Institut Puig Castellar
Santa Coloma de Gramenet

Marenostrum 0,000001

(Proyecto de desarrollo)

CFGS Administració de Sistemes Informàtics i Xarxes

Sabas Alcázar Romero
Antonio Tomás Franco
2° ASIX
2019-2020

Resumen del proyecto:

En una realidad cada vez más tecnológica y enfocada en servicios alojados en la nube, Marenostrum 0,000001 trata de llevar esta tecnología de servicios conectados a un entorno más desfavorecido.

La intención de este proyecto es llevar la creación de páginas web y microservicios, así como el acceso a internet, a entornos en los que la tecnología presente no permite a la población el desarrollo de sus capacidades creativas aplicadas a entornos web ni el uso de estas herramientas.

Para conseguirlo, hemos apostado por el uso de un pequeño *cluster* escalable de Raspberry Pi que permite construir una estructura de microservicios usando la tecnología Docker y Kubernetes o desplegar un punto de acceso a Internet cambiando las tarjetas micro SD que alojan el sistema operativo.

Esta metodología permitiría a varios usuarios tener un espacio en el que alojar sus sitios y/o aplicaciones web, pudiendo escalar fácilmente el sistema y ofrecer más potencia de cálculo, o facilitar el acceso a Internet a pequeños grupos poblacionales. Todo ello con un coste mínimo gracias a Raspberry Pi.

Este proyecto demuestra que es posible crear pequeños *clouds* o servidores destinados a reducidos grupos poblacionales para que estos puedan acceder a Internet con un punto de acceso de tamaño reducido o alojar sitios web que les permitan desde darse a conocer hasta desarrollar un negocio basado en internet, pasando por la enseñanza de tecnologías web en entornos educativos.

Palabras clave:

Raspberry Pi, Kubernetes, Docker, Web, Internet, Cluster.

Abstract:

In an increasingly technological and cloud services focused reality, Marenostrum 0.000001 tries to bring this technology of connected services to a more disadvantaged environment.

The intention of this project is to bring the creation of web pages and microservices, as well as Internet connection, to environments in which the present technology does not allow the population to develop their creative abilities applied to web environments or the use of these tools.

To achieve this, we have opted for the use of a small scalable Raspberry Pi cluster that allows building a microservices structure using Docker and Kubernetes technology or deploying an Internet access point by changing the micro SD cards that host the operating system.

This methodology would allow several users to have a space in which to host their sites and / or web applications, being able to easily scale the system

and offer more computing power, or facilitate Internet access for small population groups. All this with a minimum cost thanks to Raspberry Pi.

This project demonstrates that it is possible to create small clouds or servers aimed at small population groups so that they can access the Internet with a small access point or host websites that allow them from making themselves known to develop an internet-based business, including teaching web technologies in educational environments.

Keywords:

Raspberry Pi, Kubernetes, Docker, Web, Internet, Cluster.

Índex

Lista de figuras

1 Introducción

La época en que vivimos se caracteriza por estar inmersa en una revolución tecnológica cuyo impacto será realmente visible dentro de algún tiempo. En el inicio de la segunda década del siglo 21, casi la mitad de la población mundial dispone de un smartphone. Esto significa que más de 3500 millones de personas en el mundo tienen en su bolsillo un dispositivo con capacidad de conectarse a Internet, de ejecutar aplicaciones de todo tipo y de mostrar contenido multimedia.

Sin embargo, el acceso a Internet de muchos países en vías de desarrollo está lejos de ser un derecho, y aunque está demostrada la inmensa capacidad de creación de oportunidades que esta tecnología posee, es ahora, en 2020, cuando la tecnología 3G está llegando a los 47 territorios incluidos en la lista de los denominados países del tercer mundo.

Esto significa que más de 880 millones de personas no tienen la oportunidad de formar parte de la revolución tecnológica y humana llamada Internet. En otras palabras, casi el 12% de la población mundial no tiene la posibilidad de utilizar la más poderosa herramienta de comunicación y creación de oportunidades de la historia de la humanidad. Por suerte, gracias a organizaciones como la ONU, esta parte de la población conseguirá, poco a poco, acercarse en materia de comunicaciones por red al resto de países del planeta.

Siendo conocedores de esta realidad tan injusta, nuestro propósito con este proyecto es el de acercar una pequeña parte de la tecnología necesaria para el acceso, uso y creación de recursos de Internet a una parte de la población que por sus características tiene un difícil acceso a las herramientas más usadas, pero a la vez más caras, como pueden ser los servidores web.

Para conseguir nuestro objetivo, hemos confiado en un conjunto de máquinas asequibles Raspberry Pi de última generación, que ofrecen una relación calidad/precio fuera de toda duda. Además, gracias a su tamaño, posibilidades y apoyo de la comunidad, construir *clusters* escalables que puedan cubrir las necesidades más básicas de la población más desfavorecida es una tarea más fácil y asequible que hace tan solo 10 años.

Como hemos dicho, el enorme apoyo de la comunidad que tienen estas tarjetas hace posible llevar a cabo multitud de proyectos diferentes, tanto de índole profesional, como de índole personal o educativa. Y son precisamente las peculiares características del almacenamiento nativo de Raspberry Pi las que nos han brindado la idea de cambiar la funcionalidad de este conjunto de microordenadores mediante el intercambio de tarjetas micro SD para cubrir un mayor abanico de necesidades.

Así, una comunidad puede tener listo un punto de acceso a Internet o una nube en la que alojar aplicaciones y/o microservicios en minutos tan sólo cambiando la tarjeta que contiene el sistema operativo.

1.1 Contexto

En la actualidad, el mercado de los ordenadores de placa simple (o SBC, *Single Board Computer*) es cada vez más amplio y popular, lo que ha dado lugar a la proliferación de

numerosos ordenadores asequibles de tamaño muy reducido y de capacidades técnicas muy altas para el precio al que se venden. Esto ha dado lugar a la creación de miles de proyectos, profesionales y aficionados, que hace tan sólo diez años eran impensables. Así, estas placas prodigiosas han dado vida a robots, escáneres 3D, estaciones meteorológicas, sistemas domóticos, *clouds* privados y hasta cámaras inteligentes que son capaces de reconocer qué objeto tienen delante.

Por otra parte, Internet cada vez está más presente, el acceso a la red de redes está cada vez más extendido y las clásicas webs han ido dado paso a las aplicaciones web, que están alojadas en servidores de miles de euros, dan servicio a cientos de miles de personas a la vez, están divididas en pequeñas partes y ofrecen una fiabilidad y disponibilidad cada vez mayor.

1.2 Justificación

Raspberry Pi es un ordenador de placa simple veterano, muy asentado en el mercado y con un gran apoyo por parte de la empresa desarrolladora y de la comunidad que tiene detrás, por lo que es una apuesta segura para embarcarse en proyectos asequibles que pueden ser escalados fácilmente y con un coste mucho menor que con ordenadores más potentes, pero también más caros.

Estas placas, además, disponen de dos tarjetas de red muy rápidas y procesadores con varios núcleos que permiten una concurrencia de conexiones lo suficientemente elevada como para funcionar como punto de acceso en un espacio público.

Docker y Kubernetes, por su parte, son tecnologías relativamente nuevas pero lo suficientemente maduras, desarrolladas y apoyadas como para ser tenidas en cuenta a la hora de dar vida a la parte de microservicios de nuestro proyecto. Su enorme versatilidad hace posible migrar a estas tecnologías cualquier aplicación existente que utilice un servidor web, y existen múltiples empresas no sólo utilizando estas tecnologías, sino desarrollando para ellas.

Por lo tanto, la elección de estas tecnologías tan apoyadas, flexibles y escalables, junto con otras auxiliares que completan el conjunto, pueden hacer posible realizar este proyecto con garantías.

1.3 Objetivos

1.3.1 Objetivo general

Creación de un *cluster* de placas Raspberry Pi 4 que permita dar acceso a Internet a pequeñas comunidades desfavorecidas y que también pueda ser usado como *cluster* Kubernetes capaz de manejar automáticamente contenedores Docker que funcionen como servicio de hosting WordPress para países en vías de desarrollo.

1.3.2 Objetivos específicos

- Creación de un *cluster* de 4 x Raspberry Pi modelo 4B de 4GB de RAM funcionando de manera sincronizada como una sola unidad.
- Creación de un *cluster* Kubernetes con capacidad de desplegar *pods* personalizados con contenedores Docker y unidades de almacenamiento persistente.
- Exposición pública de dichos *pods*.
- Gestión automática de *pods* para garantizar la disponibilidad operativa.
- Creación de un sitio web que permita a los usuarios la creación de espacios de *hosting* WordPress que puedan utilizar para desplegar sus recursos.
- Creación de punto de acceso a Internet para pequeños grupos utilizando tecnologías como DNSMasq.

1.4 Estrategia y planificación del proyecto

Para llevar a cabo este proyecto se ha optado por adaptar tecnologías y productos existentes en el mercado. Esta decisión se ha tomado en base al apoyo y documentación que están disponibles de manera pública, lo que facilita en cierta medida la configuración y personalización requeridas para alcanzar unos objetivos asumibles con el tiempo de que se dispone.

1.5 Metodología de trabajo

La metodología a seguir durante el desarrollo será de tipo Scrum. Dada la naturaleza del equipo de desarrollo y las particularidades del proyecto, es la metodología más conveniente, pues permite un flujo de trabajo constante y un seguimiento periódico de resultados.

1.6 Estudio económico y presupuestario

El coste estimado para la realización de este proyecto con garantías es de unos 360 euros. Este coste se desglosa de la siguiente manera:

- Raspberry Pi de 4GB de RAM (4 x 59.95€)
- Tarjetas micro SD de 32GB (4 x 6.28€)
- Cable micro HDMI (2 x 5.95€)
- Fuentes de alimentación DC USB C 5V 3A (4 x 8.95€)
- Estructura de metacrilato (9.95€)
- Conjunto de ventilador y disipadores (4 x 2,95€)
- Switch Gigabit Ethernet 8 bocas (18.95€)
- Cables de red Ethernet (5 x 1.10€)

2 Descripción del proyecto

2.1 Análisis de requisitos

Marenostrum 0,000001 debe cumplir con las siguientes condiciones para considerarse finalizado:

- Cualquier usuario debe poder registrarse y crear un nuevo espacio WordPress personalizado en el servidor haciendo uso del sitio web creado para tal efecto.
- Dichos espacios deben de ser creados utilizando la tecnología Kubernetes y Docker con total transparencia para el usuario.
- Estos espacios deben ser seguros y estar disponibles para su utilización sin ningún tipo de contratiempo.
- Cada usuario debe poder gestionar su espacio WordPress de manera transparente.
- Debe ser posible monitorizar el estado físico del servidor, así como su seguridad, en cualquier momento.
- Debe ser posible poner en marcha un punto de acceso a Internet en poco tiempo cambiando sólo la tarjeta micro SD que contiene el sistema operativo.

2.1.1 Requisitos funcionales

El proyecto debe cumplir con los siguientes requisitos funcionales:

- La integración entre el sitio web y el sistema subyacente debe ser completo para que los espacios WordPress solicitados por el cliente puedan ser creados de manera automática si se cumplen ciertas condiciones establecidas previamente.
- Un usuario debe ser capaz de modificar sus datos, así como eliminarlos si fuera necesario, sin ningún impedimento y siguiendo las normas y legislaciones de privacidad vigentes.
- El sistema informático debe ser capaz de establecer un límite en la cantidad de espacios que puede soportar de manera automática para preservar su funcionalidad de manera automática.
- Debe ser posible recibir alertas automáticas en caso de darse funcionamientos anómalos del sistema (carga elevada de CPUs, poco espacio en disco, temperatura elevada, ataques externos, etcétera).
- En caso de usar el *cluster* como punto de acceso a Internet, éste debe ser resistente a ataques, además de ser seguro para los usuarios.

2.1.2 Requisitos no funcionales

El proyecto debe cumplir con los siguientes requisitos no funcionales:

- El sitio web utilizado para interactuar con los usuarios debe ser claro y fácil de usar.
- La seguridad de los datos de los usuarios debe estar garantizada.

