.rodri.com. Comandos bastante similares ya que su función es la misma: copiar los archivos desde el sistema de archivos del host al sistema de archivos del contenedor, cada uno de ellos desde diferentes directorios. Y por último, pero no menos importante, con la línea de código `USER bind:bind`, establece el usuario y grupo con los que serán ejecutados los comandos del contenedor, especificando que tanto el usuario como el grupo son bind.

Después de completar la configuración de nuestro archivo Dockerfile, dentro de nuestro directorio `dns`, crearemos otro directorio, en nuestro caso llamado `bind9`. En él, crearemos a su vez otros cuatro archivos de configuración en los que se especificarán los diferentes parámetros de nuestro servicio DNS.

El primer archivo de configuración que crearemos será el `named.conf.local`. Por lo que con el comando `sudo nano named.conf.local`, lo creamos y comenzamos a editarlo. En este especificaremos las zonas de nuestro servicio DNS, siendo una `GoldenArrow.com`. Con la línea de texto `zone "GoldenArrow.com"`, a continuación, abrimos corchete en el archivo y dentro de este comentamos lo siguiente: `type: master;`, con lo que especificamos que nuestro servidor es el servidor principal de esta zona. Con la línea de texto `file "/etc/bind/db.rodri.com";`, especificamos el archivo que contiene los registros de nuestro servidor DNS para la zona `GoldenArrow.com`. Y, por último, cerramos corchetes y ponemos punto y coma para no dar lugar a errores.

La siguiente zona seguirá el mismo orden de creación que la anterior. En este caso, su nombre será `168.192.in-addr.arpa` y se referirá a una zona de búsqueda inversa para la red. Con la línea de texto `type master`, especificamos que su servidor principal es el nuestro, y con la línea de texto `file "/etc/bind/db.192.168";`, especificamos el archivo que contiene los registros de búsqueda inversa. Por lo que debería quedar de esta manera..



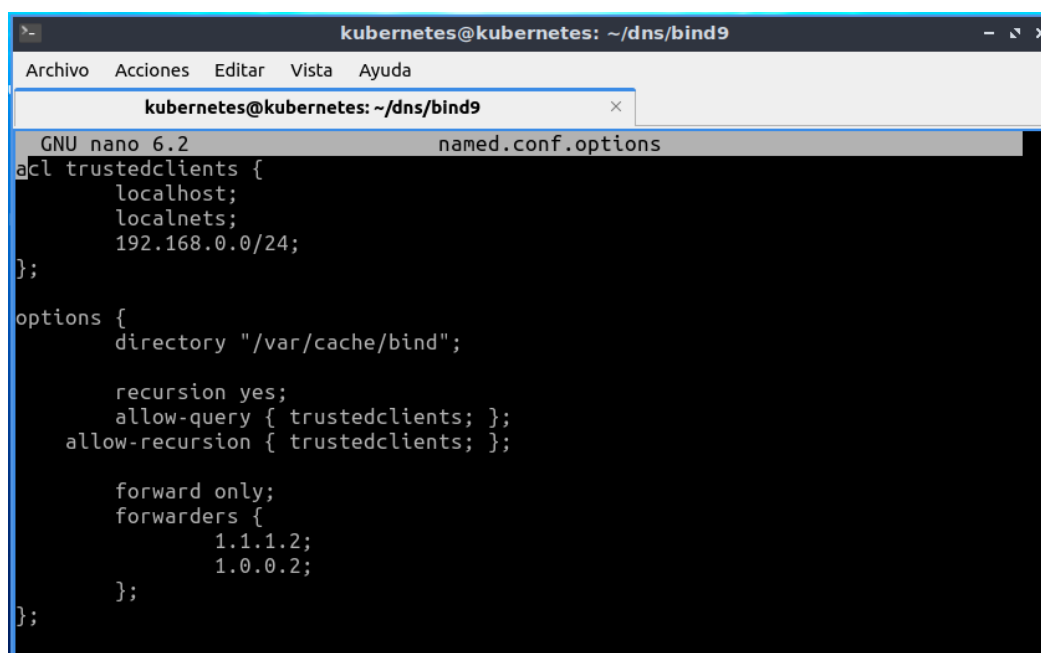
```
GNU nano 7.2      named.conf.local
zone "goldenarrow.com"{
    type master;
    file "/etc/bind/db.rodri.com";
};

zone "168.192.in-addr.arpa"{
    type master;
    file "/etc/bind/db192.168";
};
```

El siguiente archivo de configuración que crearemos será el `named.conf.options`. Por lo que, usando el comando `sudo nano named.conf.options`, creamos el archivo y lo empezamos a editar. Lo primero que haremos es definir una lista de control de acceso a la que llamamos `trustedclients`, en la que se encontrarán tanto el localhost como la localnets y la IP de nuestro servidor, es decir, la `192.168.0.0/24`.

A continuación, especificaremos las opciones de nuestro servidor DNS. Lo primero es especificar el directorio donde se almacena el caché del servidor DNS. A continuación, permitimos al servidor DNS la búsqueda recursiva. Continuando con la edición del archivo, restringiremos las consultas recursivas y comunes hacia los clientes de la lista de control de acceso. Indicamos que el servidor DNS solo debe reenviar consultas a otros servidores, ya que él no las puede responder directamente.

Lo siguiente que tendremos que introducir será una lista de servidores DNS a los que se puede reenviar las consultas. Donde especificaremos tanto el servidor `1.1.1.2` como el `1.0.0.2` y cerramos la lista de servidores que pueden recibir consultas. Por lo que nuestro archivo `named.conf.options` debería quedar así.



```
kubernetes@kubernetes: ~/dns/bind9
GNU nano 6.2      named.conf.options
acl trustedclients {
    localhost;
    localnets;
    192.168.0.0/24;
};

options {
    directory "/var/cache/bind";

    recursion yes;
    allow-query { trustedclients; };
    allow-recursion { trustedclients; };

    forward only;
    forwarders {
        1.1.1.2;
        1.0.0.2;
    };
};
```

El siguiente archivo de configuración que crearemos será el ``db.goldenarrow.com``. Haciendo uso del comando ``sudo nano db.goldenarrow.com``, creamos el archivo y comenzamos a editarlo. Lo primero que indicaremos es que funcionará durante las 24 horas del día.

A continuación, tendremos que especificar todas las entradas de nuestro servidor DNS que pertenecen al dominio ``goldenarrow.com``. Después, junto con el DNS autorizado y el administrador del dominio, definiremos el registro SOA. Continuando con el SOA, especificaremos el número de serie del registro SOA, en nuestro caso 1. También tendremos que especificar el intervalo con el que el servidor DNS se actualizará, en nuestro caso diariamente, es decir, cada 24 horas, así como el tiempo para reintento de actualización por si la inicial llegase a fallar, en este caso, le daremos un tiempo de 2 horas. Especificaremos también el límite de tiempo que el servidor DNS conservará el registro SOA en la caché antes de eliminarlo, que serán 1000 horas, así como el mínimo tiempo de vida del mismo registro SOA almacenado en la caché antes de su eliminación, que será de dos días. Con esto, damos por finalizada la configuración del SOA para nuestro servidor DNS.

A continuación, definiremos el registro NS para nuestro dominio, así como el único servidor autorizado de nuestro servidor DNS, que en nuestro caso es el ``testdns.goldenarrow.com``. Por último, tendremos que especificar los nombres de los host que se asocian con las direcciones IP dentro de nuestra zona, es decir, ``goldenarrow.com``. Estos serán ``goldenarrow1`` asociado con la dirección IP ``192.168.0.10``, ``goldenarrow2`` asociado con la dirección IP ``192.168.0.11``, ``goldenarrow3`` asociado con la dirección ``192.168.0.12``, y por último, ``testdns1.goldenarrow.com`` asociado con la dirección IP ``192.168.0.30``. Por lo que nuestro archivo ``db.goldenarrow.com`` nos queda de esta manera.

```
GNU nano 7.2 db.goldenarrow.com
$TTL 24h
$ORIGIN goldenarrow.com.
@      IN      SOA      testdns.goldenarrow.com. admin.goldenarrow.com. (
                                1          ; serial number
                                24h         ; refresh
                                2h          ; update retry
                                1000h        ; expire
                                2d          ; minimum
                                )

                                IN      NS      testdns1.goldenarrow.com.

goldenarrow1  IN      A      192.168.0.10
goldenarrow2  IN      A      192.168.0.11
goldenarrow3  IN      A      192.168.0.12
testdns1      IN      A      192.168.0.30
dockerdemo   IN      A      192.168.0.30
pbs           IN      A      192.168.0.31
```

Por último pero no menos importante, haciendo uso una vez más del comando ``sudo nano db.192.68``, creamos nuestro cuarto y último archivo de configuración. Lo primero que debemos indicar es que todas las líneas del archivo pertenecen al dominio ``goldenarrow.com``. A continuación, deberemos definir los distintos parámetros del registro SOA para el dominio ``goldenarrow.com``, es decir, nuestro dominio.

Lo primero que debemos especificar será tanto el servidor DNS autorizado para nuestro dominio, que en nuestro caso se trata de ``testdns.goldenarrow.com``, así como el administrador del dominio, que es ``admin.goldenarrow.com``. A continuación, deberemos el número de serie de nuestro registro SOA, en este caso siendo 1, así como el tiempo de actualización de nuestro

registro SOA, que especificaremos que se actualice cada día, junto con el tiempo de reintento de actualización si no se ha podido llevar a cabo inicialmente, lo que pondremos que sea cada 2 horas. A continuación, especificaremos el tiempo de expiración durante el cual nuestro servidor DNS conservará el registro SOA en la caché antes de eliminarlo, en nuestro caso 1000 horas, así como el tiempo mínimo de vida del mismo, que en nuestro caso especificamos que es de 2 días, es decir, 48 horas. Con esto, acabamos de editar los distintos parámetros del registro SOA para nuestro servidor DNS.

A continuación, definiremos el nombre del registro NameServer para nuestro servidor DNS, que en este caso será `testdns.goldenarrow.com`. Por último, tendremos que especificar las IPs relacionadas con los distintos nombres de host de nuestro servidor DNS para poder efectuar correctamente el registro inverso de las mismas. Iniciando el listado de las distintas direcciones IP, la IP `192.168.0.10` está relacionada con el host `goldenarrow1.goldenarrow.com`, la IP `192.168.0.11` está relacionada con el host `goldenarrow2.goldenarrow.com`, la IP `192.168.0.12` está relacionada con el host `goldenarrow3.goldenarrow.com`, la IP `192.168.0.30` está relacionada con el host `dockerdemo.goldenarrow.com`, y por último pero no menos importante, la IP `192.168.0.31` está relacionada con el host `pbs.goldenarrow.com`. Por lo que nuestro archivo `db.192.168` nos queda una vez completado de esta manera.

```
GNU nano 7.2 db.192.168
TTL 24h
$ORIGIN 168.192.in-addr.arpa.
@      IN      SOA      testdns1.goldenarrow.com. admin.goldenarrow.com. (
                                1          ; serial number
                                24h         ; refresh
                                2h          ; update retry
                                1000h       ; expire
                                2d          ; minimum
                                )
                                IN      NS      testdns1.goldenarrow.com.

$ORIGIN 0.168.192.in-addr.arpa.
10     IN      A       goldenarrow1.goldenarrow.com.
11     IN      A       goldenarrow2.goldenarrow.com.
12     IN      A       goldenarrow3.goldenarrow.com.
30     IN      A       dockerdemo.goldenarrow.com.
31     IN      A       pbs.goldenarrow.com.
```

Una vez completados todos los archivos de configuración, lo siguiente que tendremos que hacer será crear nuestro Docker usando el comando `sudo docker build -t goldenarrow-dns .`. A continuación, haciendo uso del comando `sudo network create --subnet=192.168.0.0/24 goldenarrownet`, crearemos y nombraremos una red para nuestro contenedor Docker. Luego, tendremos que arrancar nuestro contenedor con el comando `sudo docker run --name goldenarrowdns --net goldenarrownet --ip 192.168.0.11 --dns 192.168.0.10 -d goldenarrow-dns`, con el que, aparte de arrancar nuestro contenedor junto con los archivos de configuración creados anteriormente, asignaremos las IPs tanto al contenedor como al host del servidor.

Una vez realizado todo el proceso de creación de la imagen Docker, todo lo que resta es subir nuestra propia imagen Docker a Docker Hub. El primer comando que tendremos que usar será `docker login`, para poder acceder desde nuestro terminal a la página de Docker Hub, lo que nos pedirá tanto nuestro nombre de usuario como nuestra contraseña. Una vez conectados correctamente a Docker Hub, en este tendremos que crear un nuevo repositorio para poder alojar nuestra nueva imagen Docker. Haciendo uso del comando `sudo docker push goldenarrow-dns`, podremos subir desde el terminal de nuestra máquina virtual la imagen Docker creada.

Partiendo de cero nuevamente, ya que comenzamos con la creación de nuestra tercera y última imagen Docker, nos dimos cuenta de que la imagen Nginx era mucho más manejable y compatible con nuestro proyecto que la imagen Docker con la que pensamos inicialmente, por lo que decidimos cambiarnos a esta. Ahora, comenzando con la instalación de nuestra imagen Nginx, lo primero será introducir los comandos ``sudo apt update`` y ``sudo apt upgrade``, para así, una vez más, actualizar los paquetes de nuestra máquina e instalar las actualizaciones de estos paquetes, para poder tener la máquina en un estado óptimo.

A continuación, tendremos que volver a bajarnos todas las dependencias necesarias de Docker, así como sus complementos, con los comandos:

- ``sudo apt install apt-transport-https ca-certificates curl software-properties-common``, para la instalación de las dependencias necesarias de Docker.

- Usando el comando ``curl -fsSL https://download.docker.co/linux/ubuntu/gpg | sudo apt-key add -``, añadiremos la clave GPG.

- A continuación, haciendo uso del comando ``sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"``, añadiremos el repositorio Docker.

- Luego, llevamos a cabo de nuevo el comando ``sudo apt update``, para actualizar los nuevos paquetes de nuestra máquina virtual.

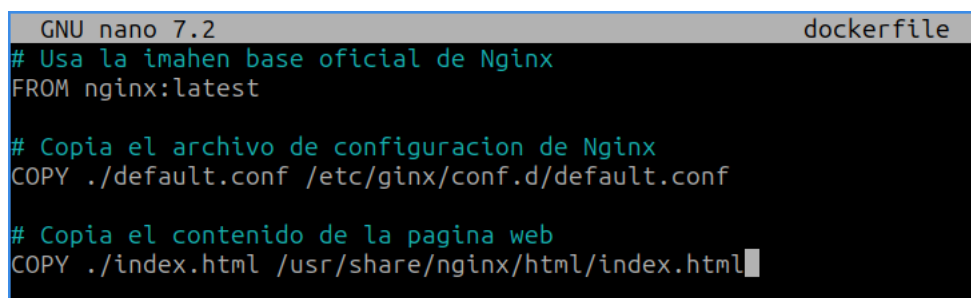
- Y ahora, con todos los programas anteriores instalados, nos volveremos a instalar Docker con el comando ``sudo apt install docker-ce``.

Comenzando con la creación de nuestra imagen Nginx, lo primero que tendremos que hacer será crear un nuevo directorio haciendo uso del comando ``sudo mkdir nginx``, ya que en nuestro caso el directorio se llama ``nginx``. Con el comando ``cd nginx/``, nos movemos dentro del directorio y dentro de este tendremos que crear tres archivos. El primero que creamos fue el archivo Dockerfile, y lo completamos con las siguientes especificaciones:

- Usamos la imagen oficial de Nginx con el comando ``FROM nginx:latest``.

- Copiamos el archivo de configuración de Nginx con el comando ``COPY ./default.conf /etc/nginx/conf.d/default.conf``.

- Y por último, copiamos el contenido de la página web usando el comando ``COPY ./index.html /usr/share/nginx/html/index.html``. Por lo que nuestro archivo Dockerfile quedará de esta manera.

A screenshot of a terminal window with a dark background. The title bar at the top shows "GNU nano 7.2" on the left and "dockerfile" on the right. The terminal content shows three lines of code in a light blue font: "# Usa la imagen base oficial de Nginx" followed by "FROM nginx:latest", "# Copia el archivo de configuracion de Nginx" followed by "COPY ./default.conf /etc/nginx/conf.d/default.conf", and "# Copia el contenido de la pagina web" followed by "COPY ./index.html /usr/share/nginx/html/index.html". The cursor is at the end of the last line.

```
GNU nano 7.2                                dockerfile
# Usa la imagen base oficial de Nginx
FROM nginx:latest

# Copia el archivo de configuracion de Nginx
COPY ./default.conf /etc/nginx/conf.d/default.conf

# Copia el contenido de la pagina web
COPY ./index.html /usr/share/nginx/html/index.html
```

A continuación, creamos nuestro siguiente archivo de configuración usando el comando ``sudo nano default.conf``, en el que especificaremos los distintos aspectos de nuestro servidor. Incluiremos el puerto por el que escucha, que es el 80, así como el nombre del mismo, que es ``goldenarrow3.goldenarrow.com``. También especificaremos la ubicación de nuestra página web en nuestra máquina virtual, la cual estará localizada en el directorio base ``/usr/share/nginx/html``, y el nombre de la misma, que en nuestro caso se llama ``index.html``.


```

GNU nano 7.2                                     default.conf
server {
    listen 80;
    server_name goldenarrow3.goldenarrow.com;

    location / {
        root /usr/share/nginx/html;
        index index.html;
    }
}

```

Por último, tendremos que crear el archivo destinado a ser nuestra página web. Haciendo uso del comando `sudo nano index.html`, lo crearemos y editaremos de manera que, una vez accedamos a la misma, tenga el nombre de "GoldenArrow3" y muestre el siguiente texto: Welcome to goldenarrow3.goldenarrow.com y ESTA PAGINA ESTA EN LA IP 192.168.0.12 y con esto, la estructura de nuestra página web quedaría definida.

```

GNU nano 7.2                                     index.html
<!DOCTYPE html>
<html>
<head>
    <title>GoldeArrow3</title>
</head>
<body>
    <h1>Welcome to goldenarrow3.goldenarrow.com</h1>
    <p>ESTA PAGINA ESTA EN LA IP 192.168.0.12</p>
</body>
</html>

```

Una vez creados los archivos anteriores, tendremos que construir y nombrar nuestra imagen Docker con el comando ``sudo docker build -t goldenarrownginx .``. Luego, creamos una red para nuestra imagen Docker en modo puente (bridge) con el comando ``sudo docker network create --subnet=192.168.0.0/24 --driver bridge goldenarrownet``. Procederemos a lanzar nuestra imagen Docker con el comando ``sudo docker run --name nginxgoldenarrow --net goldenarrownet --ip 192.168.0.0 --dns 192.168.0.10 -d goldenarrownginx``.