- La seguridad del sistema debe ser lo suficientemente sólida como para detectar y repeler accesos y ataques no deseados.

2.2 Previsión de tareas de investigación

Es necesario investigar el funcionamiento e integración de los siguientes componentes para la realización de este proyecto con garantías:

- Docker: conocer su funcionamiento y qué son las imágenes y contenedores Docker.
- Kubernetes: conocer su funcionamiento, cómo funcionan las numerosas partes que lo conforman y cómo crear un *cluster* funcional integrando todos sus componentes.
- WordPress: conocer cómo generar e integrar espacios WordPress dentro de Kubernetes y hacerlos accesibles a los usuarios finales.
- PHP: conocer cómo interactuar con el sistema a través de un sitio web para la creación de espacios Kubernetes-Docker.
- Raspberry Pi como punto de acceso: conocer cómo crear un punto de acceso robusto en base a varias tarjetas Raspberry Pi interconectadas.

2.3 Tecnologías

2.3.1 Comparativa de las tecnologías valoradas

En la siguiente tabla comparativa se muestran las dos opciones barajadas para la implantación de orquestación de contenedores:

Kubernetes (K8s)	K3s
Sistema de orquestación de contenedores desarrollado por Google	Sistema de orquestación de contenedores desarrollado por Rancher Labs a partir de Kubernetes
Recursos necesarios elevados	Recursos necesarios bajos
Posibilidad de tener más de un nodo maestro	Sólo es posible tener un nodo maestro en el sistema
Funcionalidad completa	Funcionalidad limitada sin la instalación de plugins
Enfocado a entornos de producción	Enfocado a entornos de test
Puesta en marcha compleja	Puesta en marcha simple

2.3.2 Tecnologías escogidas

Finalmente, la tecnología elegida para la orquestación de contenedores ha sido Kubernetes (K8s) por su cercanía a los entornos de producción. Durante la investigación de Kubernetes y su funcionamiento se eligió Weave Net como API para la comunicación de red entre nodos debido a que ofrece más posibilidades de configuración que otras soluciones. Como balanceador de carga se eligió MetalLB por ser la única opción factible para montar un *cluster* sobre metal en lugar de sobre servicios *cloud* como Amazon Web Services o Google Cloud. Para el control de ingreso y exposición de *pods* fuera de la red del *cluster* se ha elegido Nginx Ingress Controller por ser una opción extendida y bastante documentada.

En cuanto a la tecnología de contenedores, desde el principio se eligió Docker debido a su apoyo, popularidad y compatibilidad completa con Kubernetes.

Para el despliegue del entorno web se ha utilizado la pila LAMP (Linux, Apache, MySQL y PHP) la cual hemos instalado y configurado por separado. Esta combinación de programas *open source* hace posible la creación y el alojamiento de paginas web dinamicas.

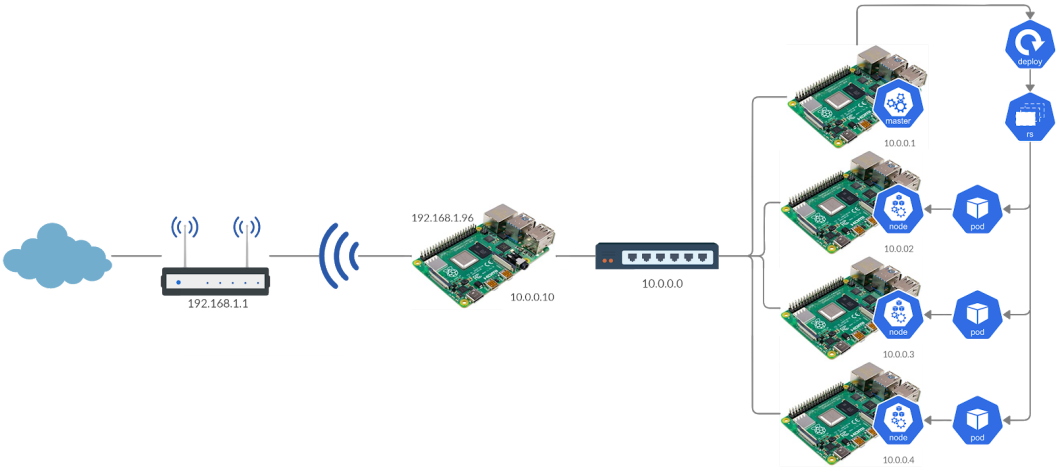
Como sistema gestor de contenido se ha optado por WordPress, ya que brilla por su sencillez a la hora de utilizarse, aparte Wordpress es open source, por lo tanto no habrá gastos por licencia y estará respaldado por una gran comunidad de la cual podrás obtener un sin fin de tutoriales, posé un código fuente completo revisado que lo hace muy seguro y fiable, contiene un sistema de plugins que lo hace una herramienta muy potente y flexible, Wordpress ofrece una puntuación alta en los motores de búsqueda, por eso es una de las plataformas favorita de los expertos en SEO .

2.4 Estructura del proyecto

En la parte *frontend* una página web creada con WordPress da la posibilidad a cualquier usuario de registrarse y solicitar un subdominio y un espacio *hosting* en el servidor que le permita crear su propia página web utilizando también WordPress.

Automáticamente, en el *backend*, se crea un subdominio asociado a un espacio WordPress funcionando sobre Kubernetes y Docker que el usuario registrado puede usar para dar vida a su página web y exponerla al público.

En el *backend*, el *cluster* Kubernetes se organiza de manera que una de las placas Raspberry Pi tiene asignada el rol de nodo maestro o *master*, mientras que las tres restantes toman el papel de nodos trabajadores o *workers*.



2.5 Descripción de los componentes

2.5.1 Raspberry Pi 4 Model B

Microordenador en formato de placa simple que integra una CPU Broadcom BCM2711 Cortex-A72, una GPU Broadcom VideoCore VI, una interfaz Gigabit Ethernet, una interfaz WiFi IEEE 802.11ac, Bluetooth 5.0, 2 puertos USB 3.0, 2 puertos USB 2.0, un lector micro SD y 2 puertos micro HDMI.

2.5.2 Kubernetes

Kubernetes funciona con el denominado Plano de Control, formado por el nodo *master* y el proceso kubelet, presente en todos los nodos. Estos componentes determinan cómo Kubernetes se comunica con el *cluster*. El Plano de Control posee un registro de todos los objetos Kubernetes presentes en el *cluster* y ejecuta bucles de control con el objetivo de gestionar sus estados.

El nodo maestro de Kubernetes ejecuta tres procesos fundamentales: kube-apiserver, kube-controller-manager y kube-scheduler. Con estos procesos, el nodo maestro actúa como orquestador del conjunto, asignando los *pods* a los nodos trabajadores disponibles en función de los recursos disponibles en cada uno. El nodo maestro es el encargado de mantener el estado deseado de un *cluster*. La interacción con el nodo maestro se lleva a cabo con la interfaz Kubectl.

Los nodos trabajadores ejecutan el proceso kubelet, y son las máquinas (virtuales o físicas) que ejecutan las aplicaciones. Son controladas por el nodo maestro, por lo que no es necesario interactuar con estas máquinas directamente.

Existen dos maneras de gestionar un *cluster* Kubernetes: imperativa o declarativa. La manera imperativa usa la interfaz Kubectl junto con los comandos `run`, expose, autoscale o create (entre algunos otros) para crear objetos Kubernetes. Esta tarea también se puede realizar con determinados ficheros de configuración. La manera declarativa utiliza la interfaz Kubectl junto con el comando apply para gestionar el *cluster* mediante ficheros de configuración.

La diferencia de concepto entre la gestión imperativa y la gestión declarativa reside en ordenar al *cluster* la creación de ciertos objetos Kubernetes con el primer tipo de gestión

o indicar al *cluster* qué objeto Kubernetes queremos crear y en qué estado desearíamos que estuviera con el segundo.

En cuanto al funcionamiento interno de Kubernetes, es imprescindible saber que esta tecnología funciona con determinados objetos y controladores, que son los que finalmente se gestionan en los nodos.

Así, existen cuatro objetos básicos y cinco controladores fundamentales:

Objetos:

- Pod: Es la unidad más básica, pequeña y simple del modelo de objetos de Kubernetes. Representa un proceso en ejecución en el *cluster* y es la encapsulación del contenedor de una aplicación (aunque en alguna ocasión puede contener más de un contenedor), recursos de almacenamiento, una dirección IP única y algunas opciones que determinan cómo debe funcionar.
- Service: Capa de abstracción de red utilizada para la comunicación entre pods. La vida de los pods es limitada: se van regenerando periódicamente. Debido a ello, sus direcciones IP van cambiando, lo que hace inviable acceder a dichos pods a través de sus direcciones. Un servicio (o *service*) coloca una capa de red por encima de los pods, proporcionando una IP estable, cuya duración está ligada a la del servicio, y que se puede usar para acceder a un conjunto de pods relacionados mediante el uso de etiquetas.
- Volume: Los datos almacenados en los contenedores no son permanentes y se pierden cuando un contenedor ubicado dentro de un pod debe reiniciarse. Los volúmenes suplen este inconveniente, proporcionando almacenamiento duradero que sobrevivirá al reinicio de un contenedor. Sin embargo, al estar ubicados dentro de pods, la vida de estos volúmenes está ligada a estos.
- Namespace: *Cluster* virtual dentro del *cluster* físico destinado a separar múltiples proyectos. Son una forma de dividir los recursos del *cluster* entre múltiples usuarios. Los nombres de los recursos ubicados dentro de un espacio de nombres deben ser únicos dentro del mismo, no así entre diferentes espacios de nombres.

Controladores:

- ReplicaSet: Es el encargado de crear un número de pods idénticos según lo deseado. Así, si una de las réplicas muere, el nodo maestro volverá a crear una réplica nueva hasta alcanzar el número establecido por un ReplicaSet.
- Deployment: Es el encargado de crear y actualizar pods y ReplicaSets. De manera declarativa, indica el estado deseado de los mismos.
- StatefulSet: Su función es casi idéntica a la de un Deployment, con la principal diferencia de que un StatefulSet está enfocado a aplicaciones persistentes. Otorga un identificador a cada pod para hacer más fácil su enlace a un almacenamiento persistente en caso de que un pod se pierda y tenga que ser sustituido por otro.
- DaemonSet: Se asegura de que los nodos ejecutan una copia de un pod.
- Job: Crea uno o más pods y se asegura de que que un número de ellos termina satisfactoriamente, dejando constancia de las terminaciones exitosas.

Kubernetes está pensado para ser portable y personalizable, por lo que ciertas características y funcionalidades que en sus inicios formaban parte del núcleo ahora han sido delegadas a APIs que desarrolladores externos pueden crear. Así, existen plugins para la comunicación entre nodos, para la comunicación entre pods o para la capa de

almacenamiento persistente. Aunque esto hace posible adaptar Kubernetes a cualquier necesidad, también eleva su complejidad.

2.5.3 Docker

Docker es un software de código abierto que permite crear aplicaciones dentro de contenedores de software. Este sistema permite aislar una aplicación cualquiera que puede utilizar recursos del sistema pero sin la necesidad de tener incluido ninguno, al contrario de lo que ocurre con las máquinas virtuales. Esto aumenta la portabilidad de las aplicaciones, que pueden ser ejecutadas en cualquier sistema Linux sin necesidad de modificaciones, así como la seguridad, ya que aunque un contenedor haga uso de los recursos del sistema, no tiene acceso directo a él, por lo que no puede afectar negativamente al funcionamiento de la máquina.

Su funcionamiento es posible gracias al Docker Engine, que permite crear contenedores en base a imágenes, o imágenes en base a contenedores. Es posible, además, interactuar con las aplicaciones de los contenedores gracias al cliente integrado.

2.5.4 Ansible

Ansible es un software que nos permite gestionar y configurar remotamente nuestro sistema, gestionando sus diferentes nodos a través de SSH.
A continuación vamos a definir sus componentes:

- **Modules**: Los módulos son librerías que utiliza para controlar servicios, ficheros paquetes o comandos.
- **Inventory:** Ficheros donde se definen los host o grupos de hosts y sus variables como podría ser el puerto ssh al que hay que conectarse.
- **Playbook:** Este fichero es el encargado de definir todas las tareas que debemos realizar sobre un conjunto de host, dicho fichero será en formato YAML.

2.5.5 ElasticStack

ElasticStack es un conjunto de software formado por diferentes programas cuya función principal es la de recabar datos de todo tipo del sistema para mejorar y facilitar la monitorización de manera muy granular y personalizable. Está formado por las siguientes piezas de software:

- Beats: programas de recolección y envío de logs hacia Logstash o ElasticSearch.
- Logstash: programa que recoge la información enviada por los Beats y que es capaz de depurarla y transformarla antes de su envío a ElasticSearch.
- ElasticSearch: programa que recoge y guarda la información suministrada por los Beats y/o Logstash en índices que pueden ser usados para generar datos sobre el uso de los diferentes sistemas de los que recoge información.
- Kibana: programa de visualización que utiliza los datos almacenados en ElasticSearch para mostrar paneles que faciliten la monitorización y análisis de sistemas.

2.5.6 Falco

Falco sirve para detectar actividad anómala en sus aplicaciones, le permite monitorear y detectar de una manera permanente contenedores, aplicaciones, hosts y redes. Con un

conjunto de reglas puede detectar y alertar comportamientos que impliquen hacer llamadas al sistema, recopiladas con Sysdig.
Por ejemplo podría detectar cosas como:

- Una shell que se ejecuta dentro de un contenedor
- Un proceso que genere un proceso hijo inesperado
- Lectura inesperada de un archivo sensible
- Un archivo que no sea del dispositivo que escriba en /dev

2.5.7 WordPress

Sistema de gestión de contenidos que permite crear y administrar sitios web de manera intuitiva. Al contrario de lo que sucede con el desarrollo web tradicional, con WordPress la creación de sitios de Internet se lleva a cabo mediante el uso de plantillas y plugins. Con un fuerte apoyo de la comunidad, cualquier persona puede crear plantillas y plugins para WordPress, pudiendo además venderlos en la plataforma.

La creación y gestión de sitios web en WordPress se realiza mediante el *dashboard* de la aplicación, que permite buscar y añadir temas y plugins, así como gestionar usuarios.

WordPress no guarda archivos HTML clásicos, sino que almacena los datos necesarios para la creación de una web en una base de datos y de manera dinámica las crea cuando un cliente solicita una página.

2.6 Definición de las tareas de investigación

2.6.1 Docker

Docker es, junto a Kubernetes, la base de nuestro proyecto. Esta tecnología de contenedores es la utilizada para aislar las aplicaciones web de nuestros usuarios.

2.6.2 Kubernetes

Esta tecnología de orquestación es el cerebro de nuestro proyecto. Kubernetes es capaz de mantener una aplicación en funcionamiento con una alta disponibilidad con total transparencia hacia el usuario gracias a su gestión de los contenedores Docker.

2.6.3 WordPress

Si bien requiere algunos conocimientos y configuración previos, WordPress se puede integrar de manera satisfactoria en un entorno Kubernetes + Docker. Además, es un sistema amigable que permite crear webs vistosas y funcionales a gente que no tiene conocimientos de programación web. Finalmente, en nuestro proyecto lo hemos utilizado para la web de registro y para proporcionar una tecnología de creación de contenidos a los usuarios.

2.6.4 PHP

La integración de PHP con Kubernetes requiere de ciertos conocimientos y configuración adicional de la capa de red para que se pueda comunicar con un servidor Apache ubicado en otro contenedor. Aunque es posible tener más de una aplicación dentro de

un contenedor, lo recomendable es que cada contenedor sólo se haga cargo de un proceso.

2.6.5 Raspberry Pi como punto de acceso

Con el cambio de tarjeta micro SD, nuestro debería aumentar la versatilidad del *cluster* de 4 Raspberry Pi, permitiendo nuevas posibilidades en entornos desfavorecidos.

2.7 Definición de las funcionalidades

2.7.1 Integración entre el sitio web y el sistema subyacente

La idea inicial era que el sitio web utilizado como *frontend* sirviera para automatizar el proceso de registro de usuarios y la creación de sus espacios de *hosting* para que estos pudieran acceder a ellos en un plazo de tiempo corto. Sin embargo, la funcionalidad no ha podido ser implementada en su totalidad y el registro y creación de espacios dentro del *cluster* requiere de la intervención de un administrador. La funcionalidad de la web en el resultado final se limita a recibir datos de usuarios a través de formularios que luego serán utilizados para la creación de espacios WordPress dentro del *cluster*.

2.7.2 Modificación y eliminación de datos de usuarios

En el planteamiento inicial del proyecto, los usuarios debían poder modificar y eliminar sus datos de manera autosuficiente y sin la intervención de un administrador. Esta funcionalidad no ha podido ser implementada y un administrador debe borrar de la base de datos los datos de los usuarios.

2.7.3 Limitación automática de número de espacios de usuarios

No ha sido posible implementar esta funcionalidad. Se iba a llevar a cabo estableciendo mecanismos de monitorización de espacio, limitando la creación de nuevos usuarios y subdominios si el espacio de almacenamiento era crítico.

2.7.4 Sistema de alertas automáticas

La idea inicial contemplaba un sistema automático de alertas basado en ElasticStack y Falco que nos avisara mediante correo electrónico y/o telegram de acontecimientos anómalos dentro del sistema como temperaturas excesivas, almacenamiento escaso o detección de intrusiones. Finalmente no se ha podido implementar esta funcionalidad.

2.7.5 Seguridad del punto de acceso

La seguridad de la funcionalidad de punto de acceso del proyecto debía proteger a los usuarios y al sistema en caso de cualquier tipo de ataque. Finalmente no se ha podido incluir la funcionalidad de punto de acceso, por lo que la seguridad tampoco ha sido implementada.

2.7.6 Facilidad de uso del sitio web

El sitio web ofrece a los usuarios la opción de enviar un formulario de registro en un entorno fácil de usar. Aunque el sitio web no ofrece todas las funcionalidades pensadas inicialmente (explicadas en puntos anteriores), sí es fácil de usar.

2.7.7 Seguridad de los datos de usuario

El sistema debía ser robusto en cuanto a la seguridad de los datos de los usuarios mediante mecanismos de detección y encriptación. Sin embargo, la seguridad del sistema no ha podido ser implementada, por lo que esta funcionalidad no ha sido incluida.

2.7.8 Seguridad del sistema

En el planteamiento inicial, el proyecto debía hacer uso de Falco (o similar) como sistema de detección de intrusiones. No obstante, esta funcionalidad no ha podido ser implementada, por lo que el sistema no es 100% seguro.

3 Errores y problemas encontrados

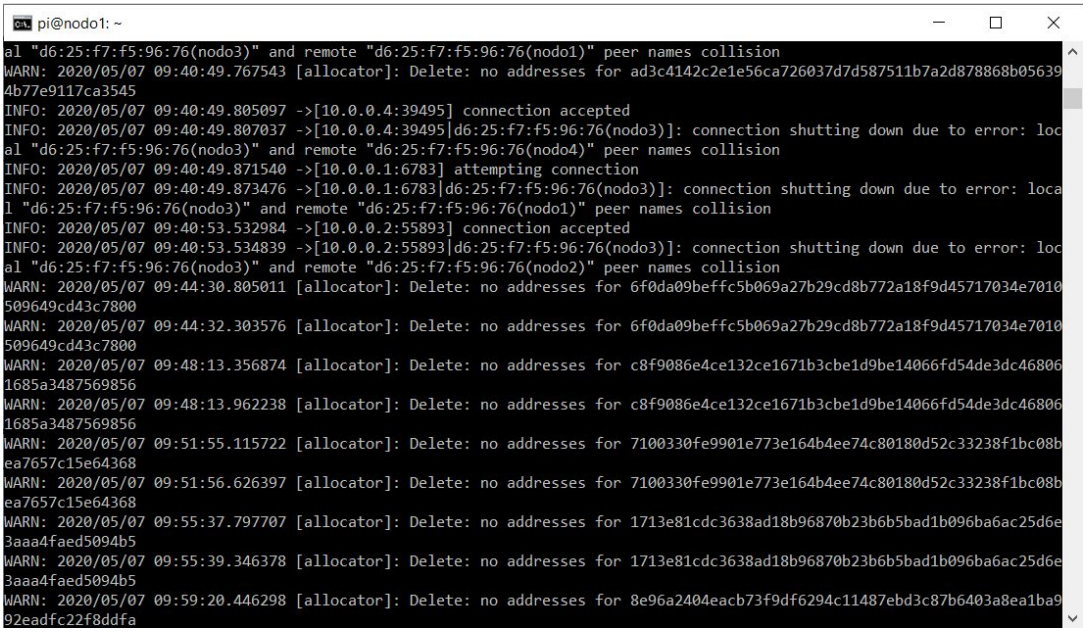
Durante la investigación y desarrollo del proyecto nos hemos encontrado con infinidad de problemas que han lastrado y condicionado la consecución de casi la totalidad de los objetivos.

A la dificultad de aprender e implementar al mismo tiempo varias tecnologías nuevas, que además debían interactuar entre sí sin ningún problema, se han añadido las circunstancias excepcionales por el COVID-19 que han provocado: planificación errónea e insuficiente, falta de tiempo, dificultad para comprender errores y problemas, aprendizaje autodidacta, búsqueda masiva de documentación en inglés para tratar de solucionar problemas…

Kubernetes es un servicio complejo y exigente, que requiere de varias configuraciones diferentes para cada aspecto del servicio. Además, su naturaleza permite que haya varios plugins diferentes para funcionalidades fundamentales como la capa de comunicación entre nodos o la creación de almacenamiento persistente. Esto provoca que se deba conocer cómo configurar cada plugin para que no entre en conflicto con ninguno de los servicios subyacentes.

3.1 Weave Net

Un ejemplo muy claro, y quizás el error que más tiempo nos ha costado solucionar, es el que tiene que ver con el CNI (*Container Network Interface*) Weave Net. Este error, que nos llevó casi tres semanas descubrir, impedía crear una capa de red entre contenedores, lo que hacía imposible además el despliegue de Pods en los nodos, quedándose en un estado permanente de *stand by*. Este error sucedía al utilizar una misma imagen Raspbian Lite para crear cuatro sistemas independientes.



La documentación al respecto es escasa, pero finalmente pudimos saber que el CNI Weave Net crea unas interfaces de red virtuales de manera automática utilizando el system UUID. De esta manera, al utilizar la misma imagen de Raspbian Lite para los 4 sistemas, Weave Net creaba una tarjeta de red con la misma dirección MAC para todos los nodos, lo que provocaba colisiones y que los pods no pudieran provisionarse en ninguno de los nodos.