Finalmente, solo nos quedará subir la imagen a Docker Hub. Siguiendo los mismos pasos que en las dos anteriores veces, lo haremos sin problemas. Lo primero es conectarnos a Docker Hub mediante el comando ``docker login``, que nos pedirá tanto nuestro nombre de usuario como nuestra contraseña. Luego, tendremos que crear un nuevo repositorio para poder alojar nuestra nueva imagen Docker. Finalmente, haciendo uso del comando ``sudo docker push goldenarrow-dns``, subiremos desde el terminal de nuestra máquina virtual la imagen Docker creada.

Al mismo tiempo que se creaban las imágenes Docker necesarias para el trabajo, estábamos llevando a cabo la creación del clúster de Kubernetes. Lo primero que hicimos fue crear la estructura base que queríamos que tuviera nuestro clúster. Después de largas deliberaciones y distintos tipos de pruebas, concluimos que nuestro clúster estaría compuesto por dos Kuproxy, cuyos roles serían de HAProxy, junto con tres Kumaster, cuya función sería de maestros, y por último, dos equipos trabajadores.

Antes de comenzar con la creación del clúster de Kubernetes, necesitamos preparar los equipos que vayamos a usar en nuestro clúster, instalando en estos los parámetros de configuración HAProxy así como los parámetros Keepalived para que nuestro clúster de Kubernetes tenga una alta disponibilidad. Comenzamos con la instalación de HAProxy y Keepalived en los equipos

introduciendo el comando `sudo apt install keepalived haproxy psmisc -y`. A continuación, empezaremos a editar el archivo de configuración de HAProxy haciendo uso del comando `sudo vi /etc/haproxy/haproxy.cfg`, y en este añadimos los siguientes parámetros:

- Definimos la configuración del frontend, que escucha en el puerto 6443, con la línea de texto `frontend kubernetes-frontend`.
- Indicamos que el HAProxy escuchará con la interfaz 192.168.1.10 en el puerto 6443, con `bind *:6443`.
- Establecemos el modo TCP para manejar el tráfico, con `mode tcp`.
- También habilitamos el registro TCP, con `option tcplog`.
- A continuación, definimos el backend por defecto, con `default_backend kubernetes-backend`, y configuramos el backend para distribuir las solicitudes entre los servidores de Kubernetes.
- Indicamos que utilizamos el algoritmo de balanceo de carga round-robin con `balance roundrobin`.
- Definimos las configuraciones por defecto para los servidores backend con `default-server`. En esta misma línea de código especificaremos el intervalo de tiempo entre las comprobaciones de salud del servidor que será de 10 segundos, el intervalo de tiempo entre las comprobaciones de salud cuando un servidor está caído, que en este caso serán cada 5 segundos, así como el número de comprobaciones que se deberán llevar a cabo con éxito para poder marcar un servidor como activo en este caso 2, y el número de veces que deberán fallar las comprobaciones para poder marcar un servidor como inactivo, en este caso 3.
- A su vez, indicaremos el tiempo durante el que el tráfico dirigido hacia el servidor se vuelve más activo, en este caso 60s.
- También indicaremos el número máximo de conexiones que podrá tener al mismo tiempo el servidor, 250, y el número máximo de solicitudes que el servidor podrá tener en cola, 256.
- Y por último, también especificaremos el peso del servidor en el balanceo de carga, que el de nuestro servidor será 100.
- Definimos un servidor backend, cuyo nombre será `kumaster1`, que tendrá la IP 172.20.10.20 y el puerto :6443, con `server kumaster1 172.20.10.20:6443 check`. También habilitaremos la opción `check` para ver el estado de salud del servidor.
- Definimos otro servidor backend, cuyo nombre será `kumaster2`, que tendrá la IP 172.20.10.21 y el puerto :6443. También habilitaremos la opción `check` para ver el estado de salud del servidor, con `server kumaster2 172.20.10.21:6443 check`.
- Definimos otro servidor backend, cuyo nombre será `kumaster3`, que tendrá la IP 172.20.10.22 y el puerto :6443. A su vez, habilitaremos la opción `check` para ver el estado de salud del servidor, con `server kumaster3 172.20.10.22:6443 check`.
- Las líneas comentadas en las que se encuentran los servidores `kuproxy0` y `kuproxy2` son ejemplos de servidores adicionales que podrían ser habilitados de ser necesario.
- Y por último, definiremos la sección `peers`, con `peers mypeers`. Lo primero que haremos será definir una sección de `peers` llamada `mypeers` destinada a la replicación de estados entre múltiples instancias de HAProxy.
- Y por último, definimos dos `peers` más: un `peer` llamado `lb0`, con la IP

Por lo que la estructura quedaria asi:

```
frontend kubernetes-frontend
    bind      *:6443
    mode      tcp
    option    tcplog
    default_backend kubernetes-backend

backend kubernetes-backend
    mode      tcp
    option    tcplog
    option    tcp-check
    balance   roundrobin
    default-server inter 10s downinter 5s rise 2 fall 3 slowstart 60s maxconn 250 maxqueue 256 weight 100
    server kumaster1 192.168.1.20:6443 check
    server kumaster2 192.168.1.21:6443 check
    server kumaster3 192.168.1.22:6443 check
    #server kuproxy0 192.168.1.10:6443 check
    #server kuproxy2 192.168.1.12:6443 check

peers mypeers
    peer lb0 192.168.1.10:1024
    peer lb2 192.168.1.12:1024
```

El siguiente paso es reiniciar el servicio HAProxy con el comando `sudo systemctl restart haproxy`, y con el comando `sudo systemctl enable haproxy`, lo volveremos a habilitar. Una vez hecho esto, comenzaremos a editar el archivo de configuración de Keepalived, que se utiliza para gestionar la alta disponibilidad mediante la creación de una IP virtual que puede ser asumida por cualquier nodo en caso de que uno falle. Por lo que usando el comando `vi /etc/keepalived/keepalived.conf`, comenzaremos a editarlo. Lo primero será definir la sección que contendrá los parámetros globales de Keepalived, con `global_defs`. Dentro de estos parámetros, crearemos una lista de correos electrónicos para enviar correos electrónicos con `notification_email`. Continuamos creando el identificador del enrutador cuya ID debe ser única para cada instancia, con `router id LVS_DEVEL`. Añadimos una opción para saltar la verificación de la dirección de anuncio VRRP con `vrrp_skip_check_adv_addr`. Creamos el intervalo para enviar anuncios gratuitos ARP o GARP, como no queremos que no se envíen ninguno le ponemos un valor de 0 con `vrrp_garp_interval 0`, y con esto terminamos con los parámetros globales para el Keepalived. A continuación, creamos un script para comprobar el VRRP, con `vrrp_script chk_haproxy`. Definimos el script con el que comprobaremos el estado del servicio HAProxy. Dentro de este script añadiremos el comando script “killall -o haproxy”, que verificará si el HAProxy está ejecutándose sin llegar a matarlo. Marcaremos el intervalo con el que queremos que se ejecute el script anterior con `interval 2`, en nuestro caso cada dos segundos. Y por último, asignamos un peso al script si este tiene éxito con `weight 2`, en nuestro caso especificamos un peso de dos. Por lo que hemos terminado de crear el script para la comprobación de VRRP. Lo siguiente que haremos será crear una instancia VRRP, por lo que con el comando `vrrp_instance haproxy-vip`, definimos una instancia VRRP llamada `haproxy-vip`. Definiremos el estado inicial como BACKUP con `state BACKUP`, para indicar que este será el nodo de respaldo. Definiremos la prioridad del nodo con `priority 100`, ya que el nodo con más prioridad se convertirá en el maestro. Definiremos la interfaz de red que usaremos para VRRP con `interface ens33`, en este caso utilizamos la interfaz `ens33`, ya que será la interfaz que se conecte con las otras máquinas. Con el comando `virtual_router_id 60`, especificamos el

enrutador virtual, el cual debe ser el mismo en todas las instancias que participan en el mismo grupo VRRP. Definiremos con qué frecuencia los anuncios VRRP son enviados con `advert_int 1`, y con `authentication`, comenzaremos a configurar la configuración de VRRP. Definiremos el tipo de autenticación con `auth_type PASS`, en este caso de contraseña simple, y definiremos la contraseña usada para la autenticación con `auth_pass 1111`, en nuestro caso la contraseña es una secuencia numérica de cuatro veces el número 1 repetido. Y con esto terminamos con la configuración de la autenticación de VRRP. A continuación, definimos la IP de origen para los mensajes unicast VRRP con `unicast_src_ip 172.20.10.10`, junto con una lista de las direcciones IP de los peers unicast con `unicast_peer`, que definiremos que esta lista recoja las IP de otros nodos VRRP en nuestro grupo es decir, 172.20.10.11, y 172.20.10.12. Y con esto habremos completado la lista de direcciones de las IP de los peers unicast. Y por último, con el comando `virtual_ipaddress`, podremos definir la IP virtual que será gestionada por esta instancia, que en nuestro servicio la IP gestionada por esta misma instancia, y la que será utilizada para acceder a los servicios gestionados por HAProxy, se trata de la 172.20.10.9/28, con su máscara subred. A continuación, lo que tendremos que hacer será habilitar el servicio Keepalived con el comando `sudo systemctl enable keepalived` y habremos acabado de preparar nuestro equipo de cara a la creación de un clúster en estos.

```
GNU nano 6.2 /etc/keepalived/keepalived.conf *
global_defs {
    notification_email {
    }
    router_id LVS_DEVEL
    vrrp_skip_check_adv_addr
    vrrp_garp_interval 0
    vrrp_gna_interval 0
}

vrrp_script chk_haproxy {
    script "killall -0 haproxy"
    interval 2
    weight 2
}

vrrp_instance haproxy-vip {
    state BACKUP
    priority 100
    interface tun0
    virtual_router_id 60
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    unicast_src_ip 192.168.1.11
    unicast_peer {
        192.168.1.10
        192.168.1.12
    }

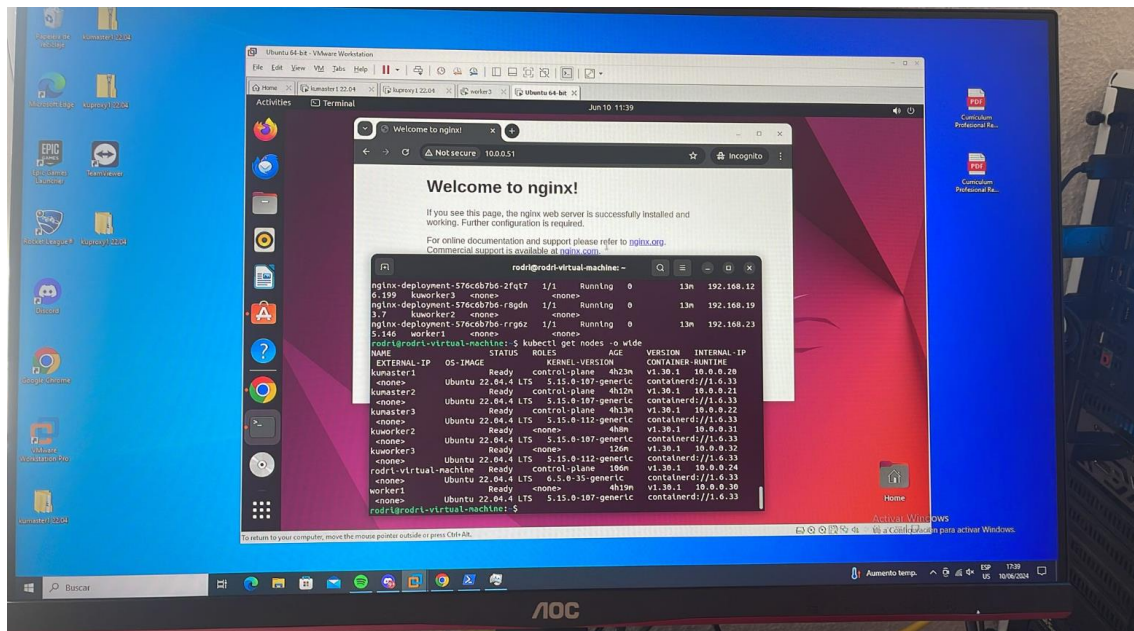
    virtual_ipaddress {
        192.168.1.9/24
    }
}
```

Pasos seguidos a la hora de crear el clúster, lo primero es introducir los comandos ``sudo apt update``, para actualizar todos los nodos de la máquina virtual, y ``sudo apt upgrade``, instalar las versiones más recientes de los paquetes instalados. El siguiente paso es desactivar el uso de la memoria swap, para asegurar un buen funcionamiento por parte de Kubernetes, por lo que usando los comandos ``sudo swapoff -a``, para desactivar el uso de todas las áreas swap en nuestra máquina, y el comando ``sudo sed -i 's/ swap / s/^(.*)$/#\1/g' /etc/fstab``, comentamos las líneas del archivo ``/etc/fstab``, que contienen información de swap, evitando así que el swap se active automáticamente al reiniciar el sistema. A continuación añadimos los parámetros del

kernel para todos los nodos, para esto tendremos que utilizar los comandos ``sudo tee /etc/module-load.d/containerd.conf <<EOF overlay br_netlifter EOF``, que crea un nuevo archivo de configuración y le introduce los módulos, overlay y br_netlifter, ya que son esenciales para el funcionamiento de Kubernetes, junto con el comando ``sudo modprobe overlay``, cargar el módulo overlay en el kernel, y el comando ``sudo modprobe br_filter``, para cargar el módulo br_netfilter en el kernel, seguido de la creación de los módulos, tendremos que configurar los parámetros del kernel con el comando, ``sudo tee /etc/sysctl.d/kubernetes.conf <<EOF net.bridge.bridge-nf-call-ip6tables = 1 net.bridge.bridge-nf-call-iptables = 1 net.ipv4.ip_forward = 1 EOF``, con el que creamos un archivo de configuración en el que estarán los parámetros del kernel necesarios para el uso de Kubernetes, y el comando, ``sudo sysctl --system``, que recarga todas las configuraciones de sysctl para que los cambios que realizamos tengan un efecto inmediato. Ahora comenzamos con la instalación del Containerd Runtime en todos los nodos, que se trata de un entorno de ejecución de contenedores necesario para el uso de Kubernetes. Lo primero que tendremos que hacer para esto será introducir el comando ``sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates``, ya que este instala las herramientas necesarias para añadir repositorios externos y gestionar certificados. A su vez debemos habilitar el repositorio de Docker, proceso que llevaremos a cabo introduciendo los siguientes comandos, ``sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/docker.gpg``, la función de este comando es la descarga y el resguardo de la clave GPG para nuestro repositorio Docker, mientras que el comando ``sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"``, añade el repositorio Docker a las fuentes APT, continuando con la instalación del Containerd Runtime, tendremos que introducir el comando ``sudo apt update``, para actualizar los paquetes disponibles para nuestra máquina y virtual, y a su vez introducir el comando ``sudo apt install -y containerd.io``, que instala el entorno de ejecución de contenedores en la máquina. Debemos configurar Containerd para que se use systemd como controlador de grupos de control o el cgroup, proceso que llevamos a cabo introduciendo los comandos ``containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1``, que generará una configuración predeterminada para el Containerd, y ``sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml``, que modifica la configuración para que el Containerd utilice systemd como cgroup. Por último deberemos reiniciar el Containerd usando el comando ``sudo systemctl restart containerd``, y volver a habilitarlo con el comando ``sudo systemctl enable containerd``. El siguiente paso que debemos llevar a cabo es añadir el repositorio de Kubernetes para todos los nodos, por lo que lo primero que tendremos que hacer será añadir los repositorios necesarios para la instalación de Kubernetes con los comandos, ``curl -fsSL https://pkgs.k8s.io/core:stable/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg``, que descarga y guarda la clave GPG para el repositorio de Kubernetes, y ``echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:stable/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list``, que añade el repositorio de Kubernetes a las fuentes de APT, siguiendo con la instalación en este paso tendremos que instalar tanto kubectl, como kubeadm y kubelet para todos los nodos, pero antes deberemos actualizar los paquetes de nuestra máquina con el comando ``sudo apt update``, a continuación podemos introducir el comando ``sudo apt install -y kubelet kubeadm kubectl``, para instalar tanto kubelet, como kubeadm, y kubectl, junto con el comando ``sudo apt-mark hold kubelet kubeadm kubectl``, para evitar que los paquetes que acabamos de instalar se actualicen de forma automática, y así asegurarnos la estabilidad de la versión que tenemos instalada. Los comandos utilizados hasta