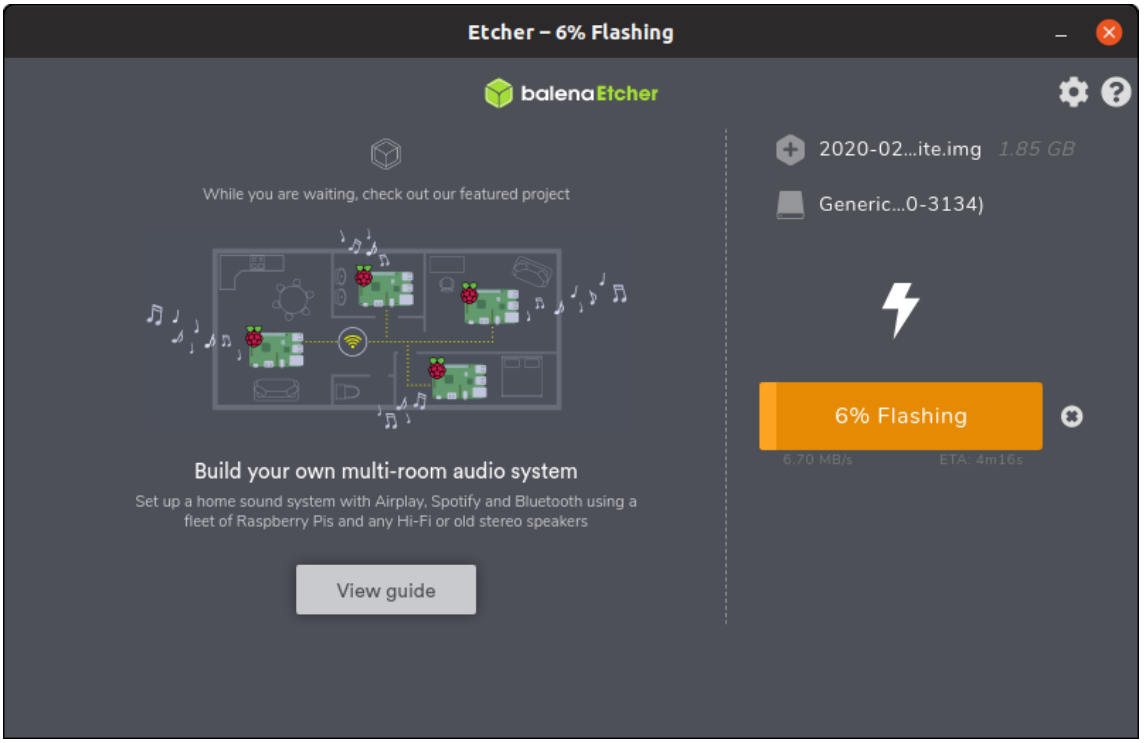
Finalmente, la solución pasaba por cambiar el UUID del sistema con los comandos:

```
sudo rm /etc/machine-id
sudo rm /var/lib/dbus/machine-id
sudo systemd-machine-id-setup
```

Tras cambiar el UUID de cada uno de los nodos, Weave Net puede crear sus interfaces de red y el sistema funciona como se espera. En un *cluster* alojado en la nube este problema es raro de ver, pero en uno creado a partir de equipos físicos que utilizan la misma imagen para crear el sistema operativo, es algo a tener muy en cuenta.

3.2 Balena Etcher

Balena Etcher es el software que utilizamos para quemar las imágenes Raspbian Lite en las tarjetas micro SD de las Raspberry Pi. En un principio, tratamos de utilizar este programa en una máquina virtual, pero a mitad de proceso ocurría un error que impedía la grabación de las imágenes en las tarjetas.



En un principio pensamos que sería problema del lector de tarjetas USB o causa de una tarjeta micro SD defectuosa. Para descartar un defecto de fabricación en la tarjeta, se utilizó otra unidad, pero el problema persistía.

Para descartar que fuera el lector USB, se utilizó un adaptador de tarjetas micro SD a tarjetas SD pero el problema seguía persistiendo.

Finalmente, en lugar de utilizar una máquina virtual Linux, se utilizó una máquina real Windows, y el problema no volvió a surgir, por lo que pensamos que debía haber algún tipo de incompatibilidad del software Balena Etcher con la máquina virtual Ubuntu 18.04 LTS utilizada en primera instancia.

3.3 Container Storage Interface

Como hemos mencionado en alguna ocasión anterior, Kubernetes es muy personalizable. Este potente software de orquestación permite elegir el software encargado de las capas de red o almacenamiento, entre otras. Además de tener problemas con la capa de red Weave Net, nos hemos visto en problemas para implementar una solución de almacenamiento persistente que permitiera a los usuarios crear una página web con WordPress.

Tras solucionar los problemas de conexión derivados de Weave Net, nos encontramos con la imposibilidad de crear almacenamiento persistente utilizando un directorio compartido NFS entre el nodo maestro y los nodos trabajadores. Utilizamos el plugin NFS-Client como provisionador de volúmenes persistentes a través de NFS, pero tras crear un Persistent Volume (PV) en el nodo maestro que pudieran utilizar los nodos trabajadores mediante un Persistent Volume Claim (PVC), éste último se quedaba en estado *Pending* permanentemente tras su creación para realizar pruebas de integración entre MySQL y Kubernetes.

Finalmente no conseguimos dar con la solución, siendo probablemente un problema de configuración el que nos privó de llegar más lejos dentro de la creación de microservicios en Kubernetes.

3.4 Montaje físico del cluster Raspberry Pi

Al inicio del proyecto, en la fase de compra de materiales, tuvimos un problema con la calidad de la estructura que daría forma al cluster de Raspberry Pi. Hacia el final del montaje nos dimos cuenta de que una de las piezas tenía un defecto de fabricación, lo que hacía imposible terminar de construir la estructura.

Tras contactar con el vendedor para tratar de solucionar el problema, éste nos envió una pieza de reemplazo en un plazo de dos semanas, pudiendo finalmente solucionar el problema de montaje.

3.5 Migración del sistema de red nativo a systemd-networkd

Por defecto, Raspbian Lite no utiliza Systemd como sistema de gestión de redes, pero era el sistema que queríamos en nuestro cluster para facilitar la gestión de las conexiones. Este sistema operativo utiliza dhcpcd como gestor de redes, pero entra en conflicto con systemd-networkd si tratamos de utilizar éste último.

Si bien no pudimos encontrar mucha información al respecto en la red, finalmente pudimos encontrar un tutorial para solucionar este problema. Finalmente, desinstalamos dhcpcd y configuramos y activamos systemd-networkd para que fuera el sistema por defecto en cuanto a la gestión de redes.

4 Conclusiones

4.1 Conclusiones generales del proyecto

Este proyecto nació del pensamiento de que la tecnología actual puede llegar a muchos más sitios de lo que lo hace actualmente. Al principio, creímos que un proyecto que pudiera integrar varias funcionalidades intercambiando tarjetas de memoria en un *cluster* sería viable con trabajo y esfuerzo. Casi al mismo tiempo que pensábamos en cómo enfocar el proyecto, descubrimos Docker y Kubernetes y soñamos con poder crear algo grande en un espacio muy pequeño.

Estas dos tecnologías, unidas a los pequeños ordenadores Raspberry Pi, nos hacían pensar que podíamos lograr algo digno de mención, algo que incluso pudiéramos usar en nuestro futuro laboral. Docker y Kubernetes son una realidad palpable en grandes empresas como Google (creadora, a la postre, de Kubernetes), Amazon o Microsoft, por lo que teníamos la ilusión de poder aprender estas tecnologías tan interesantes y apasionantes que empresas tan grandes y prestigiosas están utilizando.

Sin embargo, pronto nos dimos cuenta de que estas tecnologías eran tan complejas como apasionantes. Aún así, decidimos dedicar nuestro proyecto a ellas, a integrar muchas tecnologías nuevas que no habíamos visto para enriquecernos como estudiantes y como futuros profesionales.

Pero no todo se puede conseguir, por más esfuerzo y dedicación que se le ponga a un objetivo. La pandemia del COVID-19 llegó a mediados de marzo, justo al poco tiempo de comenzar el proyecto, y truncó nuestras aspiraciones de manera mortal para el mismo.

La pandemia ha generado un escenario difícil en el que trabajar. Con el equipo dividido, el núcleo del proyecto recayó sobre un sólo miembro del equipo, que en ese momento era el que estaba en posesión del *cluster* recién preparado para comenzar la investigación sobre Docker y Kubernetes y su posterior implantación.

Esta situación ha provocado que lo que se suponía que iba a ser un proyecto de cooperación y desarrollo se haya transformado en un proyecto casi de investigación y supervivencia en solitario. De repente, había que aprender cuatro o cinco tecnologías de manera completamente autodidacta, utilizando extensa documentación en inglés y tutoriales y vídeos que a menudo estaban desactualizados porque Docker y Kubernetes están en constante actualización. Decenas de horas se han invertido en comprender estas tecnologías y otras tantas en tratar de configurarlas y solucionar errores.

Como conclusión final, se puede decir que se ha aprendido mucho, pero los ambiciosos objetivos marcados al inicio han sido imposibles de alcanzar y han supuesto un enorme lastre en la última etapa del curso.

4.2 Consecución de los objetivos

El objetivo general no se ha podido cumplir. La situación excepcional en la que nos encontramos ha hecho muy difícil la cooperación entre los miembros del equipo, dando como resultado final la no consecución de los objetivos marcados al inicio del proyecto. Por lo tanto, nuestro *cluster* Raspberry Pi, en su estado actual, no puede cumplir con las funcionalidades derivadas de los objetivos estipulados como primordiales.

La funcionalidad principal de hosting WordPress alojado en *cluster* Kubernetes es, en este momento, muy limitada, siendo solamente posible desplegar aplicaciones que no requieran almacenamiento persistente, como servidores HTTP de prueba como Apache y Nginx, a los que se puede acceder desde la red local que soporta el sistema.

Sí se ha podido crear finalmente una página web preliminar que los usuarios utilizarían como punto de partida para la creación de subdominios en los que alojar sus contenidos web WordPress.

La funcionalidad de punto de acceso a internet no ha podido ser añadida por falta de tiempo e investigación. El tiempo dedicado a la investigación, implementación y solución de errores de Kubernetes ha sido el motivo principal de esta falta de tiempo para la creación de esta funcionalidad.

En cuanto a los objetivos específicos, la funcionalidad alcanzada es la siguiente:

- **Creación de un *cluster* de 4 x Raspberry Pi modelo 4B de 4GB de RAM funcionando de manera sincronizada como una sola unidad.** **Funcionalidad correcta**: Las cuatro tarjetas Raspberry Pi se comunican entre sí correctamente dentro de un *cluster* Kubernetes.
- **Creación de un *cluster* Kubernetes con capacidad de desplegar *pods* personalizados con contenedores Docker y unidades de almacenamiento persistente.** **Funcionalidad muy limitada**: Se pueden desplegar pods en los

diferentes nodos, pero éstos no pueden hacer uso de unidades de almacenamiento persistente.

- **Exposición pública de dichos *pods*. Funcionalidad limitada**: Es posible acceder a los pods desde fuera del *cluster*, pero sólo en red local.
- **Gestión automática de *pods* para garantizar la disponibilidad operativa. Funcionalidad limitada**: Kubernetes puede gestionar automáticamente la reposición de *pods* que dejan de estar disponibles, pero no puede vincularlos a almacenamiento persistente.
- **Creación de un sitio web que permita a los usuarios la creación de espacios de *hosting* WordPress que puedan utilizar para desplegar sus recursos. Funcionalidad limitada**: Los usuarios pueden solicitar un subdominio, pero la creación de los espacios WordPress no es automática y deben ser creados por un administrador.
- **Creación de punto de acceso a Internet para pequeños grupos utilizando tecnologías como DNSMasq. Funcionalidad ausente**: No se puede usar el *cluster* de Raspberry Pi con esta funcionalidad ya que no ha sido implementada en tarjetas micro SD adicionales.

Como reflexión, probablemente si la situación hubiera sido otra y hubiéramos podido trabajar los 2 miembros del equipo en clase, podríamos haber sacado todos los objetivos sin ningún tipo de problema.

4.3 Valoración de la metodología y planificación

Actividad	Inicio	Final
Instalación de Raspbian Lite en SD nodo Maestro	Semana 1	Semana 1
Montaje estructura física del clúster	Semana 2	Semana 2
Instalación software de clúster de Hosting	Semana 3	Semana 5
Replicación de software Hosting en nodos esclavos	Semana 6	Semana 6
Configuración de parámetros específicos de cada nodo	Semana 7	Semana 9
Realización de pruebas y ajuste de parámetros (Hosting)	Semana 10	Semana 11
Instalación de software de clúster de punto de acceso	Semana 11	Semana 12
Replicación de software punto de acceso en nodos esclavos	Semana 12	Semana 13
Realización de pruebas y ajuste de parámetros (AP)	Semana 13	Semana 14

Cuando hicimos la planificación inicial, la situación era otra mucho más favorable que la actual. Debido al confinamiento y a los problemas con los que nos hemos ido encontrando a la hora de realizar los objetivos, no se ha seguido la planificación según lo establecido. Como autocrítica, sin ninguna duda podemos apuntar que la comunicación no ha sido todo lo fluida que debería haber sido, se ha subestimado el tiempo de investigación e implementación de las tecnologías a utilizar y la carga de trabajo ha sido excesiva.