este momento los hemos usado para la creación de una máquina kubernetes, que tenga HAProxy y Keepalived, lo siguiente que tendremos que hacer es comenzar con la preparación y configuración de una máquina master, por lo que haciendo uso de los comandos usados con anterioridad, tales como ``sudo apt update``, ``sudo apt upgrade``, para actualizar nuestra máquina, ``sudo swapoff -a`` y ``sudo sed -i 's/swap / s/^(.*)$/#/1/g' /etc/fstab``, desactivaremos el swap y definiremos los parámetros esenciales del kernel, cargamos los módulos de kernel en todos los módulos con ``sudo tee /etc/modules-load.d/containerd.conf <<EOF``, y configuramos sus parámetros con ``sudo tee /etc/sysctl.d/kubernetes.conf <<EOF``, y cargamos los cambios con ``sudo sysctl --system``, instalamos Containerd y sus dependencias con ``sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates``, y con ``sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/docker.gpg``, habilitaremos el repositorio Docker necesario, también tendremos que volver a actualizar nuestra máquina con ``sudo apt update``, e instalar Containerd con ``sudo apt install -y containers.io``, por último configuraremos Containerd para que use systemd como cgroup con ``containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1 sudo sed -i 's/SystemdCgroup |= false/SystemdCgroup |= true/g' /etc/containerd/config.toml``, ahora añadimos repositorios apt para Kubernetes en todos los nodos con ``curl -fsSL https://pkgs.k8s.io/core:stable/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg``, y haciendo uso del comando ``echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:stable/v1.30/deb/ ' | sudo tee /etc/apt/sources.list.d/kubernetes.list``, instalamos los componentes esenciales de Kubernetes, después de actualizar nuevamente nuestra máquina con los comandos ``sudo apt update``, ``sudo apt install -y kubelet kubeadm kubectl``, y ``sudo apt-mark hold kubelet kubeadm kubectl``, a continuación iniciaremos el Control Panel de nuestro clúster de Kubernetes introduciendo el siguiente comando, ``kubeadm init --control-plane-endpoint 172.20.10.9:6443 --upload-certs``, que inicializa el clúster de Kubernetes especificando la dirección del endpoint del Control Plane y sube los certificados necesarios, una vez inicializado el Control Plane deberemos comenzar con la configuración del kubectl para poder interactuar con el clúster más cómodamente, para ello usamos los comandos, ``sudo mkdir -p $HOME/.kube``, para crear el directorio .kube en el equipo del usuario actual, ``sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config``, que copiará el archivo de configuración de administración de Kubernetes al directorio creado con el anterior comando, en este caso .kube, y ``sudo chown $(id -u):$(id -g) $HOME/.kube/config``, para cambiar el propietario del archivo de configuración al usuario actual, por último tendremos que unir otros nodos a nuestro clúster, será necesario unir tanto nodos de trabajo como de control, por lo que para unir estos nodos usaremos el comando ``kubeadm join``, solo que específico para cada tipo de nodo, por lo que usando, ``kubeadm join 192.168.1.9:6443 --token yiu5lw.vlf1royrxc255ehi \ --discovery-token-ca-cert-hash sha256:022c50bdd812e69468cb6505f5637fd8f3afceffdece62f9e692f4dba0e2e424 \ --control-plane --certificate-key 54412941d589bb2c19f89901943ca2e34123d1d5a063264d4cc976f50c1ea6cb``, y para los nodos de trabajo usaremos el comando, y ``kubeadm join 192.168.1.9:6443 --token yiu5lw.vlf1royrxc255ehi \ --discovery-token-ca-cert-hash sha256:022c50bdd812e69468cb6505f5637fd8f3afceffdece62f9e692f4dba0e2e424``, a su vez usando el comando ``kubeadm join 192.168.1.9:6443 --token yiu5lw.vlf1royrxc255ehi \ --discovery-token-ca-cert-hash sha256:022c50bdd812e69468cb6505f5637fd8f3afceffdece62f9e692f4dba0e2e424`` en los equipos kubernetes previamente instalados y configurados, los uniremos a los equipos master ya

que serán los responsables de estos y una vez unidos nuestros equipos podremos continuar con la instalación de Kubernetes, lo siguiente que haremos será instalar el Plugin de red Kubernetes en el nodo maestro, para así habilitar la comunicación con el resto de nodos en el clúster con `kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml`, por último tendremos que comprobar si nuestro clúster se ha creado correctamente.



Una vez acabada la instalación y configuración de nuestro clúster de Kubernetes, comenzaremos con la instalación y configuración de VPN mediante la consola de comandos de Amazon Web Services o AWS. Para ello, en la propia página de Amazon Web Services, nos moveremos hacia EC2, y dentro de este entraremos en instancias, y desde ahí lanzaremos nuestra propia instancia, que en nuestro caso se llama "ubuntuservervpn". Tendremos que ir a la interfaz de administración de OpenVPN y desde allí crear un nuevo usuario para la VPN, lo que permitirá conectarse a la VPN con este usuario además de con los nuestros, y configurar tanto su nombre como su contraseña. En nuestro caso, el usuario se llama "vpnuser1", y una vez terminada su creación haremos clic en el botón de "save" para guardar el usuario en la base de datos de OpenVPN, lo que permitirá que se conecte a la VPN. A continuación nos moveremos a la consola web de administración de OpenVPN. Una vez en ella, entraremos en la sección de Firewall, y comenzaremos a configurar un firewall para la VPN, lo que permitirá que los clientes de la VPN se conecten al servidor de esta de forma segura. Una vez terminada la instalación y configuración de la VPN, tanto en nuestros equipos servidor como en nuestros equipos clientes, deberemos instalar WireGuard. Pero antes debemos volver a actualizar los paquetes de nuestros equipos con el comando `sudo apt update`. Una vez terminada la actualización de los equipos, introducimos `sudo apt-get install wireguard`. A continuación, debemos generar las claves públicas y privadas que serán usadas por nuestro servidor usando el comando `wg genkey | tee /etc/wireguard/server_private.key | wg pubkey > /etc/wireguard/server_public.key`. Una vez generadas las claves que usará nuestro servidor, las deberemos incluir también en la configuración del firewall para que permita a usuarios externos unirse como clientes. Después de esto, deberemos comenzar a configurar el archivo principal del servidor wg0 maestro, incluyendo las claves de nuestros equipos cliente, así como definir

los parámetros principales del servidor, donde incluiremos, por parte del servidor, la IP con la que este funcionará, el puerto por el que estará escuchando, junto con su clave privada y definiciones de cómo manejar las peticiones que reciba, también junto con la lista de las claves de nuestros servidores clientes. Debemos incluir en cada uno de ellos su clave pública, para que nuestro servidor sea capaz de reconocerlos, y acabamos con la configuración de nuestro archivo de configuración.

```
root@ip-172-31-26-136: /etc/wireguard
GNU nano 6.2 wg0.conf
[Interface]
Address = 10.0.0.1/24
ListenPort = 51820
PrivateKey = YOVLJoSEMYte7cmLQFXd8zTwFeDpe9KA8CH9H6LDuHM=
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -A FORWARD -o wg0 -j ACCEPT; iptables -t na
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -D FORWARD -o wg0 -j ACCEPT; iptables -t na

# Placeholder for client peers - will be added after client keys are generat
# Peer 1
[Peer]
PublicKey = 1v60tCaTcwAI8DjQGg6gKoUairPiHPP27IwzECjqkAw=
AllowedIPs = 172.20.10.10, 10.0.0.2/32, 172.20.10.9

# Peer 2
[Peer]
PublicKey = kBUUwRZbk5SHlbn2VL9CF97V07QjoWnpX350ba9dsQY=
AllowedIPs = 172.20.10.5/32, 10.0.0.3/32

# Peer 3
[Peer]
PublicKey = 152H0xxk9Ezec0vVmeRDktilIzJv5qqGW/h05nfFMTS4=
AllowedIPs = 172.20.10.7/32, 10.0.0.4/32
```

Lo siguiente que tendremos que hacer es habilitar el servicio wg con el comando ``sudo systemctl enable wg-quick@wg0``. Antes de iniciar el servidor, lo reiniciaremos para no dar lugar a errores con el comando ``sudo systemctl restart wg-quick@wg0``, y con el comando ``sudo systemctl start wg-quick@wg0``, lo iniciaremos. Una vez iniciado el servicio en nuestro equipo cliente, introduciremos el comando ``wg genkey | tee /etc/wireguard/client_private.key | wg pubkey > /etc/wireguard/client_public.key``, para generar tanto su clave privada como su clave pública en el directorio ``/etc/wireguard``. Una vez generadas las claves nos moveremos al directorio Wireguard, haciendo uso del comando ``cd /etc/wireguard``, y dentro de este crearemos con el comando ``sudo nano wg.conf``, el archivo de configuración del servicio wg0, en el que introduciremos tanto la IP como la clave privada del equipo maestro, y nuestras claves tanto, la pública con su máscara correspondiente, como la privada al igual que las IPs permitidas. También especificaremos el intervalo de tiempo en el que se enviarán paquetes al servidor para que este compruebe que nuestro equipo está operativo. Este intervalo se especifica en segundos y en nuestro caso será de 25.

```
GNU nano 7.2 wg0.conf
[Interface]
Address = 10.0.0.5/24
PrivateKey = KOhr4vzuYBtdfTl3844M300FfU0UbPzJ/U0VGjaz7mM=

[Peer]
PublicKey = sLiXHdbYUklxt5J7KVzC2n7GE1plw5q03v9SDTMM1F0=
Endpoint = 34.224.69.63:51820
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 25
```


Una vez terminada la configuración del archivo, deberemos habilitar el servicio wg0 con el comando `sudo systemctl enable wg-quick@wg0`, e iniciarlo con el comando `sudo systemctl start wg-quick@wg0`. Por último, si queremos ver la información del servidor, lo único que tendremos que hacer será en el equipo maestro introducir el comando `wg show`.

```
root@ip-172-31-26-136:/etc/wireguard# sudo wg show
interface: wg0
  public key: sLiXHdbYUkIxt5J7KVzC2n7GE1plw5q03v9SDTMM1F0=
  private key: (hidden)
  listening port: 51820

peer: kBuUwRZbk5SHlbn2VL9CF97V07QjoWnpX350ba9dsQY=
  endpoint: 79.116.112.209:40800
  allowed ips: 172.20.10.5/32, 10.0.0.3/32
  latest handshake: 11 hours, 54 minutes, 36 seconds ago
  transfer: 7.38 MiB received, 7.33 MiB sent

peer: 1S2H0xxk9Ezec0vVmERDktLIzJv5qqGW/h05nFFMTS4=
  endpoint: 37.29.250.177:8866
  allowed ips: 172.20.10.7/32, 10.0.0.4/32
  latest handshake: 11 hours, 56 minutes, 8 seconds ago
  transfer: 872.42 KiB received, 2.00 MiB sent

peer: lv60tCaTcwAI8DjQGg6gKoUairPiHPP27lwzECjqkAw=
  endpoint: 37.29.250.177:8802
  allowed ips: 172.20.10.10/32, 10.0.0.2/32, 172.20.10.9/32
  latest handshake: 11 hours, 57 minutes, 17 seconds ago
  transfer: 75.38 KiB received, 43.26 KiB sent
root@ip-172-31-26-136:/etc/wireguard#
```

Uno de los aspectos más importantes de una empresa es su presentación, por lo que uno de nuestros principales objetivos era tener una cara por la que pudiéramos ser reconocidos fácilmente. Nos dispusimos a crear una página web completamente nueva que nos ayudaría a facilitar que posibles clientes nos conocieran, y que nuestros clientes actuales pudieran tener una manera sencilla y rápida de contactarse con nosotros.

Lo primero que hicimos cuando supimos que queríamos tener nuestra propia página web fue planificar cómo queríamos que fuera y qué mostraríamos en esta. También era importante darle un propósito final a la página, como los que ya he explicado antes: darnos a conocer y proporcionar una comunicación rápida y sencilla entre cliente y nosotros.

Una vez sabiendo qué propósito tendría nuestra página, comenzamos a pensar en el diseño. Empezamos con el archivo HTML, haciendo un primer esquema de cómo sería nuestra página. Lo primero y más llamativo sería nuestro encabezado, en el que se explica en pocas palabras quiénes somos y a qué nos dedicamos, junto con nuestros servicios.

Para facilitar la exploración durante la visualización de la página, necesitamos un menú con el que podamos acceder a los diferentes apartados. Estos incluirían un apartado de información sobre a qué nos dedicamos, nuestros orígenes y objetivos. También un apartado para proporcionar una manera rápida y sencilla de ponerse en contacto con nosotros, así como otro para iniciar sesión en la página, y así poder tener una atención más especializada para cada persona y empresa. Además, tendríamos un apartado de registro por si estás interesado en nuestros servicios pero aún no tienes una cuenta.

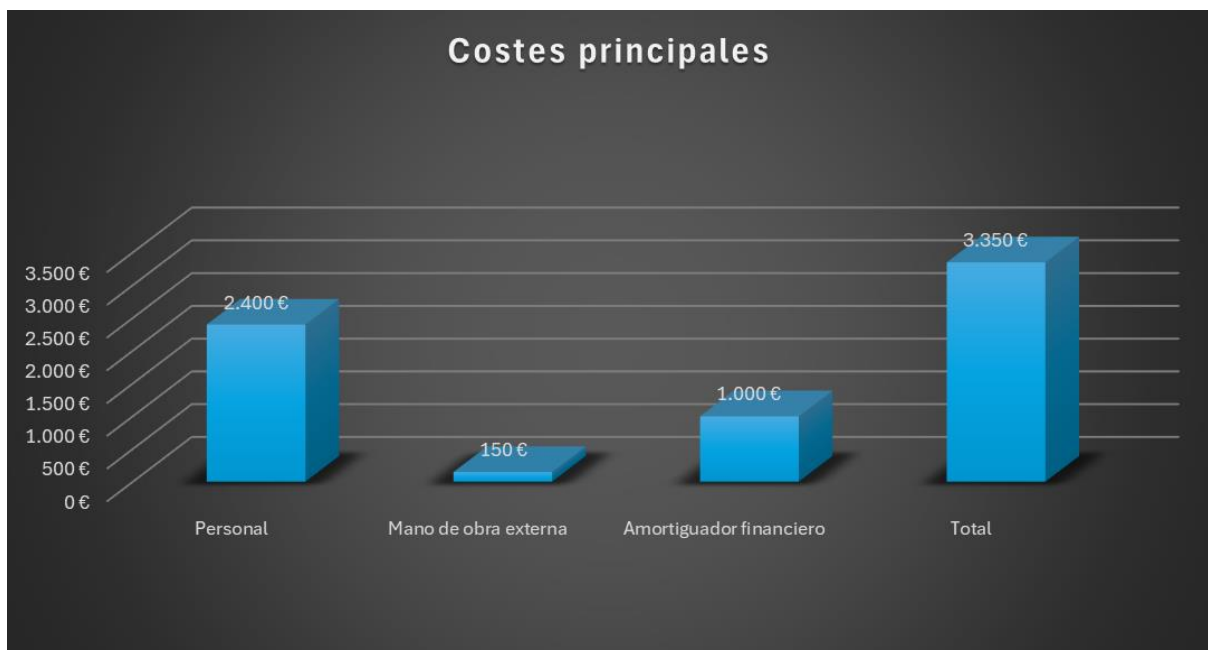
En nuestra página web mostraremos nuestros distintos planes de pago, junto con los servicios incluidos en cada uno de estos. También daremos opción a entidades con necesidades más únicas la posibilidad de mandarnos sus características para generarles un plan de gastos acorde a sus necesidades.

Objetivos finales	Creación de imágenes Docker desde 0	X
	Creación de clúster de Kubernetes	X
	Implantación de VPN	X
	Creación de Pagina Web	X

Todo proyecto importante que tenga una duración medianamente larga conllevará una serie de gastos. Comúnmente, al finalizar dicho proyecto, se suelen llevar a cabo diferentes tipos de informes en los que se recogen todos los gastos que han sido necesarios para la finalización del proyecto. En nuestro caso, hemos tenido diferentes tipos de gastos recogidos en dos grupos: los gastos más importantes y los costes principales de los materiales y recursos utilizados a lo largo de la elaboración del proyecto.