4.4 Visión de futuro

La situación surgida por la crisis del coronavirus ha provocado que aquellas funcionalidades que creíamos que estarían presentes en el producto final acaben siendo un proyecto de futuro. Sí es cierto que hemos conseguido un *cluster* Kubernetes casi funcional, pero nos ha faltado la incorporación de volúmenes persistentes, la integración con WordPress y la posibilidad de desplegar aplicaciones complejas de manera fácil. Además, debido a la falta de tiempo y las dificultades en la cooperación, muchas

características beneficiosas para la seguridad y gestión del proyecto, como Ansible, Falco o ElasticStack se han quedado fuera.

Es por ello que en el futuro queremos seguir comprendiendo el funcionamiento de Kubernetes y Docker, además de implementar todas esas tecnologías que habrían hecho de nuestro proyecto un producto redondo.

Trabajaremos para implementar Ansible como software para la gestión de los nodos en la red, Falco como sistema de detección y alerta de amenazas y ElasticStack como sistema de monitorización. Pero no nos quedaremos ahí, y ya pensamos en introducir tecnologías igual de interesantes, ya sin la presión de crear algo desde cero para que sea evaluado.

Han sido muchas las cosas que se han quedado en el tintero, pero a la vez suponen una motivación para seguir trabajando en este proyecto.

Además de explorar nuevas tecnologías de software, es posible que busquemos ampliar la capacidad de cálculo y almacenamiento del *cluster*, añadiendo nuevas placas y discos duros con los que mejorar el sistema.

Por último, nos gustaría explorar la posibilidad de llevar al público general esta idea que tuvimos hace ya algunos meses, y para ello estudiaremos la posibilidad de alquilar un dominio para exponer públicamente nuestra idea de *hosting* gratuito para entornos desfavorecidos.

5 Glosario

Kubernetes: Software open source de orquestación de contenedores creada por Google.

Docker: Tecnología open source de creación de contenedores de software.

Ansible: Software de gestión remota de sistemas Linux mediante SSH.

Cluster: Conjunto de sistemas informáticos conectados que trabajan como un único sistema.

Nodo: En Kubernetes, cada uno de los sistemas informáticos (virtuales o reales) que tienen un rol dentro del cluster.

Front-end: Es la parte del desarrollo web que se dedica a la parte frontal de un sitio web, es decir el diseño de la estructura, del sitio hasta los estilos como colores, fondos, tamaños… lo que verían nuestros usuarios.

Back-end: es el área que se dedica a la parte lógica de un sitio web, es el encargado de que todo funcione como debería, el back-end es la parte de atrás que de alguna manera no es visible para el usuario ya que no se trata de diseño, o elementos gráficos, se trata de programar las funciones que tendrá un sitio.

Contenedor: Espacio virtual donde poder aislar aplicaciones o servicios que se ejecutarán sobre un mismo núcleo del sistema.

Hosting: Servicio donde se aloja un sitio web para que los usuarios puedan acceder a él.

Dominio: Es el nombre que recibirá nuestro sitio web, podríamos decir que es el equivalente a una dirección física.

6. Bibliografía

Docker:

Orientation and setup 30/04/2020
#LearnDocker 30/04/2020
Install Docker Engine on Ubuntu 30/04/2020
Reference documentation 1/05/2020

Kubernetes:

https://github.com/mrlesmithjr/ansible-rpi-k8s-cluster#routing 03/04/2020
Everything I know about Kubernetes I learned from a cluster of Raspberry Pis 08/04/2020
Deploy WordPress in K3S Cluster (based on Kubernetes Official Documentation) 08/04/2020
How to build your own Raspberry Pi Kubernetes Cluster 08/04/2020
Creating a Raspberry Pi cluster running Kubernetes, the installation (Part 2) 29/04/2020
¿Qué es Kubernetes? 29/04/2020
Make your very own Kubernetes cluster from a Raspberry PI 29/04/2020
Building a kubernetes cluster on Raspberry Pi and low-end equipment. Part 1 29/04/2020
Kubernetes hosting 29/04/2020
Hands-on: Creating a simple web server in Kubernetes | Canviz 29/04/2020
Kubernetes get started — Deploy a simple web server 29/04/2020
Configuring your kubernetes cluster to self-host the control plane 29/04/2020
How to deploy WordPress and MySQL on Kubernetes 29/04/2020
How to run a multi-tenant WordPress platform on Google Kubernetes Engine 29/04/2020
Docker and Kubernetes 30/04/2020
Play with Kubernetes Classroom 30/04/2020
helm/helm: The Kubernetes Package Manager 30/04/2020
Kubernetes for Beginners 30/04/2020
Run Kubernetes on a Raspberry Pi with k3s 30/04/2020
K3s: Lightweight Kubernetes 30/04/2020
Corre una aplicación stateless usando un Deployment 30/04/2020
Tutorial: Run a custom LAMP application on Kubernetes — Heptio Docs 30/04/2020
Deploy your first scaleable PHP/MySQL Web application in Kubernetes 30/04/2020
Tutoriales 01/05/2020
Learn Kubernetes Basics 01/05/2020
Exposing an External IP Address to Access an Application in a Cluster 02/05/2020
Storage 02/05/2020
Comparing Kubernetes Networking Providers 03/05/2020
Getting started 03/05/2020
Instalar y Configurar kubectl 03/05/2020
Entender los Objetos de Kubernetes 04/05/2020
Run a Single-Instance Stateful Application 04/05/2020
Step by Step slow guide — Kubernetes Cluster on Raspberry Pi 4B — Part 2 04/05/2020
https://www.weave.works/docs/net/latest/kubernetes/kube-addon/#blocked-connections 05/05/2020
MetalLB, bare metal load-balancer for Kubernetes 05/05/2020
4.1 Kubernetes and iptables Rules 05/05/2020

https://kubernetes.github.io/ingress-nginx/deploy/baremetal/ 05/05/2020
[Kube 33] Set up MetalLB Load Balancing for Bare Metal Kubernetes 05/05/2020
[Kube 31] Set up Nginx Ingress in Kubernetes Bare Metal 05/05/2020
[Kube 1] Setup Kubernetes Cluster using Kubeadm on CentOS 7 05/05/2020
Maestro de Kubernetes desde cero 06/05/2020
Kubernetes, VMs & Bare Metal Pushing The Limits Of Your Private Cloud 06/05/2020
How I made a Kubernetes cluster with five Raspberry Pis 06/05/2020
How to Install Kubernetes on a Bare Metal Server 06/05/2020
Baking a Pi Router for my Raspberry Pi Kubernetes Cluster: How I used a Raspberry Pi 3 as a dhcp server and router for my Pi-based Kubernetes Cluster 06/05/2020
Unlimited Power! My Unstoppable Raspberry Pi Kubernetes Cluster: How I built a Kubernetes cluster out of a handful of Raspberry Pis 06/05/2020
Kubernetes Metal LB for On-Prem / BareMetal Cluster in 10 minutes 06/05/2020
Get a LoadBalancer for your private Kubernetes cluster 06/05/2020
Using MetalLB to implement a network load balancer on Self-Hosted or Metal Kubernetes Installation 07/05/2020
Pods encallados en estado CreatingContainer en Kubernetes con nodos creados usando Vagrant 07/05/2020
Managing Kubernetes Objects Using Imperative Commands 10/05/2020
Imperative Management of Kubernetes Objects Using Configuration Files 10/05/2020
Declarative Management of Kubernetes Objects Using Kustomize 10/05/2020
Example: Deploying WordPress and MySQL with Persistent Volumes 10/05/2020
Using Weave Net for NetworkPolicy in Kubernetes | Weaveworks 27/05/2020
Storage Classes 29/05/2020

7 Anexos

7.1 Página web

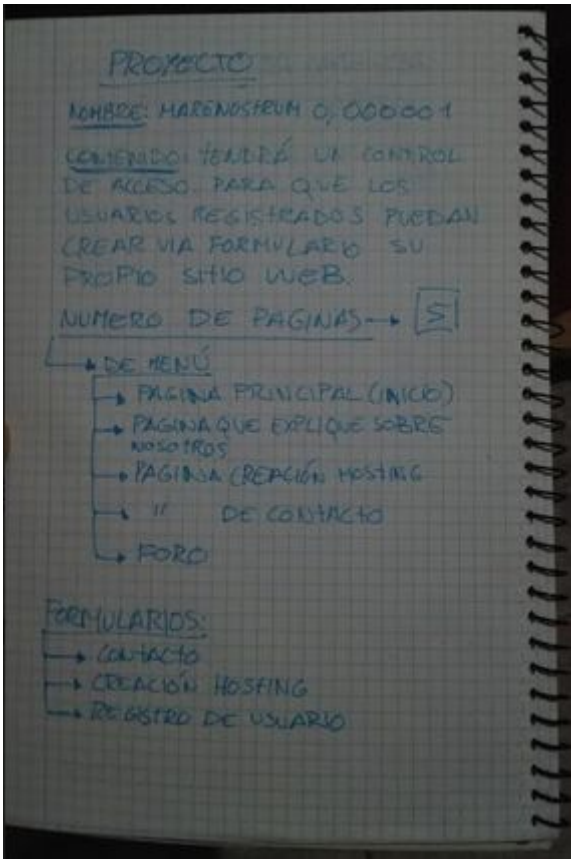
Para poder desplegar nuestro entorno web deberemos tener una serie de requisitos que nos permitirá alcanzar nuestro objetivo, para ello vamos a detallar una lista de todo lo necesario y como lo hemos instalado/configurado:
Lo primero de todo necesitaremos un Servidor web, en nuestro caso hemos optado por un servidor Apache. Este nos permitirá servir tanto sitios como aplicaciones webs accesibles vía navegador.

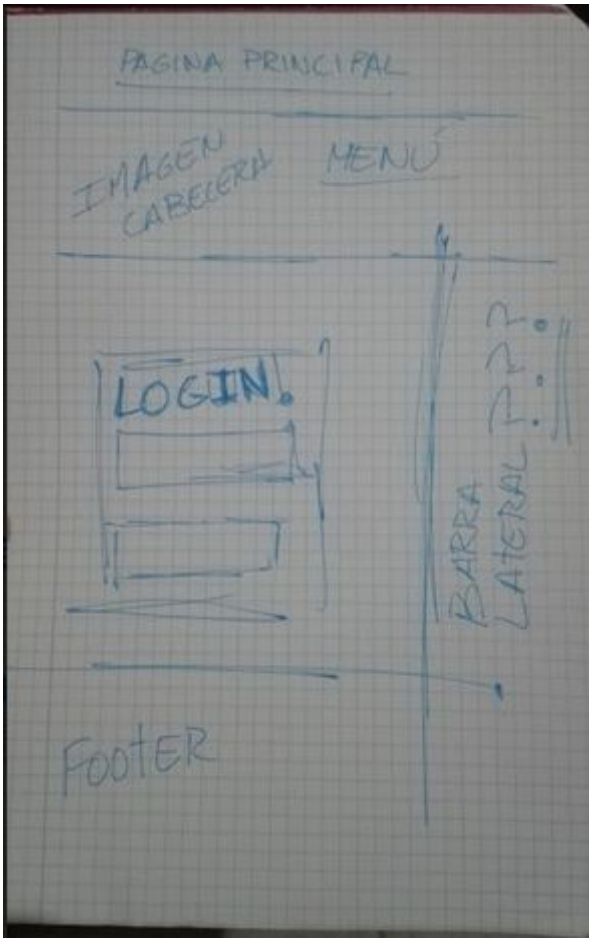
Una vez tenemos nuestro servidor web listo y funcionando es hora de pasar a instalar un Sistema Gestor de Base de Datos (SGBD), en nuestro caso hemos optado por MySQL, el cual nos hará de intermediario entre las aplicaciones web y las bases de datos del servidor.
Es decir, gestionará el acceso, consultará, añadirá y modificará la información de nuestra base de datos.

Una vez hemos instalado nuestro SGBD, pasaremos a instalar PHP, que será nuestro motor web, con el podremos generar contenido dinámico, conectarnos con nuestro SGBD para acceder a la base de datos y mostrar la información procesada sobre nuestro servidor web.

Estas instalaciones que acabamos de realizar y configurar son conocidas como la pila LAMP (Linux Apache MySQL PHP), se puede instalar de forma conjunta, aunque nosotros hemos preferido instalarlo uno por uno.
La pila lamp ya la tenemos funcionando, ahora pasaremos a instalarnos Wordpress.
Wordpress es un CMS (Sistema de Gestión de Contenido) este nos permitirá configurar nuestro sitio web sobre MySQL y PHP manejando el back-end.

Una vez instalado y configurado nuestro CMS empezaremos a planear el diseño que tendrá nuestra web sobre papel, a continuación mostramos algunos de lo bocetos:


PROYECTO
NOMBRE: MARENOSTRUM 0,000001
CONTENIDO: TENDRÁ UN CONTROL DE ACCESO PARA QUE LOS USUARIOS REGISTRADOS PUEDAN CREAR VIA FORMULARIO SU PROPIO SITIO WEB.
NUMERO DE PAGINAS → 5
DE MENÚ
PAGINA PRINCIPAL (INICIO)
PAGINA QUE EXPLIQUE SOBRE NOSOTROS
PAGINA CREACIÓN HOSTING
" DE CONTACTO
FORO
FORMULARIOS:
CONTACTO
CREACIÓN HOSTING
REGISTRO DE USUARIO

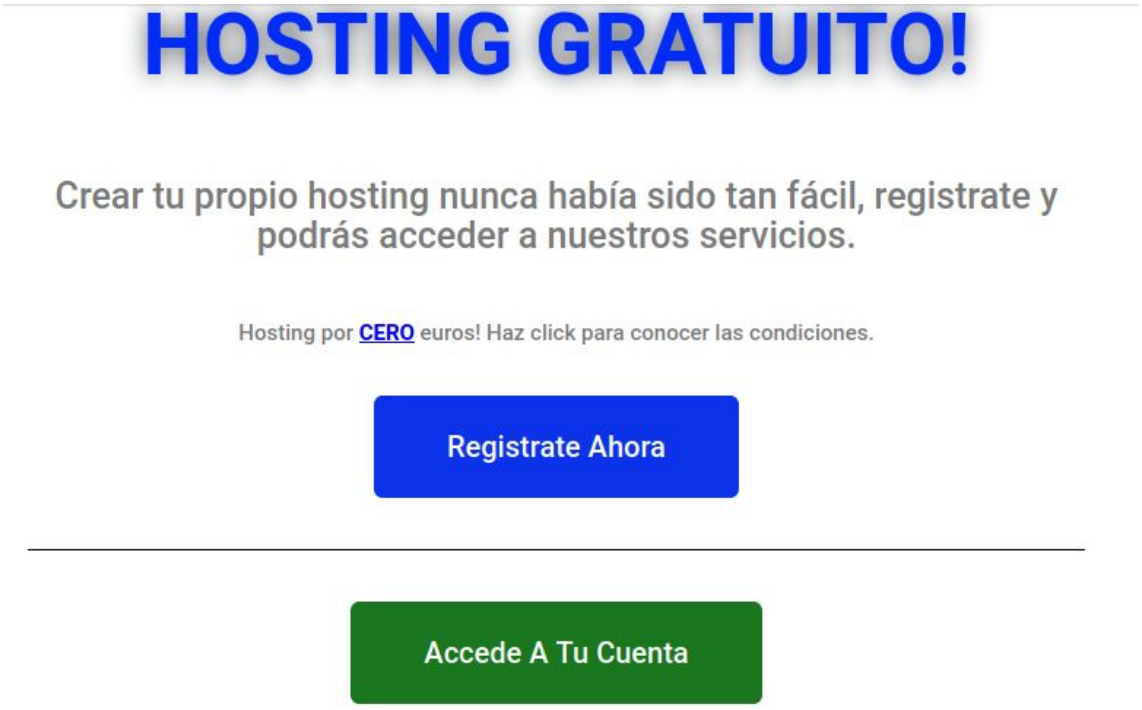

PAGINA PRINCIPAL
IMAGEN CABECERA
MENÚ
LOGIN
BARRA LATERAL
FOOTER

Una vez tengamos las ideas sobre papel bien claras es hora de plasmarlo sobre la pantalla.
Comenzaremos instalandonos el tema más adecuado para nuestra web, y empezaremos a crear las páginas y darle contenido, nosotros para nuestra web hemos hecho 5 páginas que definiremos a continuación:

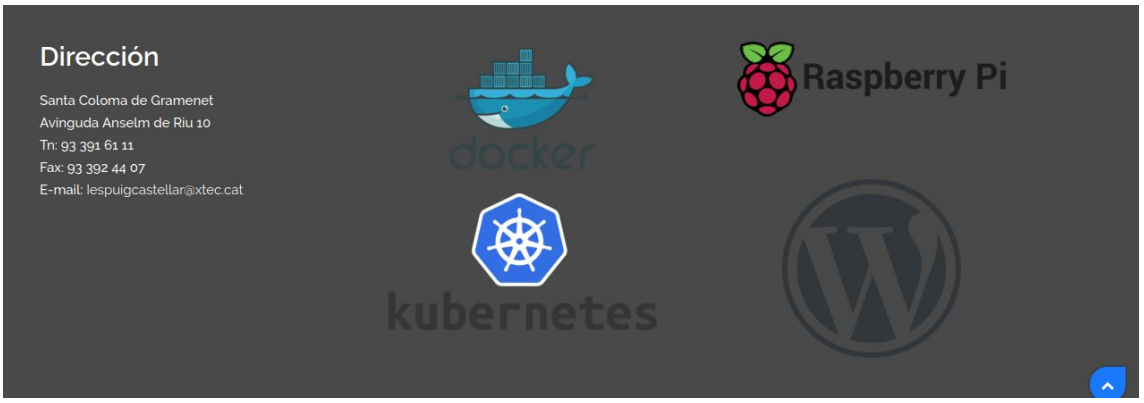
- Página de inicio: Es la página principal de la web, básicamente tiene 2 botones los cuales te permiten registrarte a nuestra web y hacer login una vez registrado.
- Página denominada ¿Quienes somos?: Pues como su nombre indica esta página explica la idea del proyecto, porque lo llevamos a cabo y sería básicamente la que nos presenta como organización.
- Página de Nuestros servicios: Esta página está restringida a todos los usuarios que no están registrados en nuestra web, una vez registrados lo que muestra es un formulario que permite crear el dominio que el usuario desee.
- Página de contacto: Básicamente es una página con un formulario en el cual los usuarios se pueden poner en contacto con los administradores del sitio web.
- Página de blogs: En esta página los administradores irán poniendo blogs relacionados con sus tecnologías, y temas de carácter parecido.

A continuación mostraremos algunos de los detalles de dichas páginas:

En esta foto podemos observar una parte del contenido de la página de inicio.



En esta foto podemos observar el footer, en el cual vemos nuestra dirección y los logos corporativos de las principales tecnologías que utilizamos, dichos logos son linkeables y nos redirigen a las páginas oficiales de las propias tecnologías.



A continuación mostramos lo que vería un usuario que no se haya registrado y por lo tanto no tendrá acceso a nuestra pagina que le proporciona nuestro servicio.

En la siguiente foto podemos ver el formulario para que un usuario se registre a nuestra web.

Otra parte importante del Back-end de nuestro sitio web es el tema de los plugin, Wordpress posé una cantidad de plugins que pueden mejorar facilitar a la hora de cambiar la apariencia de nuestro sitio web, pero también pueden potenciar y mejorar el rendimiento y cubrir labores de seguridad.

A continuación nombraremos los plugin más relevantes que se han utilizado en nuestra web:

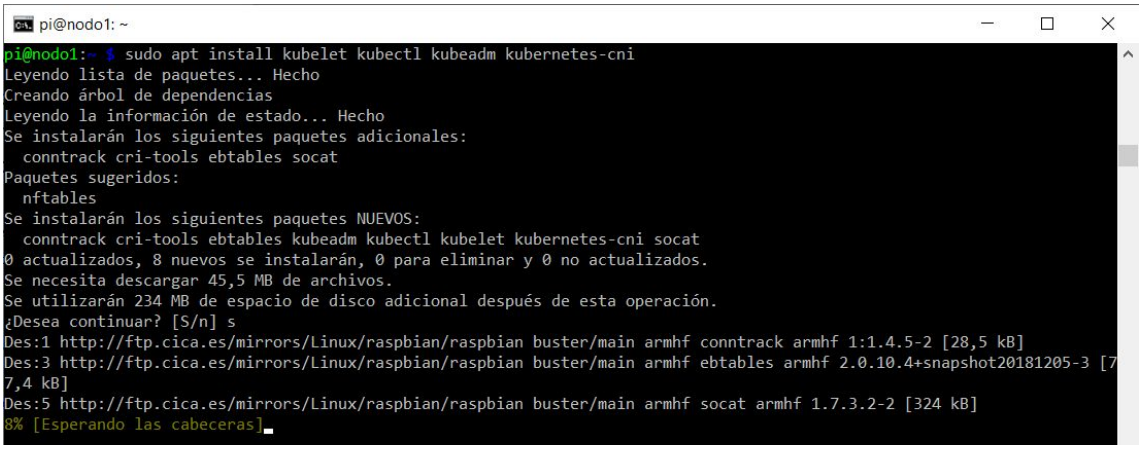
- **Elementor:**Este plugin nos permite crear páginas de forma muy rápida y sencilla con su método de arrastrar y soltar.
- **EmailLog:** Con este plugin se nos creará un log de cada formulario que envien los usuarios, de esta manera lo podrán recibir los administradores via email.
- **Titan Anti-spam & Security:** Este plugin nos aportará un extra de seguridad(Anti-spam, Anti-virus, Firewall and Malware Scan).
- **UpdraftPlus - Backup/Restore:** Este plugin nos permite realizar restauraciones y copias de seguridad que podremos guardar en local o subirlas a servicios como Drive o Dropbox
- **W3 Total Cache:** Plugin de mejora de rendimiento.
- **WordPress Simple Membership:** Plugin que nos permite crear contenido gratuito y de pago en nuestro sitio web.
- **WPForms Lite:** Este plugin nos permite crear furmularios de una forma rápida y sencilla.
- **Yoast SEO:** Plugin que mejora el SEO y permite análisis de páginas.

Cabe destacar dentro del tema de los plugins, que en un principio habíamos creado nosotros mismo ciertos plugin de manera exclusiva y para un uso concreto en este proyecto, pero la avería del Pc de uno de los miembros del equipo supuso la pérdida de dichos plugin, se trataba de un plugin equivalente a EmailLog, pero que solo permitía recibir los log en el panel de administración de wordpress.
Y otro plugin que nos permite mediante un formulario crear a un usuario con diferentes roles para y registrarlo en nuestra web.

7.2 Instalación de Kubernetes en cluster de Raspberry Pi

7.2.1 Instalación de la interfaz Kubectl

Para poder crear satisfactoriamente un *cluster* Kubernetes debemos instalar en primer lugar la interfaz Kubectl junto con la herramienta kubeadm y el agente kubelet. Es importante saber que la interfaz Kubelet sólo se debe instalar en el nodo maestro. En los nodos trabajadores instalaremos kubeadm y kubelet.