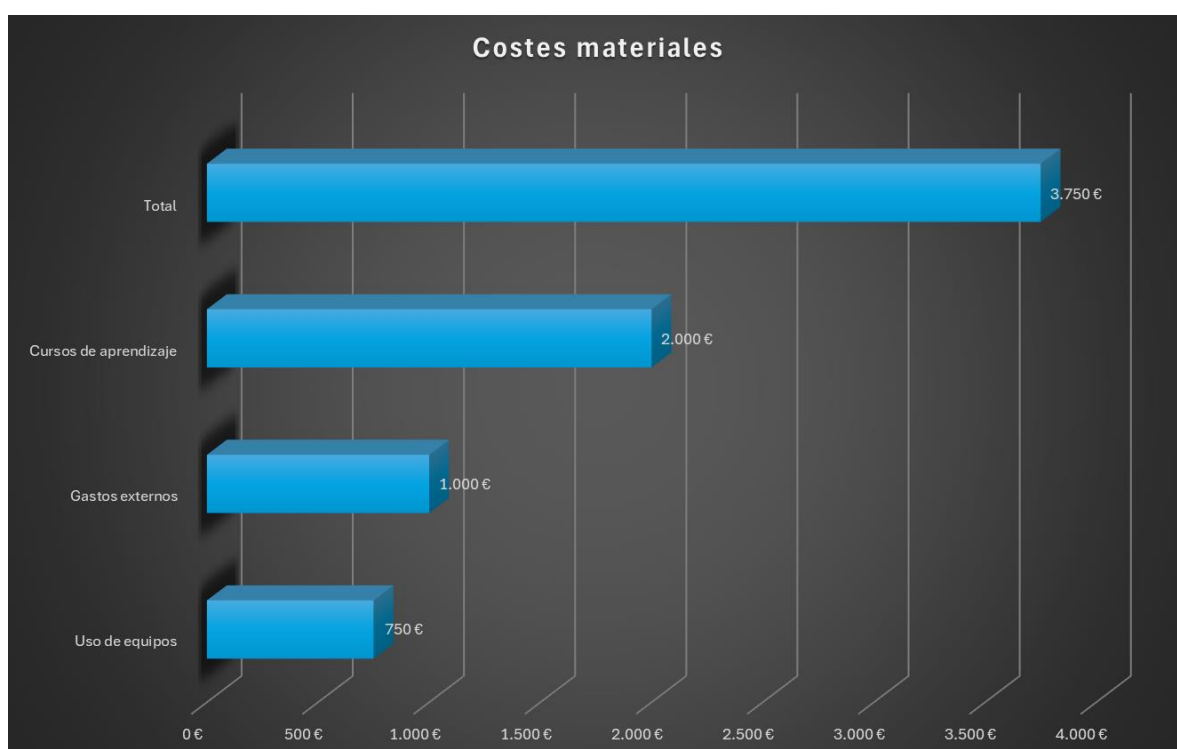
Iniciando con los gastos principales a lo largo de la duración del proyecto, hemos gastado la cantidad de 2.400 € en costes personales. Esto incluye tanto la mano de obra individual de cada integrante del grupo como el gasto energético y mental de cada uno. Además, durante algunos puntos del proyecto, tuvimos que pedir ayuda a diferentes personas, quienes nos ayudaron a resolver dudas principales y nos dieron explicaciones sobre cómo solucionar errores que no nos dejaban avanzar. Por último, en caso de posible fallo catastrófico, generamos lo que se suele llamar un amortiguador financiero para no quedarnos a cero.

Gráfico costes principales



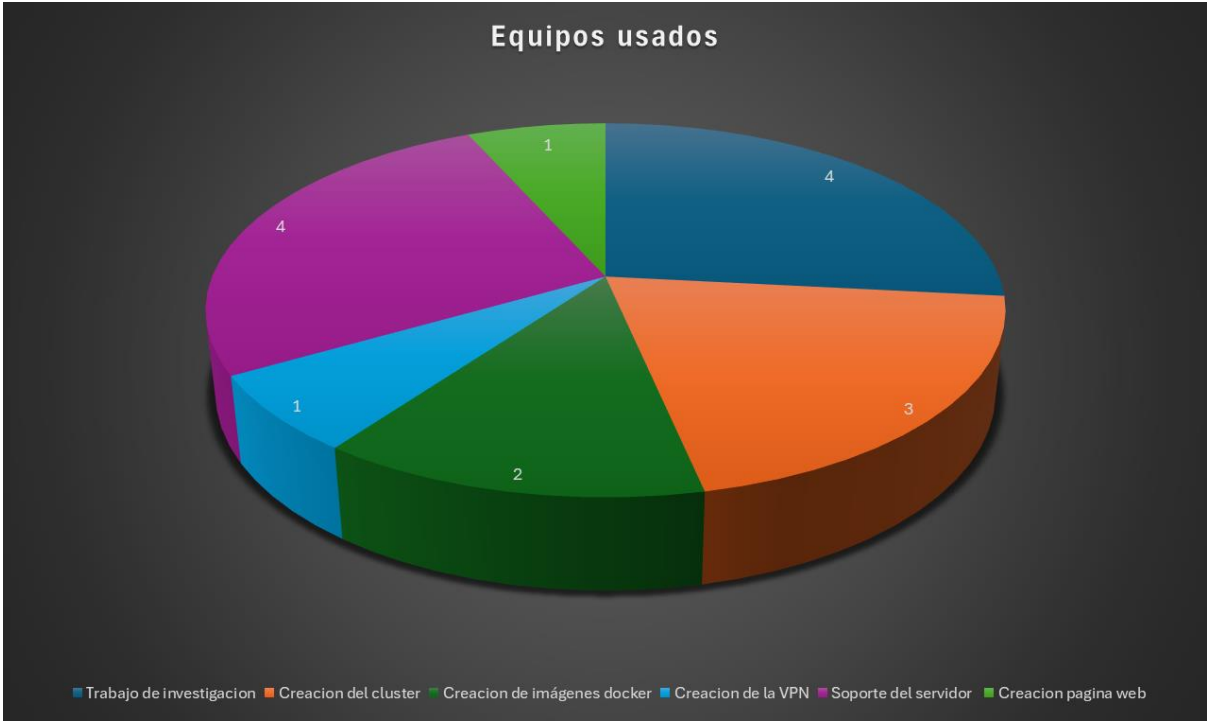
Una vez listados los costes principales pasamos a comprobar los costes materiales de nuestro proyecto, el continuo desgaste de los equipos a la hora de generar y soportar los distintos archivos generó un gasto de 750 €, también debido a que tuvimos que mejorar los sistemas de los equipos utilizados a lo largo del proyecto, y el uso de recursos externos tales como, el costo de la gasolina cada vez que teníamos que hacer una reunión de gran importancia, o el costo de vuelo por parte de uno de los integrantes del equipo generaron lo que llamamos gastos externos, junto con los que el costo de distintos tipos de cursos usados, para aprender las nuevas herramientas y tecnologías que necesitábamos para completar nuestro proyecto nos dejan con un gráfico de diferentes tipos de gastos.

Gráfico costes materiales



Por último, al tratarse de un proyecto de ámbito informático, también realizamos un gráfico especificando a qué se dedicaron los 8 equipos que teníamos a nuestra disposición a lo largo del proyecto.

Gráfico uso de equipos



2.4. Resultados y validación

Como resultado final, hemos conseguido montar un clúster de Kubernetes con alta disponibilidad totalmente operativo. Además, hemos incluido una VPN que nos permitirá añadir equipos al clúster de manera sencilla y acceder a los servicios del clúster desde cualquier red de manera segura. Esta VPN también nos facilitará la incorporación de nuevos equipos al clúster cuando sea necesario. Aunque notamos que el clúster tarda unos pocos segundos en empezar a ejecutarse, lo que podría ser minutos en caso de un error catastrófico y pérdida de datos, la alta disponibilidad del mismo dificulta que deje de funcionar por medios externos, sin que nosotros mismos lo apaguemos.

Una vez terminada la creación del clúster, hemos comprobado que efectivamente funciona sin problemas en las distintas simulaciones y entornos que hemos creado. Hemos probado que el equipo maestro del clúster puede ser tanto el de Rodrigo, Carlos, Manuel o Alejandro, así como distintas IPs en cada uno de nuestros equipos y cambiando las propias IPs tanto del equipo maestro como de los equipos cliente. El único problema que hemos encontrado es cuando las IPs de alguno de los equipos principales, como el maestro en ese momento, o alguno de los clientes internos, están mal implantadas. En esos casos, pueden surgir errores como la imposibilidad de la comunicación entre equipos o el mal funcionamiento del clúster. Hemos experimentado ambos casos a lo largo de la creación del clúster. También hemos comprobado que es imposible ponerlo en funcionamiento mediante el uso del internet de la universidad debido a sus restricciones. En ese caso, hemos tenido que usar el internet de nuestros móviles en los equipos para poder iniciar y mantener el servidor. Gracias a todas las pruebas realizadas y los distintos entornos en los que hemos probado el clúster, tenemos una gran confianza en su buen funcionamiento y pensamos que su instalación y uso a nivel laboral es completamente posible.

La normativa que hemos seguido ha sido tanto la de OpenVPN ofrecida por Amazon Web Services como la oficial por parte de Kubernetes. En algunos momentos durante la creación del clúster, hemos tenido que usar distintos tipos de documentación y normativas debido a la desactualización de la oficial. Sin embargo, esos casos están documentados en la bibliografía y, en su inmensa mayoría, hemos seguido la normativa oficial.

En el apartado "Creación de clúster de Kubernetes", que se encuentra en la sección de anexos y al que se puede acceder desde el índice, se puede comprobar tanto el diseño como la configuración, junto con la documentación necesaria para su comprensión y las imágenes del resultado final del clúster.

Después de un trabajo continuo pero incesante por parte de todos los miembros del grupo, hemos obtenido una serie de resultados satisfactorios. El clúster mantiene un uptime del 99%, demostrando así su alta disponibilidad. Ante diferentes técnicas de pérdida de datos, como la destrucción de uno de los servicios o el apagado de otro de ellos, el clúster vuelve a levantarlos de manera rápida y eficiente, generando así la implementación exitosa de una estrategia de recuperación de datos. El throughput del clúster siempre estuvo dentro de nuestros límites, aunque muchas veces fue menos de lo esperado debido a nuestra infraestructura y la configuración de la VPN. Por último, comprobamos que el clúster soporta sin problemas hasta 40 nodos en funcionamiento sin que haya ningún tipo de error en el rendimiento del mismo.

Además, tiene una distribución efectiva de la carga a través del clúster, evitando así problemas típicos como el cuello de botella y manteniendo el equilibrio del clúster. Gracias a la VPN, hemos podido implementar distintas herramientas de monitorización y login para proporcionar un mayor control sobre el clúster. También, una vez más gracias a la VPN, se implementa un acceso limitado únicamente a usuarios autenticados, añadiendo así una capa extra de seguridad y ofreciendo una mayor fiabilidad. Como una mejora más resultada de la implementación de la VPN, encontramos la facilitación de la administración del clúster, permitiéndonos acceder de manera segura al mismo desde cualquier lugar. Por último, pudimos comprobar el despliegue efectivo y seguro de las distintas aplicaciones, gracias a los canales cifrados para la transmisión de imágenes y datos.

3. CONCLUSIONES

Comenzamos con la investigación de Kubernetes junto con el uso de imágenes Docker. Una vez completado todo el trabajo de investigación necesario para empezar con el proyecto, comenzamos con la creación de las imágenes Docker de DHCP, DNS, Apache, NAS y virtualizador. Las imágenes de DHCP, DNS y Apache las creamos sin mucha dificultad, pero después de una reunión nos dimos cuenta de que necesitábamos crear las imágenes nosotros mismos desde cero. Después de deliberar cuidadosamente, decidimos reducir el número de imágenes de las cinco iniciales a las de DHCP, DNS y Nginx. No incluimos ni la imagen de NAS ni la de virtualizador, ya que vimos que no tendríamos el tiempo suficiente para crearlas desde cero, y también nos dimos cuenta de que no nos eran completamente necesarias para el clúster. Además, decidimos cambiar la imagen de Apache por una imagen de Nginx, ya que comprobamos que tenía una mayor compatibilidad con Kubernetes.

Al poco de iniciar con la creación de las imágenes, nos dieron la recomendación de crear todas las imágenes en un mismo docker, haciendo uso del comando Docker Compose. Sin embargo, después de una semana entera intentándolo y gracias a opiniones externas, nos dimos cuenta de que era prácticamente imposible. Por lo tanto, volvimos a comenzar con la creación de las imágenes Docker, pero esta vez en máquinas separadas.

Durante todo el proceso de creación de imágenes Docker, también comenzamos con la creación del clúster de Kubernetes. Para esto, utilizamos la documentación oficial del mismo. El problema era que daba por hecho que ya se tenían una serie de paquetes específicos instalados de base, por lo que al no tenerlos instalados, daba error. Sin embargo, en la página oficial no explicaba el error ni cómo solucionarlo ni los paquetes que se necesitaban instalar. Después de investigar y encontrar los paquetes una vez ya entrados en la instalación y configuración de Kubernetes, comenzamos a instalar las herramientas que íbamos a utilizar. Pero el problema fue que Kubernetes te da opción a utilizar múltiples herramientas y técnicamente todas funcionan para la creación de clústeres, pero cada una tiene sus limitaciones y usos específicos, cosa que no se especifica bien en ningún sitio y no se sabe cuál es la oficial de Kubernetes. Resultó que algunas son solo para clústeres en un solo nodo, y otras solo te permiten trabajar en local.

Una vez solucionado el problema de las herramientas, nos dimos cuenta de que para realizar los clústeres en diferentes nodos es esencial la utilización de IP fijas y de tener una conexión estable a todos los nodos junto con internet, cosa que en local o en una red LAN es muy sencillo, pero cuando es necesario emplearlo en cualquier red con las conexiones capadas es básicamente imposible. Por lo tanto, nos vimos obligados a utilizar redes que no estuvieran capadas, tales como las generadas por los móviles, lo que nos limitaba el uso de IPs y conexiones VPN. Si realizábamos una unión de redes privadas a través de VPN, todas las IPs locales chocaban entre ellas ya que al conectar cada dispositivo a cada teléfono para tener conexión, nos daba una IP automática de entre 172.20.10.2 y 172.20.10.14, lo que hacía que las IPs de los nodos coincidieran y generaran conflicto. La opción era descartar las VPN y realizar una simulación en local como si fuera la vida real. Sin embargo, gracias al trabajo duro y no rendirnos, conseguimos encontrar una VPN que nos permite establecer IPs estáticas y que de esta forma no haya conflicto y mediante VPN pueda ser posible inicializar el clúster y usarlo sin ningún problema.

Una vez terminada la creación del clúster, lo único que nos quedaba era mapear las imágenes Docker para que se instalaran dentro de los equipos de nuestros clientes y crear una página web para nuestra empresa, procesos en los que no tuvimos ningún problema mayor. Por último, si tuviéramos que listar los resultados obtenidos de todo nuestro trabajo, diríamos que, después de un esfuerzo continuo por parte de todos los miembros del grupo, obtuvimos resultados satisfactorios en la creación del clúster de Kubernetes. Mantuvimos un uptime del 99%, demostrando así alta disponibilidad. Además, demostramos capacidad de recuperación ante fallos, levantando rápidamente servicios caídos. A su vez, el throughput del clúster se mantuvo dentro de los límites esperados, aunque ocasionalmente fue menor debido a la infraestructura y configuración de la VPN. También, el clúster soportó hasta 40 nodos sin problemas de rendimiento, distribuyendo eficazmente la carga y evitando cuellos de botella. Por último, gracias a la VPN, se implementaron herramientas de monitorización y logging, proporcionando mayor control y seguridad al restringir el acceso solo a usuarios autenticados. La VPN también facilitó la administración remota y segura del clúster, y permitió el despliegue seguro de aplicaciones a través de canales cifrados para la transmisión de imágenes y datos.

3.1. Aportaciones

Gracias a la realización de este proyecto, hemos podido obtener distintos tipos de nuevos conocimientos, como la creación de imágenes Docker desde cero y su configuración completa, junto con una base bastante amplia de conocimientos sobre Kubernetes y sus distintos tipos de usos, además de una gran variedad de herramientas ofrecidas por este. A su vez, también hemos conseguido conocimientos bastante importantes sobre la creación de una VPN, así como sus distintos usos. Hemos encontrado estos conocimientos obtenidos por el camino como bastante novedosos y con un gran impacto en nuestras vidas diarias.

Si tuviéramos que listar los aspectos que nos han impactado en mayor medida, serían toda la información recogida y aprendida sobre VPNs. Aunque en nuestro caso no las hayamos usado en gran medida, con una simple mirada en ese mundo tan interesante y extenso, hemos sido capaces de ver el gran potencial que tienen, junto con la gran magnitud de aplicaciones y utilidades que pueden tener, así como todos los campos a los que son aplicables y mejorarían en gran medida muchos aspectos, tal y como nos ha pasado a nosotros.

Otro de los puntos más impactantes de nuestro proyecto ha sido mayoritariamente todo lo aprendido sobre Docker. Al igual que las VPN, es todo un mundo apasionante, con una gran cantidad de posibilidades y la capacidad de aplicar contenedores en todo tipo de campos. Lo aprendido sobre Docker en nuestro caso nos ha enseñado una gran cantidad de elementos, como la creación de las imágenes y subirlas a la página oficial de Docker, donde junto con las nuestras se pueden utilizar todo tipo de imágenes para todo tipo de cosas.

Por último, se encuentra Kubernetes, que aunque es verdad que es muy interesante y hemos tenido la oportunidad de aprender mucho de él, no creemos que tenga tanto uso como VPN o Docker, al menos en nuestro caso. Hemos encontrado muy interesante el proceso de creación del clúster, así como la configuración del mismo.

3.2. Trabajo futuro

Durante la creación y el desarrollo de todo proyecto, se generan una serie de ideas extras que no se terminan de implementar en el mismo debido a distintos inconvenientes, como pueden ser el límite de tiempo, la alta complicación de las mismas, o el hecho de que no son completamente necesarias para la creación del proyecto, por lo que son descartadas. En nuestro caso, hemos tenido las mismas ideas que no pudimos implementar, por lo que a continuación las presentamos:

- Creación de un sistema SAN/NAS privado para cada empresa implementado en Kubernetes: En la primera versión de nuestro proyecto, nos hubiera gustado incluir también en nuestro clúster un sistema NAS, pero debido a contratiempos y diferentes problemas y errores que nos encontramos, acabamos por descartar la idea, y finalmente nos quedamos con nuestros 3 servicios finales.
- Creación de un sistema de virtualización para hostear equipos de usuarios en el propio servidor utilizando Proxmox: Otra de las ideas que tuvimos que excluir debido a las mismas razones presentadas en el caso anterior, junto con la dificultad a la hora de completar el sistema y nuestro poco conocimiento sobre el tema, nos llevaron a descartar también este sistema de nuestro clúster.
- Unión de una web propia conectada a un script que, al coger un paquete de pago, se le hará al usuario rellenar un formulario sobre el servicio contratado y se creará automáticamente un namespace privado para el usuario: Aunque fuimos capaces de crear nuestra propia página web, tuvimos que descartar esta idea debido a su gran dificultad y al poco tiempo del que disponíamos.
- Creación de una pequeña empresa dedicada a la creación, configuración e implementación de clústeres de Kubernetes: Todo el mundo sueña con la idea de crear una empresa completamente nueva con su proyecto como idea. Aunque creemos que sería posible mejorar distintos puntos de nuestro proyecto para crear una empresa que tenga como base nuestro proyecto, todavía lo dejamos como un proyecto a futuro, debido a nuestra joven edad y nuestro deseo de adquirir conocimientos relacionados con otros campos.



4. BIBLIOGRAFÍA Y WEBGRAFÍA

Server World. (2007) www.server-world.info. Fecha de consulta: 17:30, abril 02, 2024 de https://www.server-world.info/en/note?os=Ubuntu_22.04&p=kubernetes&f=14

YouTube. (2005) www.youtube.com. Fecha de consulta: 19:15, abril 03, 2024 de https://www.youtube.com/watch?v=y_c_tPXusqM

KubeSphere. (2020) www.kubesphere.io. Fecha de consulta: 12:34, abril 12, 2024 de <https://www.kubesphere.io/docs/v3.3/installing-on-linux/high-availability-configurations/setup-ha-cluster-using-keepalived-haproxy/>

Server World. (2007) www.server-world.info. Fecha de consulta: 17:30, abril 13, 2024 de https://www.server-world.info/en/note?os=Ubuntu_22.04&p=kubernetes&f=14

VMware. (2005) <https://docs.vmware.com/es/>. Fecha de consulta: 09:45, abril 15, 2024 de <https://docs.vmware.com/en/vRealize-Operations/8.10/vrops-manager-load-balancing/GUID-EC001888-776B-42D5-9843-719EF08AB940.html>

Docker. (2013) docker.com. Fecha de consulta: 09:40, abril 15, 2024 de <https://www.docker.com/products/docker-hub/>

Chakray. (2015) www.chakray.com. Fecha de consulta: 14:00, abril 20, 2024 de <https://www.chakray.com/es/alta-disponibilidad-en-nginx-mediante-keepalived-ip-virtual/>

Kubernetes. (2014) <https://kubernetes.io/>. Fecha de consulta: 16:45, abril 19, 2024 de <https://kubernetes.io/es/docs/tasks/tools/included/install-kubectlinux/>

GitHub. (2008) <https://github.com/>. Fecha de consulta: 15:50, abril 27, 2024 de <https://github.com/linuxizate/UEM2024/tree/main>

Apuntes. (2013) <https://apuntes.de/#gsc.tab=0>. Fecha de consulta: 11:50, abril 17, 2024 de <https://apuntes.de/docker-certificacion-dca/docker-hub/#gsc.tab=0>

Domingo, J. (2020) www.josedomingo.org. Fecha de consulta: 18:00, mayo 1, 2024 de <https://www.josedomingo.org/pledin/2021/02/kubernetes-con-kind/>

YouTube. (2005) www.youtube.com. Fecha de consulta: 10:50, mayo 2, 2024 de <https://www.youtube.com/watch?v=8tOIYGZJON8>

Kubernetes. (2014) kubernetes.io. Fecha de consulta: 08:30, mayo 2, 2024 de <https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/>

YouTube. (2005) www.youtube.com. Fecha de consulta: 19:15, mayo 3, 2024 de https://www.youtube.com/watch?v=y_c_tPXusqM



YouTube. (2005) [www.youtube.com](https://www.youtube.com/watch?v=e9co-Y0eolk). Fecha de consulta: 15:10, mayo 5, 2024 de <https://www.youtube.com/watch?v=e9co-Y0eolk>

Kubernetes. (2014) [kubernetes.io](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/). Fecha de consulta: 11:00, mayo 8, 2024 de <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/>

Kubernetes. (2014) [kubernetes.io](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/). Fecha de consulta: 13:20, mayo 8, 2024 de <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/>

justmeandopensource. (2008) <https://github.com/>. Fecha de consulta: 13:35, mayo 7, 2024 de <https://github.com/justmeandopensource/kubernetes>

DigitalOcean. (2011) <https://www.digitalocean.com/>. Fecha de consulta: 13:00, mayo 7, 2024 de <https://www.digitalocean.com/community/questions/how-to-fix-docker-got-permission-denied-while-trying-to-connect-to-the-docker-daemon-socket>

Atlassian. (2002) www.atlassian.com. Fecha de consulta: 10:40, mayo 12, 2024 de https://www.atlassian.com/software/statuspage?utm_campaign=discordstatus.com&utm_content=SP-notifications&utm_medium=powered-by&utm_source=inapp

Amazon Web Services. (2002) [aws.amazon.com](https://aws.amazon.com/es/?nc2=h_lg). Fecha de consulta: 12:00, mayo 14, 2024 de https://aws.amazon.com/es/?nc2=h_lg

YouTube. (2005) [www.youtube.com](https://www.youtube.com/watch?v=NtPpeK0h_pl). Fecha de consulta: 12:45, mayo 15, 2024 de https://www.youtube.com/watch?v=NtPpeK0h_pl

Grafana. (2014) <https://grafana.com/>. Fecha de consulta: 14:55, mayo 16, 2024 de <https://grafana.com/>

Keepalived. (2000) <https://keepalived.org/>. Fecha de consulta: 10:20, mayo 18, 2024 de <https://keepalived.org/manpage.html>

Prometheus. (2012) <https://prometheus.io/>. Fecha de consulta: 13:10, mayo 18, 2024 de <https://prometheus.io/>

Mvallim. (2008) [mvallim.github.io](https://mvallim.github.io/kubernetes-under-the-hood/documentation/kube-metallb.html). Fecha de consulta: 15:10, mayo 19, 2024 de <https://mvallim.github.io/kubernetes-under-the-hood/documentation/kube-metallb.html>

Cherry Servers. (2001) www.cherryservers.com. Fecha de consulta: 15:20, mayo 19, 2024 de <https://www.cherryservers.com/blog/install-prometheus-ubuntu>

MetallB. (2020) [metallb.universe.tf](https://metallb.universe.tf/configuration/). Fecha de consulta: 16:05, mayo 20, 2024 de <https://metallb.universe.tf/configuration/>

MetallB. (2020) [metallb.universe.tf](https://metallb.universe.tf/apis/). Fecha de consulta: 18:20, mayo 20, 2024 de <https://metallb.universe.tf/apis/>

GitHub. (2008) [docs.github.com](https://docs.github.com/es). Fecha de consulta: 16:15, mayo 21, 2024 de <https://docs.github.com/es>



Containerd. (2013) <https://containerd.io/>. Fecha de consulta: 09:50, mayo 22, 2024 de <http://containerd.io/>

MetalLB. (2020) <https://metallb.universe.tf/>. Fecha de consulta: 11:35, mayo 24, 2024 de <https://metallb.universe.tf/installation/>

Amazon Web Services. (2002) [aws.amazon.com](https://aws.amazon.com/es/what-is-aws/). Fecha de consulta: 18:45, mayo 25, 2024 de <https://aws.amazon.com/es/what-is-aws/>

Amazon Web Services. (2002) [aws.amazon.com](https://aws.amazon.com/es/route53/what-is-dns/). Fecha de consulta: 14:10, mayo 26, 2024 de <https://aws.amazon.com/es/route53/what-is-dns/>

Aula Software Libre. (2012) <https://aulasoftwarelibre.github.io/taller-de-docker/>. Fecha de consulta: 10:50, mayo 15, 2024 de <https://aulasoftwarelibre.github.io/taller-de-docker/dockerfile/>

Apuntes. (2013) <https://apuntes.de/#gsc.tab=0>. Fecha de consulta: 11:50, abril 17, 2024 de <https://apuntes.de/docker-certificacion-dca/docker-hub/#gsc.tab=0>

DigitalOcean. (2011) <https://www.digitalocean.com/>. Fecha de consulta: 13:00, mayo 7, 2024 de <https://www.digitalocean.com/community/questions/how-to-fix-docker-got-permission-denied-while-trying-to-connect-to-the-docker-daemon-socket>

ChatGPT. (2024) chatgpt.com. Fecha de consulta: 17:00, mayo 1, 2024 de <https://chatgpt.com/>

OpenVPN. (2002) openvpn.net. Fecha de consulta: 14:00, mayo 29, 2024 de <https://openvpn.net/>



5. ANEXOS

Ahora procederemos a enseñar el proceso de creación de las imágenes Docker, el clúster y la página web, junto con la demostración de que funcionan, apoyándonos masivamente en el uso de listados de comandos y capturas de pantalla.

5.1. Creación de imágenes Docker

Imagen Docker DHCP

Preparación de la máquina virtual.

```
alejandro@alejandro-virtualbox:~$ sudo apt update  
[sudo] contraseña para alejandro:
```

```
alejandro@alejandro-virtualbox:~$ sudo apt upgrade  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias... Hecho  
Leyendo la información de estado... Hecho  
Calculando la actualización... Hecho
```

Instalación de Docker y sus dependencias.

```
alejandro@alejandro-virtualbox:~$ sudo apt install apt-transport-https  
ca-certificates curl software-properties-common  
[sudo] contraseña para alejandro:
```

```
alejandro@alejandro-virtualbox:~$ curl -fsSL https://download.docker.c  
om/linux/ubuntu/gpg | sudo apt-key add -
```

```
alejandro@alejandro-virtualbox:~$ sudo add-apt-repository "deb [arch=a  
md64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stab  
le"
```

```
alejandro@alejandro-virtualbox:~$ sudo apt update
```

```
alejandro@alejandro-virtualbox:~$ sudo apt install docker-ce
```

Creación del archivo Dockerfile y el archivo de configuración en el directorio del servicio.

```
alejandro@alejandro-virtualbox:~$ sudo mkdir docker-dhcp
```

```
alejandro@alejandro-virtualbox:~$ cd docker-dhcp/
```

```
alejandro@alejandro-virtualbox:~/dhcp$ sudo nano dockerfile
```



```
GNU nano 2.9.2 Dockerfile
#Usa una imagen base de Ubuntu
FROM ubuntu:latest

#Instala el servidor DHCP (isc-dhcp-server) y herramientas de red
RUN apt-get update && apt-get install -y isc-dhcp-server net-tools

#Copia el archivo de configuracion del servidor DHCP
COPY dhcpd.conf /etc/dhcp/dhcpd.conf

#Crea el directorio para la base de datos de arrendamientos
RUN mkdir -p /var/lib/dhcp/

#Crea manualmente el archivo de base de datos de arrendamientos
RUN touch /var/lib/dhcpd.leaves

#Establece permisos en el directorio
RUN chown -R dhcpd:dhcpd /var/lib/dhcp/

#Establece las opciones predeterminadas para el servidor DHCP
RUN echo 'INTERFACESv4="eth0" >> /etc/default/isc-dhcp-server'

#Puerto para escuchar peticiones DHCP
EXPOSE 67/udp

#Comando para iniciar el servidor DHCP
CMD ["/usr/sbin/dhcpd", "-f", "-d", "--no-pid"]
```

```
lejandro@alejandro-virtualbox:~/docker-dhcp$ sudo nano dhcpd.conf
```

Creación y puesta en marcha del contenedor Docker.

```
alejandro@alejandro-virtualbox:~/docker_dhcp$ sudo docker network create --subnet=192.168.0.0/24 my_custom_network
```

```
alejandro@alejandro-virtualbox:~/docker_dhcp$ sudo docker build -t docker-dhcp .
[+] Building 0.8s (12/12) FINISHED docker:default
=> [internal] load build definition from dockerfile 0.0s
=> => transferring dockerfile: 846B 0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest 0.7s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/7] FROM docker.io/library/ubuntu:latest@sha256:3f85b7caa 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 32B 0.0s
=> CACHED [2/7] RUN apt-get update && apt-get install -y isc-d 0.0s
=> CACHED [3/7] COPY dhcpd.conf /etc/dhcp/dhcpd.conf 0.0s
=> CACHED [4/7] RUN mkdir -p /var/lib/dhcp/ 0.0s
=> CACHED [5/7] RUN touch /var/lib/dhcpd.leaves 0.0s
=> CACHED [6/7] RUN chown -R dhcpd:dhcpd /var/lib/dhcp/ 0.0s
=> CACHED [7/7] RUN echo 'INTERFACESv4="eth0" >> /etc/default/ 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:263d35789e3d85b04edc84cbb96a43e3a4d 0.0s
=> => naming to docker.io/library/docker-dhcp 0.0s
```



```
alejandro@alejandro-virtualbox:~/docker_dhcp$ sudo docker run --name m  
aestro --net my_custom_network --ip 192.168.0.11 -d docker-dhcp
```

```
sudo apt update  
sudo apt upgrade  
  
sudo apt install apt-transport-https ca-certificates curl software-properties-common  
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  
  
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu  
$(lsb_release -cs) stable"  
sudo apt update  
sudo apt install docker-ce  
sudo systemctl status Docker  
  
sudo mkdir docker-dhcp  
cd docker-dhcp  
  
sudo nano Dockerfile  
# Usa una imagen base de Ubuntu  
FROM ubuntu:latest  
# Instala el servidor DHCP (isc-dhcp-server) y herramientas de red  
RUN apt-get update && apt-get install -y isc-dhcp-server net-tools  
# Copia el archivo de configuración del servidor DHCP  
COPY dhcpd.conf /etc/dhcp/dhcpd.conf  
# Crea el directorio para la base de datos de arrendamientos  
RUN mkdir -p /var/lib/dhcp/  
# Crea manualmente el archivo de base de datos de arrendamientos  
RUN touch /var/lib/dhcp/dhcpd.leases  
# Establece permisos en el directorio  
RUN chown -R dhcpd:dhcpd /var/lib/dhcp/  
# Establece las opciones predeterminadas para el servidor DHCP  
RUN echo 'INTERFACESv4="eth0"' >> /etc/default/isc-dhcp-server'  
# Puerto para escuchar peticiones DHCP  
EXPOSE 67/udp  
# Comando para iniciar el servidor DHCP  
CMD ["/usr/sbin/dhcpd", "-4", "-f", "-d", "--no-pid"]  
  
sudo nano dhcpd.conf  
  
subnet 192.168.0.0 netmask 255.255.255.0 {  
    range 192.168.0.10 192.168.0.100;  
    option routers 192.168.0.1;  
    option domain-name-servers 192.168.0.1;  
    option broadcast-address 192.168.0.255;  
    default-lease-time 600;  
    max-lease-time 7200;  
}  
  
sudo docker network create --subnet=192.168.0.0/24 my_custom_network  
sudo docker build -t docker-dhcp .  
sudo docker run --name servidor --net my_custom_network --ip 192.168.0.11 -d docker-  
dhcp
```

Listado 1: Imagen DHCP comandos



Imagen Docker DNS

Preparación de la máquina virtual.

```
alejandro@alejandro-virtualbox:~$ sudo apt update  
[sudo] contraseña para alejandro:
```

```
alejandro@alejandro-virtualbox:~$ sudo apt upgrade  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias... Hecho  
Leyendo la información de estado... Hecho  
Calculando la actualización... Hecho
```

Instalación de Docker y sus dependencias.

```
alejandro@alejandro-virtualbox:~$ sudo apt install apt-transport-https  
ca-certificates curl software-properties-common  
[sudo] contraseña para alejandro:
```

```
alejandro@alejandro-virtualbox:~$ curl -fsSL https://download.docker.c  
om/linux/ubuntu/gpg | sudo apt-key add -
```

```
alejandro@alejandro-virtualbox:~$ sudo add-apt-repository "deb [arch=a  
md64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stab  
le"
```

```
alejandro@alejandro-virtualbox:~$ sudo apt update
```

```
alejandro@alejandro-virtualbox:~$ sudo apt install docker-ce
```