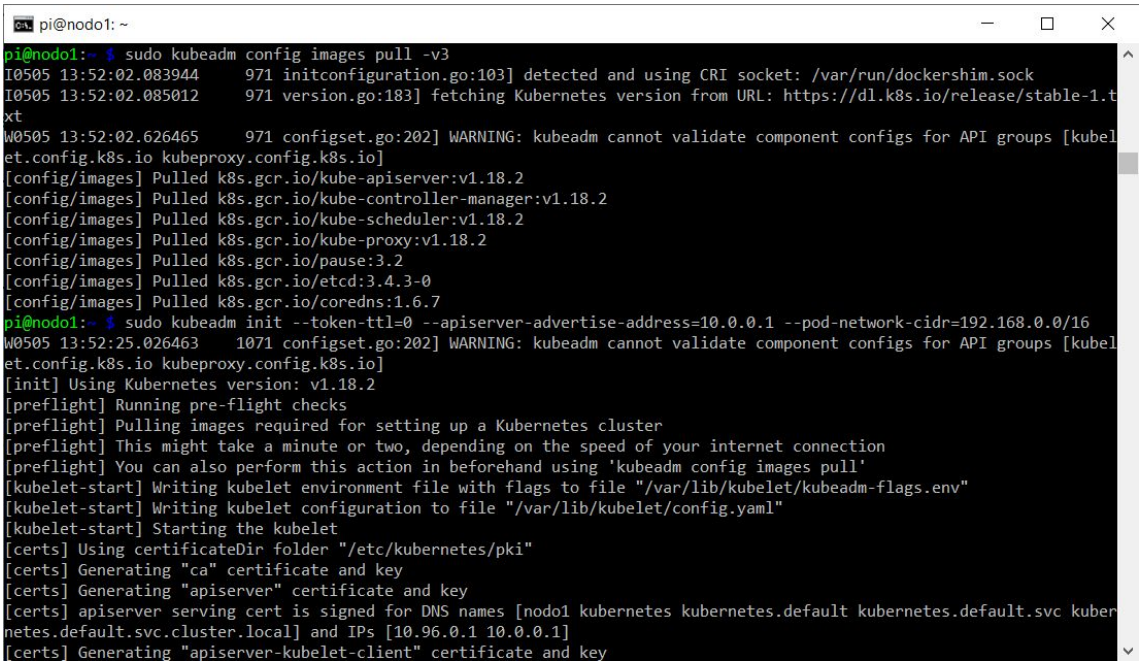


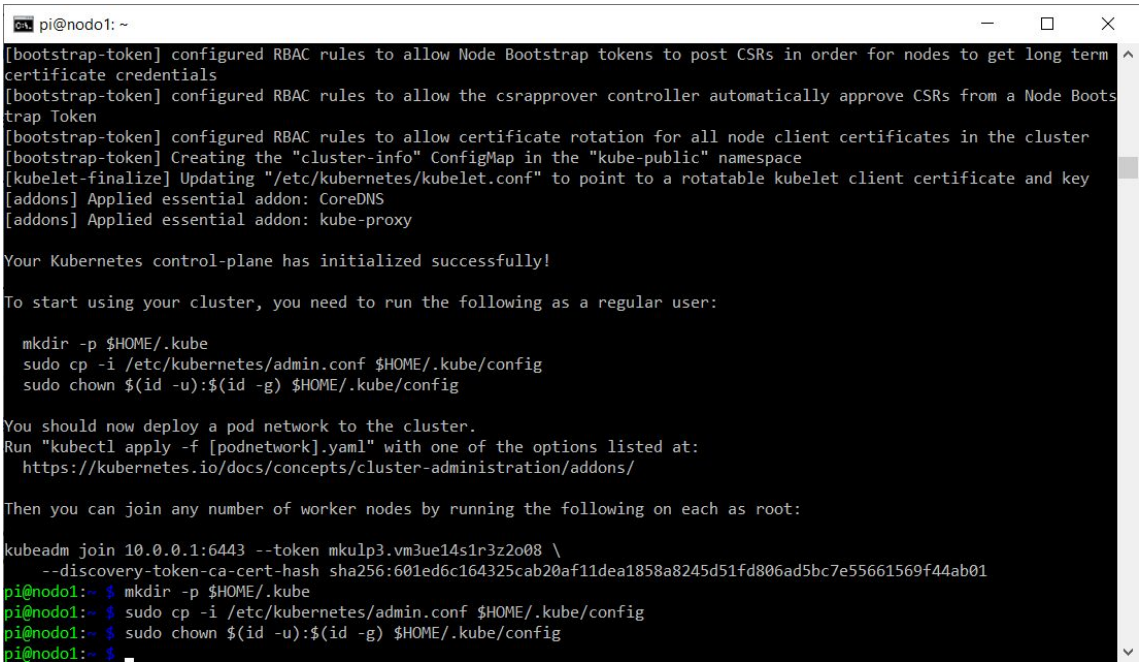
La interfaz Kubectl permite gestionar el *cluster*, la herramienta kubeadm permite poner en marcha el *cluster* y añadirle nodos de manera rápida y el agente kubelet es el proceso que permite identificar a los nodos dentro del mismo.

7.2.2 Configuración inicial

Kubeadm es la herramienta que nos permitirá crear un *cluster* de manera rápida y, además, hará posible la adición de nuevos nodos mediante el uso de un token (que deberemos guardar si queremos usarlo más adelante para añadir nuevos nodos, aunque es posible generar un nuevo código *join* en el momento).

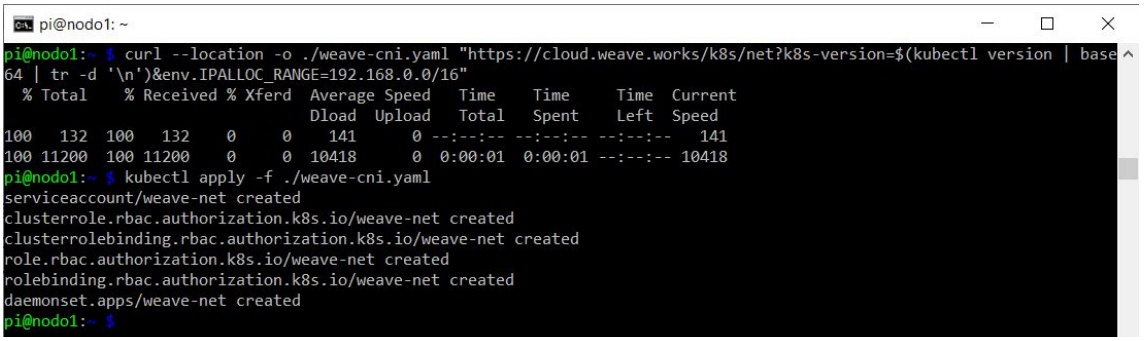
Para configurar por primera vez un *cluster*, primero debemos usar kubeadm para bajar los componentes que le darán vida. Después, con el parámetro `init` iniciaremos el *cluster* con las opciones que especifiquemos. En nuestro caso, y para propósitos de test, hemos establecido que el token que usarán los nodos trabajadores para unirse al *cluster* no caduque nunca. También hemos indicado la IP física que usará el nodo maestro para comunicarse así como la red que usarán los pods que creemos.



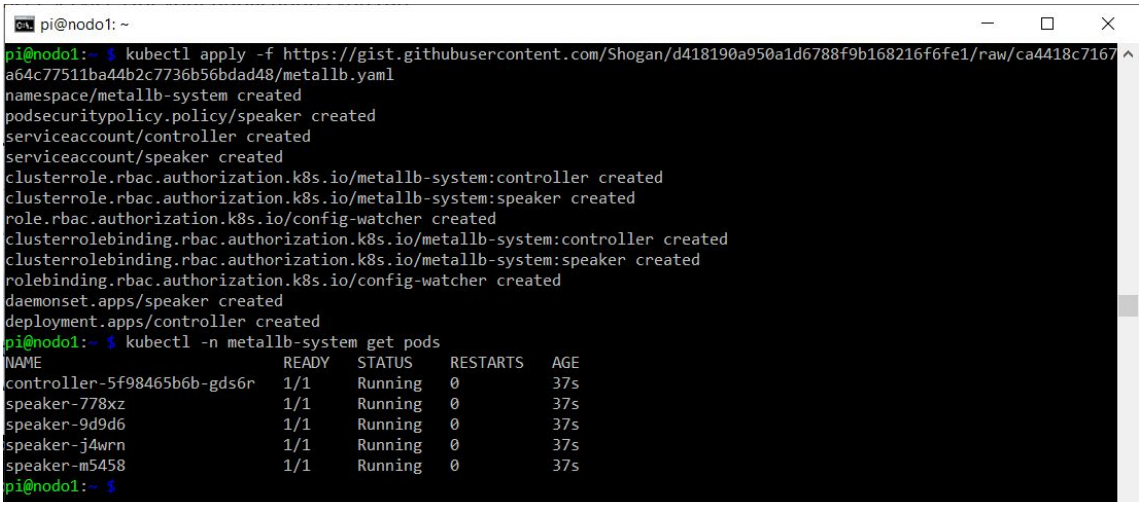


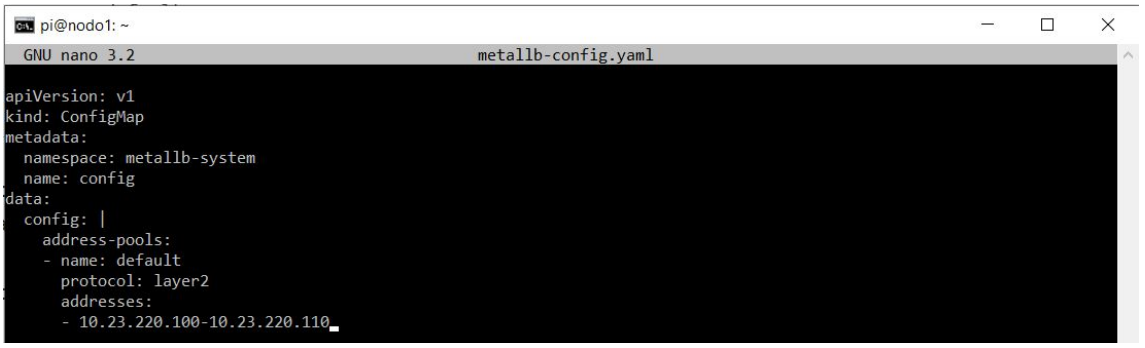
Cuando el proceso de inicialización en el nodo maestro finaliza, debemos ejecutar los comandos que se nos indica para terminar de configurar el *cluster* en su estado inicial.

Una vez este proceso ha finalizado, debemos instalar la capa de red. Como hemos mencionado anteriormente en esta memoria, la capa de red elegida finalmente ha sido Weave Net. Es muy importante asegurarse de que el UUID del sistema es único, porque de lo contrario la capa de red no funcionará por colisiones en las direcciones físicas de las tarjetas virtuales que crea este plugin.



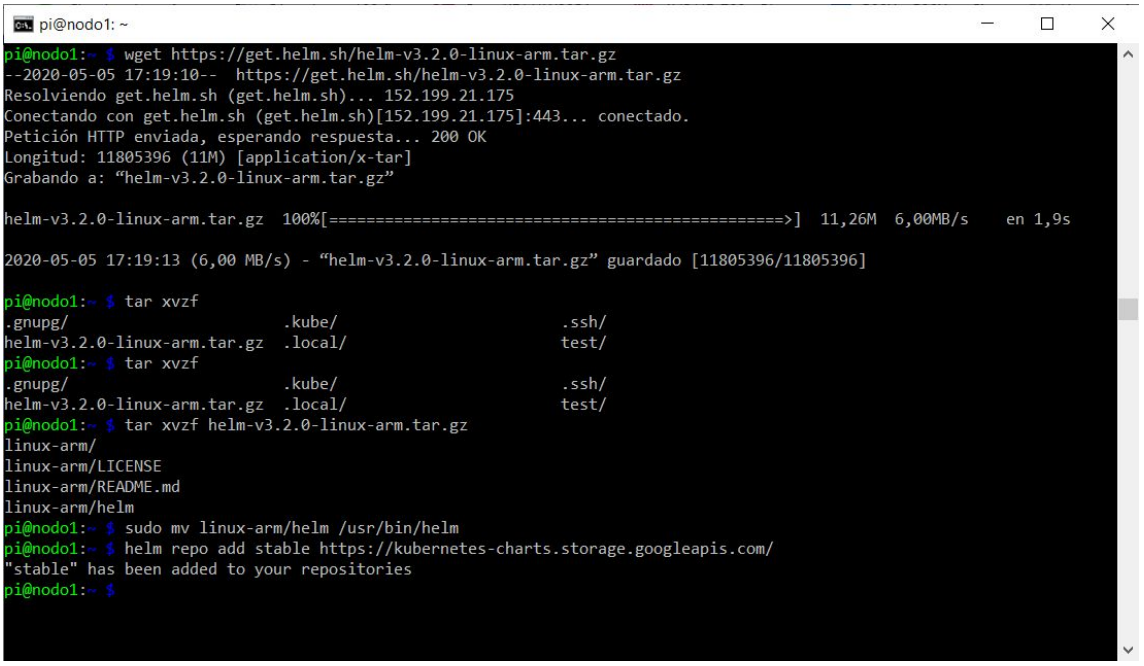
Para poder crear servicios de tipo LoadBalancer y que nuestros pods puedan exponerse al exterior con un Ingress Controller (como Nginx), debemos usar un balanceador de carga, ya que Kubernetes sobre metal no incluye uno. En servicios en la nube, esta tarea la realizan los diferentes alojamientos.