Creación del archivo Dockerfile y los documentos de configuración.

```
alejandro@alejandro-virtualbox:~$ sudo mkdir dns
```

```
alejandro@alejandro-virtualbox:~$ cd dns
```

```
alejandro@alejandro-virtualbox:~/dns$ sudo nano dockerfile
```



```
kubernetes@kubernetes: ~/dns
Archivo Acciones Editar Vista Ayuda
kubernetes@kubernetes: ~/dns x
GNU nano 6.2 dockerfile
FROM debian:latest

RUN apt update
RUN apt install bind9 -y

EXPOSE 53/tcp
EXPOSE 53/udp

COPY ./bind9/named.conf.local /etc/bind/named.conf.local
COPY ./bind9/named.conf.options /etc/bind/named.conf.options
COPY ./bind9/db.192.168 /etc/bind/db.192.168
COPY ./bind9/db.rodri.com /etc/bind/db.rodri.com

USER bind:bind

[ 18 líneas leídas ]
^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^J Justificar ^_ Ir a línea
```

```
alejandro@alejandro-virtualbox:~/dns$ sudo mkdir bind9
```

```
alejandro@alejandro-virtualbox:~/dns$ cd bind9
```

```
alejandro@alejandro-virtualbox:~/dns/bind9$ sudo nano named.conf.local
```

```
GNU nano 7.2 named.conf.local
zone "goldenarrow.com"{
    type master;
    file "/etc/bind/db.rodri.com";
};

zone "168.192.in-addr.arpa"{
    type master;
    file "/etc/bind/db192.168";
};
```

```
alejandro@alejandro-virtualbox:~/dns/bind9$ sudo nano named.conf.options
```



```
kubernetes@kubernetes: ~/dns/bind9
Archivo Acciones Editar Vista Ayuda
kubernetes@kubernetes: ~/dns/bind9
GNU nano 6.2 named.conf.options
acl trustedclients {
    localhost;
    localnets;
    192.168.0.0/24;
};

options {
    directory "/var/cache/bind";

    recursion yes;
    allow-query { trustedclients; };
    allow-recursion { trustedclients; };

    forward only;
    forwarders {
        1.1.1.2;
        1.0.0.2;
    };
};
```

```
alejandro@alejandro-virtualbox:~/dns/bind9$ sudo nano db.goldenarrow.com
```

```
GNU nano 7.2 db.goldenarrow.com
$TTL 24h
$ORIGIN goldenarrow.com.
@      IN      SOA      testdns.goldenarrow.com. admin.goldenarrow.com. (
                                1          ; serial number
                                24h         ; refresh
                                2h          ; update retry
                                1000h       ; expire
                                2d          ; minimum
                                )

                                IN      NS      testdns1.goldenarrow.com.

goldenarrow1  IN      A       192.168.0.10
goldenarrow2  IN      A       192.168.0.11
goldenarrow3  IN      A       192.168.0.12
testdns1      IN      A       192.168.0.30
dockerdemo    IN      A       192.168.0.30
pbs           IN      A       192.168.0.31
```

```
alejandro@alejandro-virtualbox:~/dns/bind9$ sudo nano db.192.168
```



```
GNU nano 7.2 db.192.168
;TTL 24h
$ORIGIN 168.192.in-addr-arpa.
@      IN      SOA      testdns1.goldenarrow.com. admin.goldenarrow.com. (
                                1          ; serial number
                                24h         ; refresh
                                2h          ; update retry
                                1000h       ; expire
                                2d          ; minimum
                                )
                                IN      NS      testdns1.goldenarrow.com.

$ORIGIN 0.168.192.in-addr.arpa.
10     IN      A        goldenarrow1.goldenarrow.com.
11     IN      A        goldenarrow2.goldenarrow.com.
12     IN      A        goldenarrow3.goldenarrow.com.
30     IN      A        dockerdemo.goldenarrow.com.
31     IN      A        pbs.goldenarrow.com.
```

```
alejandro@alejandro-virtualbox:~/dns/bind9$ ls
db.192.168  db.goldenarrow.com  named.conf.local  named.conf.options
```

```
alejandro@alejandro-virtualbox:~/dns$ ls
bind9  dockerfile
```

Creación y puesta en marcha del contenedor Docker.

```
alejandro@alejandro-virtualbox:~/dns/bind9$ sudo docker build -t goldenarrow-dns .
```

```
alejandro@alejandro-virtualbox:~/dns$ sudo docker network create --subnet=192.168.0.0/24 goldenarrownet
```

```
alejandro@alejandro-virtualbox:~/dns$ sudo docker run --name dnsgoldenarrow --net rodrinet --ip 192.168.0.11 --dns
192.168.0.10 -d goldenarrow-dns
```

```
sudo apt update
sudo apt upgrade

sudo apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
sudo apt update
sudo apt install docker-ce
sudo systemctl status Docker

sudo mkdir dns
sudo nano dockerfile
FROM debian:latest

RUN apt update
RUN apt install bind9 -y
```



```
EXPOSE 53/tcp
EXPOSE 53/udp

COPY ./bind9/named.conf.local /etc/bind/named.conf.local
COPY ./bind9/named.conf.options /etc/bind/named.conf.options
COPY ./bind9/db.192.168 /etc/bind/db.192.168
COPY ./bind9/db.rodri.com /etc/bind/db.rodri.com

USER bind:bind

sudo mkdir bind9
sudo nano named.conf.local
zone "rodri.com" {
    type master;
    file "/etc/bind/db.rodri.com";
};
zone "168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192.168";
};
acl trustedclients {
    localhost;
    localnets;
    192.168.0.0/24;
};

sudo nano named.conf.options
options {
    directory "/var/cache/bind";
    recursion yes;
    allow-query { trustedclients; };
    allow-recursion { trustedclients; };
    forward only;
    forwarders {
        1.1.1.2;
        1.0.0.2;
    };
};

sudo nano db.goldenarrow.com
$TTL 24h
```



```
$ORIGIN goldenarrow.com.
@ IN SOA testdns.goldenarrow.com. admin.goldenarrow.com. (
    1      ; serial number
    24h    ; refresh
    2h     ; update retry
    1000h  ; expire
    2d     ; minimum
)

    IN NS testdns1.goldenarrow.com.
goldenarrow1 IN A 192.168.0.10
goldenarrow2 IN A 192.168.0.11
goldenarrow3 IN A 192.168.0.12
testdns1     IN A 192.168.0.30
dockerdemo   IN A 192.168.0.30
pbs          IN A 192.168.0.31
sudo nano db.192.168
$TTL 24h
$ORIGIN 168.192.in-addr.arpa.
@ IN SOA testdns1.goldenarrow.com. admin.goldenarrow.com. (
    1      ; serial number
    24h    ; refresh
    2h     ; update retry
    1000h  ; expire
    2d     ; minimum
)

    IN NS testdns1.goldenarrow.com.
$ORIGIN 0.168.192.in-addr.arpa.
10 IN A goldenarrow1.goldenarrow.com.
11 IN A goldenarrow2.goldenarrow.com.
12 IN A goldenarrow3.goldenarrow.com.
30 IN A dockerdemo.goldenarrow.com.
31 IN A pbs.goldenarrow.com.
sudo docker build -t goldenarrow.dns .
sudo docker network create --subnet=192.168.0.0/24 goldenarrownet
sudo docker run --name dnsgoldenarrow --net goldenarrownet --ip 192.168.0.11 --dns 192.168.0.10
-d goldenarrow-dns
```

Listado 2: Imagen DNS comandos



Imagen Docker Nginx

Preparación de la máquina virtual.

```
alejandro@alejandro-virtualbox:~$ sudo apt update  
[sudo] contraseña para alejandro:
```

```
alejandro@alejandro-virtualbox:~$ sudo apt upgrade  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias... Hecho  
Leyendo la información de estado... Hecho  
Calculando la actualización... Hecho
```

Instalación de Docker y sus dependencias.

```
alejandro@alejandro-virtualbox:~$ sudo apt install apt-transport-https  
ca-certificates curl software-properties-common  
[sudo] contraseña para alejandro:
```

```
alejandro@alejandro-virtualbox:~$ curl -fsSL https://download.docker.c  
om/linux/ubuntu/gpg | sudo apt-key add -
```

```
alejandro@alejandro-virtualbox:~$ sudo add-apt-repository "deb [arch=a  
md64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stab  
le"
```

```
alejandro@alejandro-virtualbox:~$ sudo apt update
```

```
alejandro@alejandro-virtualbox:~$ sudo apt install docker-ce
```

Creación del archivo Dockerfile, archivos de configuración y página web.

```
alejandro@alejandro-virtualbox:~$ sudo mkdir nginx
```

```
alejandro@alejandro-virtualbox:~$ cd nginx/
```

```
alejandro@alejandro-virtualbox:~/nginx$ sudo nano dockerfile
```

```
GNU nano 7.2 dockerfile  
# Usa la imagen base oficial de Nginx  
FROM nginx:latest  
  
# Copia el archivo de configuración de Nginx  
COPY ./default.conf /etc/nginx/conf.d/default.conf  
  
# Copia el contenido de la página web  
COPY ./index.html /usr/share/nginx/html/index.html
```

```
alejandro@alejandro-virtualbox:~/nginx$ sudo nano default.conf
```



```
GNU nano 7.2 default.conf
server {
    listen 80;
    server_name goldenarrow3.goldenarrow.com;

    location / {
        root /usr/share/nginx/html;
        index index.html;
    }
}
```

```
alejandro@alejandro-virtualbox:~/nginx$ sudo nano index.html
```

```
GNU nano 7.2 index.html
<!DOCTYPE html>
<html>
<head>
    <title>GoldeArrow3</title>
</head>
<body>
    <h1>Welcome to goldenarrow3.goldenarrow.com</h1>
    <p>ESTA PAGINA ESTA EN LA IP 192.168.0.12</p>
</body>
</html>
```

Montamos y arrancamos el Docker.

```
alejandro@alejandro-virtualbox:~/nginx$ sudo docker build -t goldenarrownginx .
```

```
alejandro@alejandro-virtualbox:~/nginx$ sudo docker network create --subnet=192.168.0.0/24 --driver bridge goldenarrownet
```

```
alejandro@alejandro-virtualbox:~/nginx$ sudo docker run --name nginxgoldenarrow --net goldenarrownet --ip 192.168.0.12 --dns 192.168.0.10 -d goldenarrownginx
```

Welcome to goldenarrow3.goldenarrow.com

ESTA PAGINA ESTA EN LA IP 192.168.0.12



```
sudo apt update
sudo apt upgrade

sudo apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

sudo apt update
sudo apt install docker-ce
sudo systemctl status Docker
sudo mkdir nginx
cd nginx/
sudo nano dockerfile
# Usa la imagen base oficial de Nginx

FROM nginx:latest

# Copia el archivo de configuración de Nginx
COPY ./default.conf /etc/nginx/conf.d/default.conf

# Copia el contenido de la página web
COPY ./index.html /usr/share/nginx/html/index.html

sudo nano default.conf

server {
    listen 80;

    server_name goldenarrow3.goldenarrow.com;

    location / {
        root /usr/share/nginx/html;
        index index.html;
    }
}

sudo nano index.html

<!DOCTYPE html>

<html>

<head>

    <title>Goldenarrow3 Page</title>

</head>

<body>

    <h1>Welcome to goldenarrow3.goldenarrow.com</h1>

    <p>ESTA PAGINA ESTA EN LA IP 192.168.0.12</p>

</body>

</html>

sudo docker build goldenarrownginx .

sudo docker network create --subnet=192.168.0.0/24 --driver bridge rodrinet

sudo docker run --name nginxgoldenarrow --net goldenarrownet --ip 192.168.0.12 --dns 192.168.0.10 -d
goldenarrow.nginx
```

Listado 3: Imagen Nginx comandos



5.2. Creación clúster de Kubernetes

Estructura del clúster

Nube			
Host	IP LAN	IP VPN	Public
Ubuntu	172.31.26.136/20	10.0.0.1/24	34.224.69.63
VIRTUAL		10.0.0.9/24	

PC1/PcRodrigo		
Host	IP LAN	IP VPN
kuproxy1	172.20.10.?	10.0.0.10/24
kumaster1	172.20.10.?	10.0.0.20/24
kuworker1	172.20.10.?	10.0.0.30/24
VIRTUAL		10.0.0.9/24

PC2/PcCarlos		
Host	IP LAN	IP VPN
kuproxy2	172.20.10.?	10.0.0.11/24
kumaster2	172.20.10.?	10.0.0.21/24
kuworker2	172.20.10.?	10.0.0.31/24
VIRTUAL		10.0.0.9/24

PC3/PcCasaRodri		
Host	IP LAN	IP VPN
kuproxy3	172.20.10.?	10.0.0.12/24
kumaster3	172.20.10.?	10.0.0.22/24
kuworker3	172.20.10.?	10.0.0.32/24
VIRTUAL		10.0.0.9/24



```
sudo apt install keepalived haproxy psmisc -y
vi /etc/haproxy/haproxy.cfg

frontend kubernetes-frontend
    bind *:6443
    mode tcp
    option tcplog
    default_backend kubernetes-backend

backend kubernetes-backend
    mode tcp
    option tcplog
    option tcp-check
    balance roundrobin
    default-server inter 10s downinter 5s rise 2 fall 3 slowstart 60s maxconn 250 maxqueue
256 weight 100
    server kumaster1 172.20.10.20:6443 check
    server kumaster2 172.20.10.21:6443 check
    server kumaster3 172.20.10.22:6443 check
    #server kuproxy0 192.168.1.10:6443 check
    #server kuproxy2 192.168.1.12:6443 check

peers mypeers
    peer lb0 172.20.10.10:1024
    peer lb2 172.20.10.12:1024
)
systemctl enable haproxy

systemctl restart haproxy
systemctl enable haproxy

sudo nano /etc/keepalived/keepalived.conf
global_defs {
    notification_email {
    }
    router_id LVS_DEVEL
    vrrp_skip_check_adv_addr
    vrrp_garp_interval 0
    vrrp_gna_interval 0
}
vrrp_script chk_haproxy {
    script "killall -0 haproxy"
    interval 2
    weight 2
}
vrrp_instance haproxy-vip {
    state BACKUP
    priority 100
    interface ens33
    virtual_router_id 60
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    unicast_src_ip 172.20.10.10
    unicast_peer {
        172.20.10.11
        172.20.10.12
    }
    virtual_ipaddress {
        172.20.10.9/28
    }
}
systemctl enable keepalived
```

Listado 4: Configuración HAProxy y Keepalived comandos



```
/etc/hosts
/etc/hostname

sudo apt update
sudo apt upgrade

sudo swapoff -a
sudo sed -i 's/ swap / s/^\(.*\)$/#\1/g' /etc/fstab

sudo tee /etc/modules-load.d/containerd.conf <<EOF
overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter

sudo tee /etc/sysctl.d/kubernetes.conf <<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF

sudo sysctl --system

sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/trusted.gpg.d/docker.gpg
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"

sudo apt update
sudo apt install -y containerd.io

containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1
sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml

sudo systemctl restart containerd
sudo systemctl enable containerd

curl -fsSL https://pkgs.k8s.io/core:stable:v1.30/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:stable:v1.30/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list

sudo apt update
sudo apt install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

Listado 5: Comandos creación máquina Master

```
kubeadm init --control-plane-endpoint IPMASTER:6443 --upload-certs

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

Listado 6: Comandos creación cluster máquina Master



```
kubeadm join 192.168.1.9:6443 --token yiu5lw.vlf1royrxc255ehi \
--discovery-token-ca-cert-hash
sha256:022c50bdd812e69468cb6505f5637fd8f3afceffdece62f9e692f4dba0e2e424 \
--control-plane --certificate-key
54412941d589bb2c19f89901943ca2e34123d1d5a063264d4cc976f50c1ea6cb

kubect1 apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml

kubect1 get pods -n kube-system
kubect1 get nodes
```

Listado 7: Comandos unión cluster máquina Master

```
kubeadm join 192.168.1.9:6443 --token yiu5lw.vlf1royrxc255ehi \
--discovery-token-ca-cert-hash
sha256:022c50bdd812e69468cb6505f5637fd8f3afceffdece62f9e692f4dba0e2e424

kubect1 apply -f "nombre.yaml"

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3 # Número de réplicas (pods) que deseas ejecutar
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest # Imagen de Docker de NGINX
          ports:
            - containerPort: 80 # Puerto en el contenedor

apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: NodePort
```

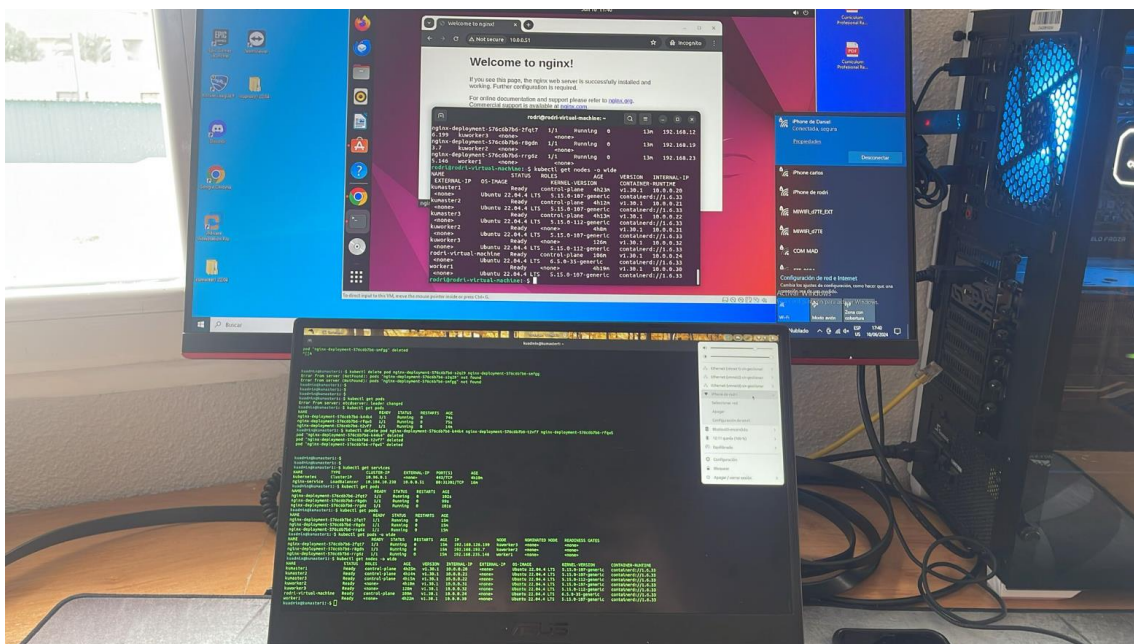
Listado 8: Comandos unión cluster máquina Worker

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
export KUBECONFIG=/etc/kubernetes/admin.conf
Run "kubectl apply -f [podnetwork].yaml"

kubeadm join 10.0.0.9:6443 --token o25bh0.d8b9kir4yeacuzd5 \
--discovery-token-ca-cert-hash
sha256:d97a18ffddc143f43fdbb56a3ceec2ad1ded86948b812cacc57de480ccf4ebe7 \
--control-plane --certificate-key
3ddb30d2d1d4540ece854e2908f0f76893ad1151d6ccc59b3b046a9d80a10721
kubeadm join 10.0.0.9:6443 --token o25bh0.d8b9kir4yeacuzd5 \
--discovery-token-ca-cert-hash
sha256:d97a18ffddc143f43fdbb56a3ceec2ad1ded86948b812cacc57de480ccf4ebe7
```

Listado 9: Comandos iniciar clúster

Capturas funcionamiento del clúster



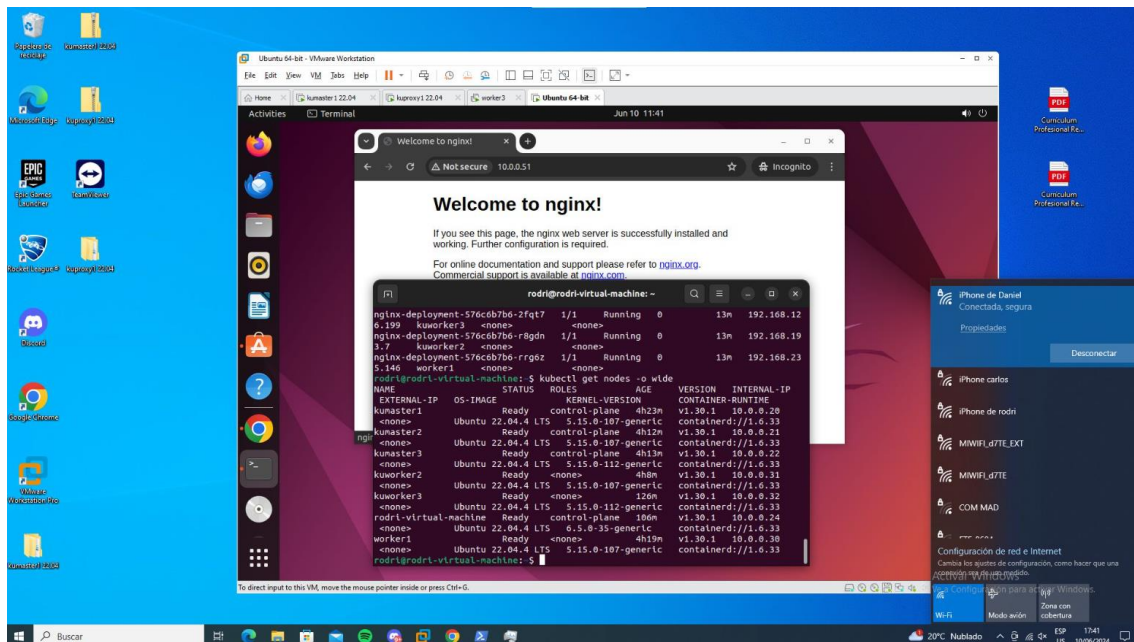


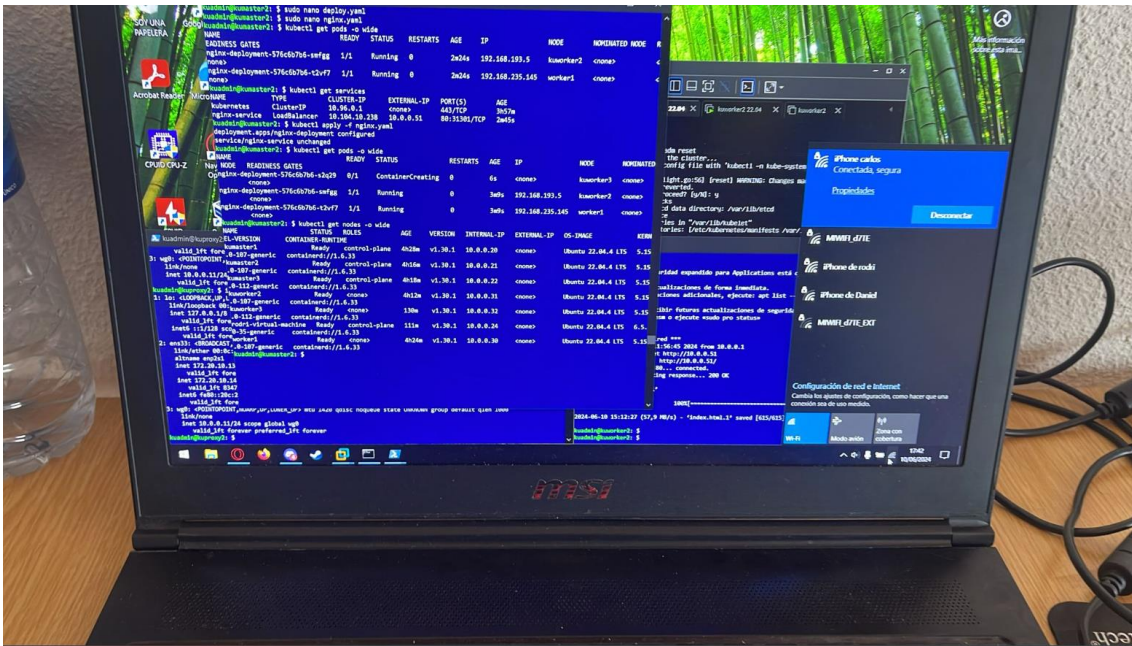
```
pod "nginx-deployment-576c6b7b6-snfqg" deleted
[EA]

koadmin@kumaster1:~$ kubectl delete pod nginx-deployment-576c6b7b6-s2q29 nginx-deployment-576c6b7b6-snfqg
Error from server (NotFound): pods "nginx-deployment-576c6b7b6-s2q29" not found
Error from server (NotFound): pods "nginx-deployment-576c6b7b6-snfqg" not found
koadmin@kumaster1:~$
koadmin@kumaster1:~$ kubectl get pods
Error from server: etcdserver: leader changed
koadmin@kumaster1:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-576c6b7b6-k44k4    1/1     Running   0           74s
nginx-deployment-576c6b7b6-rfqd5    1/1     Running   0           72s
nginx-deployment-576c6b7b6-t2vf7    1/1     Running   0           14s
koadmin@kumaster1:~$ kubectl delete pod nginx-deployment-576c6b7b6-k44k4 nginx-deployment-576c6b7b6-t2vf7 nginx-deployment-576c6b7b6-rfqd5
pod "nginx-deployment-576c6b7b6-k44k4" deleted
pod "nginx-deployment-576c6b7b6-t2vf7" deleted
pod "nginx-deployment-576c6b7b6-rfqd5" deleted

koadmin@kumaster1:~$
koadmin@kumaster1:~$ kubectl get services
NAME      TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes ClusterIP   10.96.0.1      <none>         443/TCP          4018m
nginx-service LoadBalancer 10.104.10.238 10.0.0.51      80:31101/TCP     16m

koadmin@kumaster1:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-576c6b7b6-2fq77    1/1     Running   0           102s
nginx-deployment-576c6b7b6-r8gdn    1/1     Running   0           99s
nginx-deployment-576c6b7b6-r8gdn    1/1     Running   0           101s
koadmin@kumaster1:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-576c6b7b6-2fq77    1/1     Running   0           15s
nginx-deployment-576c6b7b6-r8gdn    1/1     Running   0           15s
nginx-deployment-576c6b7b6-r8gdn    1/1     Running   0           15s
koadmin@kumaster1:~$ kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE             NOMINATED NODE   READINESS GATES
nginx-deployment-576c6b7b6-2fq77    1/1     Running   0           15s   192.168.128.199 kuworker3        <none>            <none>
nginx-deployment-576c6b7b6-r8gdn    1/1     Running   0           15s   192.168.128.172 kuworker2        <none>            <none>
nginx-deployment-576c6b7b6-r8gdn    1/1     Running   0           15s   192.168.235.146 worker1          <none>            <none>
koadmin@kumaster1:~$ kubectl get nodes -o wide
NAME                                STATUS   ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
kumaster1                            Ready   control-plane 4h25m v1.30.1 10.0.0.20    <none>        Ubuntu 22.04.4 LTS   5.15.0-107-generic containerd://1.6.33
kumaster2                            Ready   control-plane 4h14m v1.30.1 10.0.0.21    <none>        Ubuntu 22.04.4 LTS   5.15.0-107-generic containerd://1.6.33
kumaster3                            Ready   control-plane 4h15m v1.30.1 10.0.0.22    <none>        Ubuntu 22.04.4 LTS   5.15.0-112-generic containerd://1.6.33
kuworker2                            Ready   <none>     4h18m v1.30.1 10.0.0.31    <none>        Ubuntu 22.04.4 LTS   5.15.0-107-generic containerd://1.6.33
kuworker3                            Ready   <none>     128m   v1.30.1 10.0.0.32    <none>        Ubuntu 22.04.4 LTS   5.15.0-112-generic containerd://1.6.33
rodri-virtual-machine                Ready   <none>     149m   v1.30.1 10.0.0.24    <none>        Ubuntu 22.04.4 LTS   6.5.0-35-generic   containerd://1.6.33
worker1                             Ready   <none>     4h22m v1.30.1 10.0.0.30    <none>        Ubuntu 22.04.4 LTS   5.15.0-107-generic containerd://1.6.33
koadmin@kumaster1:~$
```







5.3. Creación y configuración de VPN

Launch an instance

Name and tags

ubuntuservvpn

Application and OS Images (Amazon Machine Image)

AMI from catalog

Amazon Machine Image (AMI)

ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20240411-ami-0e001c9271cf7f3b9

Summary

Number of Instances: 1

Software Image (AMI): Ubuntu Server 22.04 LTS (HVM),...read more

Virtual server type (Instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Network settings

Network

vpc-0b529925e3de7e9b3

Subnet

No preference (Default subnet in any availability zone)

Auto-assign public IP

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups)

Create security group

Select existing security group

We'll create a new security group called 'launch-wizard-6' with the following rules:

Allow SSH traffic from: Anywhere

Allow HTTPS traffic from the Internet

Allow HTTP traffic from the Internet

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Configure storage

1x 30 GiB gp2 Root volume (Not encrypted)

Summary

Number of Instances: 1

Software Image (AMI): Ubuntu Server 22.04 LTS (HVM),...read more

Virtual server type (Instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 30 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.



Instalación del servicio en el cliente y maestro.

```
ubuntu@ip-172-31-26-136: ~  
ubuntu@ip-172-31-26-136:~$ sudo apt-get install wireguard
```

Generación de claves para el servidor.

```
ubuntu@ip-172-31-26-136:~$ wg genkey | tee /etc/wireguard/server_private.key | wg pubkey > /etc/wireguard/server_public.key
```

Creación del archivo de configuración.

```
root@ip-172-31-26-136: /etc/wireguard  
GNU nano 6.2 wg0.conf  
[Interface]  
Address = 10.0.0.1/24  
ListenPort = 51820  
PrivateKey = YOVLJo5EMYte7cmLQFXd8zTwFeDpe9KA8CH9H6LDuHM=  
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -A FORWARD -o wg0 -j ACCEPT; iptables -t na  
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -D FORWARD -o wg0 -j ACCEPT; iptables -t >  
  
# Placeholder for client peers - will be added after client keys are generat  
# Peer 1  
[Peer]  
PublicKey = iv60tCaTcwAI8DjQGg6gKoUairPiHPP27iwzECjqkAw=  
AllowedIPs = 172.20.10.10, 10.0.0.2/32, 172.20.10.9  
  
# Peer 2  
[Peer]  
PublicKey = kBuUwRZbk5SHlbn2VL9CF97V07QjoWnpX350ba9dsQY=  
AllowedIPs = 172.20.10.5/32, 10.0.0.3/32  
  
# Peer 3  
[Peer]  
PublicKey = 1S2H0xxk9EzecOvVmERDktlIzJv5qqGW/h05nffMTS4=  
AllowedIPs = 172.20.10.7/32, 10.0.0.4/32  
  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo  
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify    ^_ Go To Line M-E Redo
```

```
root@ip-172-31-26-136: /etc/wireguard  
ubuntu@ip-172-31-26-136:~$ sudo su  
root@ip-172-31-26-136:/home/ubuntu# cd /etc/wireguard/  
root@ip-172-31-26-136:/etc/wireguard# ls  
privatekey  publickey  wg0.conf  
root@ip-172-31-26-136:/etc/wireguard#
```



Habilitación del servicio.

```
root@ip-172-31-26-136:/etc/wireguard# nano wg0.conf
root@ip-172-31-26-136:/etc/wireguard# sudo systemctl enable wg-quick@wg0
sudo systemctl start wg-quick@wg0
```

```
Sudo apt-get update
wg genkey | tee /etc/wireguard/server_private.key | wg pubkey >
/etc/wireguard/server_public.key

sudo nano wg0.conf
[Interface]
Address = 10.0.0.1/24
ListenPort = 51820
PrivateKey = <CLAVE_PRIVADA_DEL_SERVIDOR>
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -A FORWARD -o wg0 -j ACCEPT; iptables -
t nat -A POSTROUTING -o eth0 -j MASQUERADE
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -D FORWARD -o wg0 -j ACCEPT; iptables
-t nat -D POSTROUTING -o eth0 -j MASQUERADE

[Peer]
PublicKey = <CLAVE_PUBLICA_DEL_CLIENTE1>
AllowedIPs = 10.0.0.2/32

[Peer]
PublicKey = <CLAVE_PUBLICA_DEL_CLIENTE2>
AllowedIPs = 10.0.0.3/32

[Peer]
PublicKey = <CLAVE_PUBLICA_DEL_CLIENTE3>
AllowedIPs = 10.0.0.4/32
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -A FORWARD -o wg0 -j ACCEPT; iptables -
t nat -A POSTROUTING -o ens33 -j MASQUERADE
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -D FORWARD -o wg0 -j ACCEPT; iptables
-t nat -D POSTROUTING -o ens33 -j MASQUERADE

sudo systemctl enable wg-quick@wg0
sudo systemctl restart wg-quick@wg0
sudo systemctl start wg-quick@wg0
```

Listado 10: Comandos configuración Wireguard máquina Maestro

```
wg genkey | tee /etc/wireguard/client_private.key | wg pubkey >
/etc/wireguard/client_public.key

sudo nano /etc/wireguard/wg0.conf
[Interface]
Address = 10.0.0.2/24
PrivateKey = <CLAVE_PRIVADA_DEL_CLIENTE>

[Peer]
PublicKey = <CLAVE_PUBLICA_DEL_SERVIDOR>
Endpoint = <IP_PUBLICA_DEL_SERVIDOR>:51820
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 25

sudo systemctl enable wg-quick@wg0
sudo systemctl start wg-quick@wg0
```

Listado 11: Comandos configuración Wireguard máquina cliente



5.4. Creación Página Web

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" href="css/style.css" />
  <link rel="shortcut icon" href="img/logo.png" type="image/x-icon" />
  <title>Index</title>
</head>

<body class="body">
  <!-- Menu principal -->
  <header>
    <div class="menu">
      <input type="checkbox" id="menu-toggle">
      <label for="menu-toggle" class="menu-icon">
        <i class="fas fa-bars"></i>
        <i class="fas fa-times"></i>
      </label>
      
      <nav>
        <ul>
          <li>
            <a href="index.html">Inicio</a>
          </li>
          <li>
            <a href="quienes somos.html">Quienes somos</a>
          </li>
          <li>
            <a href="contactanos.html">Contáctanos</a>
          </li>
          <li>
            <a href="login.html">Log in</a>
          </li>
        </ul>
      </nav>
    </div>
  </header>

  <!-- Presentacion 1 -->
  <div class="primer">
    <div class="txt1">
      <h1>La solución para pequeñas empresas</h1>
      <p>Acceso a todos los datos y posibilidad de cancelar cuando quieras</p>
      <a href="#precios" class="subs">Servicios más populares</a>
      <a href="coming_soon.html" class="subs">Más Información</a>
    </div>
  </div>

  <!-- Zona de los precios de los recursos -->
  <div id="precios">
    <div class="prec1">
      <p class="mes0">Plan Estándar</p>
      <strong class="din0">€59,99 / mes</strong>
      <p class="tt0">Acceso a:</p>
      <ul>
        <li>Cluster de Kubernetes local</li>
        <li>DHCP</li>
        <li>DNS</li>
        <li>Hosting de página web</li>
        <li>VPN</li>
        <li>Cluster de alta disponibilidad</li>
        <li>Almacenamiento en la nube</li>
      </ul>
      <a class="subs0" href="coming_soon.html">¡Suscríbete!</a>
    </div>
    <div class="prec1">
      <p class="mes0">Plan Premium</p>
      <strong class="din0">€79,99 / mes</strong>
      <p class="tt0">Acceso a:</p>
      <ul>
        <li>Cluster de Kubernetes local</li>
        <li>DHCP</li>
        <li>DNS</li>
        <li>Hosting de página web</li>
      </ul>
    </div>
  </div>
```



```
<li>VPN</li>
<li>Cluster de alta disponibilidad</li>
<li>Almacenamiento en la nube</li>
</ul>
<a class="subs0" href="coming_soon.html">¡Suscribete!</a>
</div>
<div class="prec1">
<p class="mes0">Plan Anual</p>
<strong class="din0">€779,99 / año</strong>
<p class="tt0">Acceso a:</p>
<ul>
<li>Cluster de Kubernetes local</li>
<li>DHCP</li>
<li>DNS</li>
<li>Hosting de página web</li>
<li>VPN</li>
<li>Cluster de alta disponibilidad</li>
<li>Almacenamiento en la nube</li>
</ul>
<a class="subs0" href="coming_soon.html">¡Suscribete!</a>
</div>
</div>

<!-- Presupuesto gratuito -->
<div class="contenedor">
<form class="formulario2">
<div class="txt2">
<h1>¡PRESUPUESTO GRATUITO!</h1>
</div>
<button class="button">
<a class="log" href="coming_soon.html">¡Presupuesto gratuito!</a>
</button>
</form>
</div>

<!-- Información de servicios -->
<div class="formulario3">
<table class="latabla">
<tr>
<td>
<h1>Somos tu empresa de mantenimiento informático, Soporte IT y proveedor de servicios.</h1>
</td>
<td>
<h3>¡Un sinfín de servicios informáticos!</h3>
<ul>
<li>Mantenimiento informático</li>
<li>Consultoría tecnológica e informática</li>
<li>Servicios IT para empresas</li>
<li>Soporte técnico</li>
<li>Alta disponibilidad</li>
</ul>
</td>
</tr>
</table>
<div class="formulario2texto">
INFORMACIÓN PROTECCIÓN DE DATOS DE GOLDEN ARROW S.L. Finalidades: Responder a sus solicitudes y remitirle información
comercial de nuestros productos y servicios,
incluso por correo electrónico. Legitimación: Consentimiento del interesado. Destinatarios: No están previstas cesiones
de datos. Derechos: Puede retirar su consentimiento
en cualquier momento, así como acceder, rectificar, suprimir sus datos y demás derechos en thegoldenarrow@services.com.
Información Adicional: Puede ampliar la información en el
enlace de Avisos Legales.
</div>
</div>

<!-- Footer -->
<footer>
<div class="footdiv"><span><a href="coming_soon.html">Políticas de Privacidad</a></span></div>
<div class="footdiv"><span><a href="coming_soon.html">Condiciones de contratación</a></span></div>
<div class="footdiv"><span><a href="coming_soon.html">Política de cookies</a></span></div>
<div class="footdiv"><span><a href="coming_soon.html">Aviso Legal</a></span></div>
<div class="footdiv"><span><a href="contactanos.html">Ayuda</a></span></div>
<div class="footdiv"><span><a href="coming_soon.html">Blog</a></span></div>
</footer>
</body>

</html>
```

Listado 12: Código Index.html



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" href="css/style.css" />
  <link rel="shortcut icon" href="img/logo.png" type="image/x-icon" />
  <title>Coming Soon</title>
</head>

<body class="body">
  <header>
    <div class="menu">
      
      <nav>
        <ul>
          <li><a href="index.html">Inicio</a></li>
          <li><a href="#precios">Precios</a></li>
          <li><a href="quienes_somos.html">Quienes somos</a></li>
          <li><a href="contactanos.html">Contáctanos</a></li>
          <li><a href="login.html">Log in</a></li>
        </ul>
      </nav>
    </div>
  </header>

  <main>
    <h1 class="coming">Página en mantenimiento</h1>
    <br>
    <h4>Agradecemos su espera</h4>
  </main>
</body>

</html>
```

Listado 13: Código Coming_soon.html



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" href="css/style.css" />
  <link rel="shortcut icon" href="img/logo.png" type="image/x-icon" />
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css/all.css" />
  <title>Contáctanos</title>
</head>

<body class="bodylogin">
  <header>
    <div class="menu">
      <input type="checkbox" id="menu-toggle">
      <label for="menu-toggle" class="menu-icon">
        <i class="fas fa-bars"></i>
        <i class="fas fa-times"></i>
      </label>
      
      <nav>
        <ul>
          <li><a href="index.html">Inicio</a></li>
          <li><a href="quienes_somos.html">Quienes somos</a></li>
          <li><a href="contactanos.html">Contáctanos</a></li>
          <li><a href="login.html">Log in</a></li>
        </ul>
      </nav>
    </div>
  </header>

  <div class="contenedor">
    <form class="formulario" action="mailto:thegoldenarrowsevice@gmail.com" method="POST"
    enctype="text/plain">
      <div>Para cualquier duda o problema acerca de nuestros servicios contáctenos directamente
      mandando un correo a nuestro equipo de atención al cliente. Si tienes una
      serie de características únicas ponte en contacto con nosotros para así de forma
      completamente gratuita hacerte un plan a medida.</div>
      <input type="submit" value="Abrir ticket" class="button">
    </form>
  </div>
</body>

</html>
```

Listado 14: Código contáctanos.html



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="css/style.css">
  <link rel="shortcut icon" href="img/logo.png" type="image/x-icon" />
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css/all.css">
  <title>Login</title>
</head>

<body class="bodylogin">
  <header>
    <div class="menu">
      <input type="checkbox" id="menu-toggle">
      <label for="menu-toggle" class="menu-icon">
        <i class="fas fa-bars"></i>
        <i class="fas fa-times"></i>
      </label>
      
      <nav>
        <ul>
          <li><a href="index.html">Inicio</a></li>
          <li><a href="quienes_somos.html">Quienes somos</a></li>
          <li><a href="contactanos.html">Contáctanos</a></li>
          <li><a href="login.html">Log in</a></li>
        </ul>
      </nav>
    </div>
  </header>

  <div class="contenedor">
    <form class="formulario">
      <h1 class="formh1">Login</h1>
      <div class="input-contenedor">
        <i class="fas fa-envelope icon"></i>
        <input type="email" placeholder="Correo Electrónico" required>
      </div>
      <div class="input-contenedor">
        <i class="fas fa-key icon"></i>
        <input type="password" placeholder="Contraseña" required>
      </div>
      <button class="button"><a class="log" href="Coming_soon.html">Login</a></button>
      <p>Al registrarte, aceptas nuestras <a href="Condiciones_de_uso.html">Condiciones de uso</a> y <a href="Politica_de_privacidad.html">Política de privacidad</a>.</p>
      <p>¿No tienes una cuenta? <a class="link" href="Registro.html">Regístrate</a></p>
    </form>
  </div>
</body>

</html>
```

Listado 15: Código login.html



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="css/style.css">
  <link rel="shortcut icon" href="img/logo.png" type="image/x-icon" />
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css/all.css">
  <title>Quienes Somos</title>
</head>

<body class="bodyquienes">
  <header>
    <div class="menu">
      <input type="checkbox" id="menu-toggle">
      <label for="menu-toggle" class="menu-icon">
        <i class="fas fa-bars"></i>
        <i class="fas fa-times"></i>
      </label>
      
      <nav>
        <ul>
          <li><a href="index.html">Inicio</a></li>
          <li><a href="quienes_somos.html">Quienes somos</a></li>
          <li><a href="contactanos.html">Contáctanos</a></li>
          <li><a href="login.html">Log in</a></li>
        </ul>
      </nav>
    </div>
  </header>

  <div class="quienes">
    <div class="divquien">
      <span>Nuestra empresa, formada por cuatro compañeros de la universidad, nació como un proyecto final de grado y se ha convertido en una entidad sólida y comprometida con la innovación en el sector de servicios informáticos. Desde nuestros primeros días en la facultad, compartimos la pasión por la tecnología y una visión común: simplificar y optimizar la infraestructura IT para empresas de todos los tamaños.</span>
    </div>
    <div class="divquien">
      <span>Nos especializamos en ofrecer soluciones basadas en Kubernetes, una plataforma líder en la gestión de contenedores que garantiza la escalabilidad y eficiencia de los servicios. Proveemos una amplia gama de servicios esenciales como VPN, DHCP y DNS, diseñados para mejorar la seguridad, conectividad y gestión de redes de nuestros clientes. Nuestro enfoque se centra en la automatización y la resiliencia, permitiendo a las empresas concentrarse en su crecimiento mientras nosotros nos ocupamos de su infraestructura tecnológica.</span>
    </div>
    <div class="divquien">
      <span>Estamos orgullosos de nuestro origen académico y del rigor técnico que esto implica. Cada miembro de nuestro equipo aporta su experiencia y conocimientos para crear soluciones innovadoras y personalizadas. Nos dedicamos a brindar un servicio excepcional y a construir relaciones duraderas con nuestros clientes, ayudándoles a navegar y prosperar en el complejo mundo de la tecnología moderna.</span>
    </div>
  </div>
</body>

</html>
```

Listado 16: Código quienes_somos.html



XXIX



```
/* General Styles */
* {
  margin: 0;
  padding: 0;
  text-decoration: none;
  text-align: center;
  font-family: Georgia, 'Times New Roman', Times, serif;
}

/* Body Background for Different Pages */
.body, .bodylogin, .bodyquienes {
  background: url(..img/fondo.jpg) no-repeat fixed;
  background-size: cover;
}

/* Link Styles */
a {
  color: white;
}

/* Table Style */
.latabla {
  background-color: #030d1fd5;
  box-shadow: 0px 3px 3px 1px rgb(12, 83, 190);
  border-radius: 20px;
}

/* Header and Navigation Menu */
.logo {
  height: 60px;
  float: left;
}

header {
  width: 100%;
  overflow: hidden;
  height: 60px;
  background-color: #af7a08;
}

.menu input[type="checkbox"] {
  display: none;
}

.menu .fa-bars, .menu .fa-times {
  font-size: 24px;
  cursor: pointer;
}

nav {
  position: absolute;
  top: -20px;
  left: 0;
  right: -15%;
  margin: 20px auto;
  width: 80%;
}

nav ul {
  list-style: none;
}

nav > ul {
  display: table;
  width: 100%;
  background-color: #af7a08;
}

nav > ul li {
  display: table-cell;
}

nav > ul > li > ul {
  display: block;
  position: absolute;
  background-color: black;
  left: 0;
  right: 0;
  height: 0;
  overflow: hidden;
  transition: height 0.3s ease;
}
```



```
}

nav > ul li a {
  color: white;
  display: block;
  line-height: 20px;
  padding: 20px;
  position: relative;
  text-align: center;
  transition: all 0.3s ease;
}

nav > ul > li a span {
  background-color: #af7a00;
  display: block;
  height: 100%;
  width: 100%;
  position: absolute;
  top: -60px;
  left: 0;
  transition: top 0.3s ease;
  line-height: 60px;
}

nav > ul > li a:hover > span {
  top: 0;
}

/* Index Page Sections */
.primer {
  color: white;
  width: 100%;
  height: 600px;
  background: url(..img/banner.png) no-repeat center;
  background-size: cover;
}

.txt1 {
  position: relative;
  top: 40%;
  padding-left: 70%;
  width: 400px;
}

.txt1 p, .txt1 h1 {
  text-align: left;
}

.txt1 .subs {
  background-color: blue;
  color: white;
  border-radius: 20px;
  padding: 8px 20px;
  transition: 0.2s;
}

.txt1 .subs:hover {
  box-shadow: 0px 3px 3px 1px rgb(71, 71, 71);
}

.txt2 {
  position: relative;
  top: 10%;
  padding-left: 20%;
  width: 350px;
}

.txt2 h1 {
  color: white;
}

/* Pricing Section */
#precios {
  display: flex;
  width: 100%;
  height: 520px;
  padding-bottom: 5%;
  align-content: center;
  top: 80px;
}
```



```
#imgprecios {
  display: inline-block;
  padding-left: 10%;
}

#precios h2, #precios h3 {
  color: #ddb96b;
}

#precios h2 {
  font-size: 30px;
  padding-bottom: 20px;
  padding-left: 30%;
}

#precios h3 {
  font-size: 20px;
  padding-left: 30%;
  top: 50px;
}

#precios .prec1 {
  display: inline-block;
  width: 350px;
  height: 400px;
  border: 1px solid black;
  margin: auto;
  border-radius: 5px;
  background-color: rgb(230, 229, 229);
  transition: transform 0.2s;
}

#precios .prec1:hover {
  transform: scale(1.1);
}

#precios .prec1 .mes0 {
  font-size: 20px;
  padding: 10px;
  padding-top: 40px;
}

#precios .prec1 .din0 {
  font-size: 30px;
  color: #af7a08;
  padding: 10px;
}

#precios .prec1 .tt0, #precios .prec1 .tt01 {
  font-size: 20px;
  padding-bottom: 5px;
}

#precios .prec1 .tt01 {
  text-decoration: line-through;
}

#precios .prec1 .subs0 {
  background-color: blue;
  color: white;
  border-radius: 20px;
  padding: 8px 20px;
  transition: 0.2s;
}

#precios .prec1 .subs0:hover {
  box-shadow: 0px 3px 3px 1px rgb(71, 71, 71);
}

/* Footer */
footer {
  background-color: #af7a08;
  display: flex;
}

footer > div span {
  padding: 0;
}

.footdiv {
  display: inline-block;
```



```
width: 17%;
padding: 10px 0;
border: 2px solid #af7a08;
}

/* Login and Registration Pages */
.bodylogin {
background: url(..img/fondo.jpg) no-repeat fixed;
background-size: cover;
}

.contenedor {
display: flex;
align-items: center;
justify-content: center;
width: 98%;
padding: 15px;
height: 100%;
}

.formulario {
background: #ffffffa2;
margin-top: 150px;
padding: 20px;
border-radius: 5px;
}

.formulario h1 {
color: #473a0f;
font-size: 40px;
}

.formulario input[type="text"],
.formulario input[type="password"] {
font-size: 20px;
width: 82%;
padding: 10px;
border: none;
}

.input-contenedor {
margin-bottom: 15px;
}

.icon {
min-width: 50px;
text-align: center;
color: #999;
}

.button {
border: none;
width: 100%;
font-size: 20px;
background: #473a0f;
padding: 15px 20px;
border-radius: 5px;
cursor: pointer;
color: white;
}

.button:hover {
background: cadetblue;
}

p {
text-align: center;
}

.link {
color: #473a0f;
font-weight: 600;
}

.link:hover {
color: cadetblue;
}

.log {
color: white;
}
```



```
/* Quienes Somos Page */
.bodyquienes {
  background: url(../img/fondo.jpg) no-repeat fixed;
  background-size: cover;
}
```

```
.quienes {
  width: 98%;
  margin: 20px;
  height: auto;
}
```

```
.divquien {
  width: 50%;
  margin-top: 15px;
  padding-left: 23%;
  display: flex;
  height: 250px;
}
```

```
.divquien span {
  font-weight: 600;
  font-size: large;
  align-self: center;
  padding-inline-start: 5%;
}
```

```
/* Coming Soon Page */
.coming {
  padding-top: 300px;
}
```

```
/* Media Queries for Responsive Design */
@media (min-width: 576px) {
  .formulario {
    width: 500px;
    margin: 150px auto;
  }

  .formulario2, .formulario3, .formulario2texto {
    width: 500px;
    margin: 50px auto;
    border-radius: 2%;
  }

  .formulario3 {
    width: 600px;
  }

  .formulario2texto {
    width: 600px;
    color: white;
  }
}
```

```
@media (max-width: 576px) {
  .divquien {
    width: 70%;
    text-align: center;
    margin-top: 1%;
    display: inline-block;
  }

  .divquien span {
    font-size: medium;
  }

  .letras {
    display: inline-block;
    text-align: justify;
    padding-top: 0;
  }
}
```

```
@media (max-width: 1100px) {
  .menu input[type="checkbox"] {
    display: block;
    opacity: 0;
  }
}
```



```
.menu .fa-bars, .menu .fa-times {
  position: absolute;
  right: 0;
  top: 0;
  width: 48px;
  height: 48px;
  opacity: 1;
  font-size: 48px;
}

.menu nav {
  display: none;
}

.menu input:checked ~ nav {
  display: block;
}

.menu input:checked ~ .fa-bars {
  display: none;
}

.menu input:not(:checked) ~ .fa-times {
  display: none;
}

/* Index Page Adjustments */
.primar {
  background-position: 30%;
  background-size: 200%;
}

.ultimos {
  display: inline;
}

.ultimos .curs1, .ultimos .curs2, .ultimos .curs3 {
  position: relative;
  top: 80px;
}

.ultimos .curs3 {
  margin-bottom: 100px;
}

/* Pricing Section Adjustments */
#precios {
  display: inline-block;
  height: 1200px;
}

#precios h2, #precios h3 {
  padding-left: 10%;
}

#precios .prec1, #precios .prec2, #precios .prec3 {
  position: relative;
  top: 150px;
}

header {
  width: 100%;
}

.primar {
  height: 550px;
}
}
```

Listado 18: Código style.css