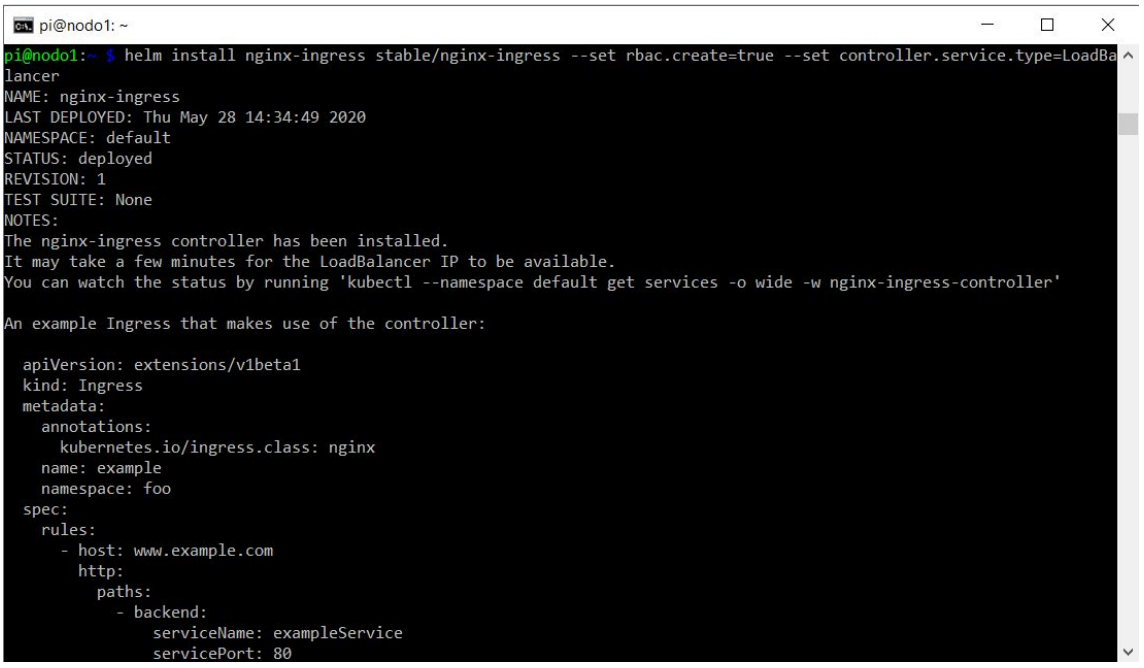


Es importante configurar MetalLB para que funcione correctamente. En nuestro caso, hemos establecido una configuración de capa 2, que es la más sencilla de configurar y cubre las necesidades de nuestro proyecto.

Siguiendo con la configuración inicial, es recomendable instalar Helm, un gestor de paquetes para Kubernetes al estilo apt o dnf pero para instalar componentes en Kubernetes.

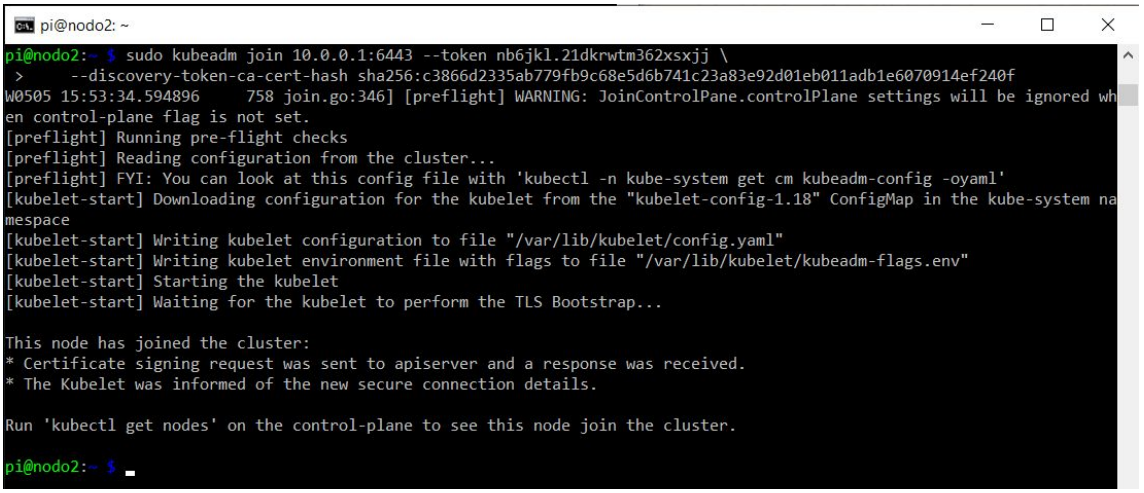


Para finalizar con la configuración inicial del nodo maestro, utilizando el recién instalado Helm instalaremos un Ingress Controller que nos permita exponer nuestras aplicaciones fuera del *cluster*. En nuestro caso, el elegido ha sido Nginx Ingress Controller.



7.2.3 Adición de nodos trabajadores al cluster

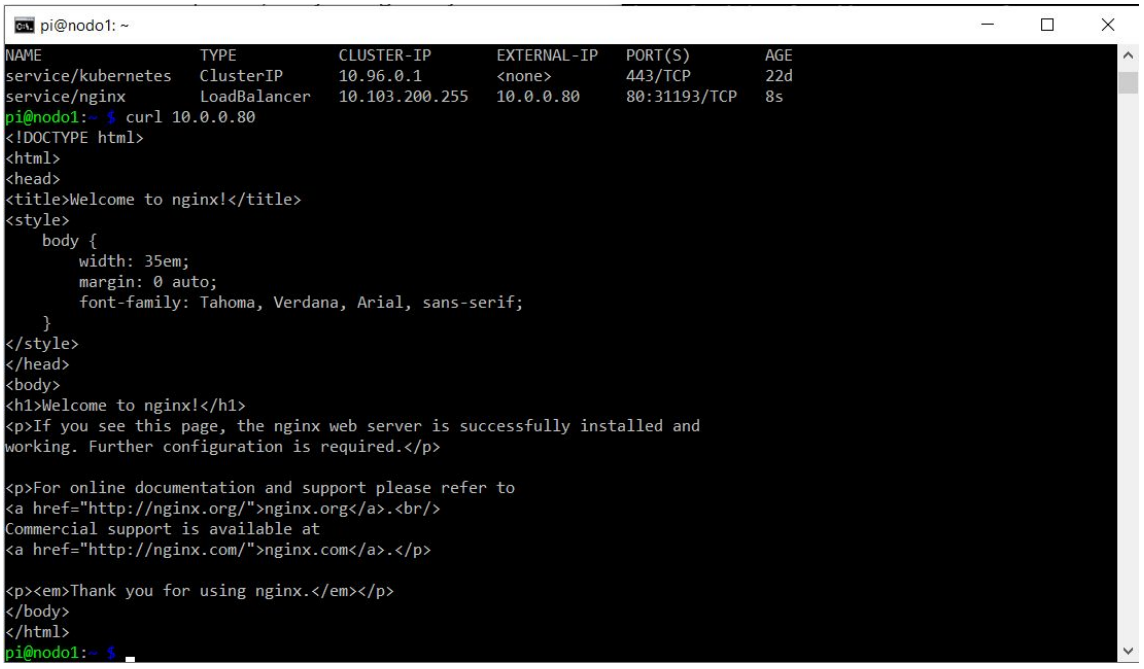
Para añadir nuevos nodos al *cluster* usaremos la herramienta kubeadm en aquellos sistemas que queramos usar como nodos trabajadores. Esta vez, en lugar de usar el parámetro `init` usaremos el parámetro `join`, indicando como parámetros adicionales la dirección IP del nodo maestro, el puerto y el token que se generó con el comando `kubeadm init` de la configuración inicial.



Cuando hayamos añadido a todos los nodos trabajadores al *cluster*, ejecutando el comando `kubectl get nodes` obtendremos una lista de los nodos que lo conforman, así como el estado en el que se encuentran, el rol que tienen asignado, así como el tiempo de vida y la versión.

7.2.3 Exponer un pod al exterior del cluster

Podemos exponer un *pod* o un *deployment* creando un servicio de tipo LoadBalancer para dichos *pods* o *deployments* con el comando `kubectl expose`. Con este comando estaremos otorgando una IP externa al objeto en cuestión, lo que hará posible que podamos acceder a él desde el exterior del *cluster*.



7.3 Bloques de Formación en Centros de Trabajo

Antonio Tomás Franco

Bloque 2

Como persona que comienza a dar sus primeros pasos en el ámbito profesional de la informática, soy consciente de que no puedo ponerme metas muy ambiciosas a corto plazo. Sin embargo, y aunque es un reto destacar en un entorno laboral cada vez más competitivo, mi objetivo a medio plazo será haber podido aprender todavía más de lo que he aprendido en estos dos últimos años.

Durante este ciclo formativo he descubierto que me gustaría aprender más sobre bases de datos y administración de servidores. Es un reto difícil, porque probablemente sean dos de los ámbitos en los que cualquier empresa pone más celo y cuidado, pero creo que con constancia y trabajo podría empezar a desarrollar una carrera en la administración de bases de datos o gestión de servidores y, quién sabe, quizá puedo ir poniéndome objetivos más difíciles y ambiciosos conforme pase el tiempo.

Tengo claro que no vale con graduarse y empezar a trabajar. Soy consciente de que la informática es un universo paralelo en constante cambio, un ámbito laboral en el que lo que hoy es tecnología punta, en menos tiempo del esperado esa misma tecnología ya no vale para nada porque alguien ha encontrado una forma mejor de hacer las cosas. No todas las tecnologías corren la misma suerte, pero la innovación, el cambio y la mejora forman

parte del ADN de esta rama laboral. Y es precisamente lo que me gustaría adoptar como filosofía personal ya en el corto plazo, cuando me incorpore a esta empresa: el cambio constante y el aprendizaje continuo para aportar soluciones que ayuden a mejorar aquellos proyectos en los que me toque participar.

En la empresa en la que empezaré a desarrollarme como profesional espero crecer, no sólo quedarme con tareas repetitivas sin ánimo de cambio. Aunque he empezado tarde a formarme como técnico de sistemas, creo que lo he hecho en un momento de madurez importante, lo cual me permitirá afrontar problemas y retos de una manera más objetiva que tiempo atrás.

En lo personal, empezar una vida laboral en el ámbito informático supone un doble reto a corto plazo, ya que este nuevo camino que comienza para mí no parte de un inicio como cuando tenía 18 años, sino que es una bifurcación que abre una nueva vía y rompe con mi pasado profesional. Tras más de 15 años trabajando en algo que poco tiene que ver con ordenadores y después de tanto tiempo sin estudiar, dejar atrás todo lo que conozco y cambiar de arriba a abajo todo aquello a lo que he dedicado casi media vida va a ser duro y difícil, pero como dije antes, y creo que me lo he demostrado a mí mismo, con trabajo y perseverancia todo es posible.
En cuanto al medio plazo, no puedo planteármelo mucho, y menos con las circunstancias tan excepcionales que nos ha tocado vivir y que han trastocado todos aquellos planes que uno se monta en la cabeza con ilusión cuando ve el nuevo comienzo tan cerca.

Sin embargo, y si la suerte acompaña, diré que para el medio plazo espero poder haber obtenido ese perfil multidisciplinar que quiero conseguir a partir del curso que viene con el CFGS de Desarrollo de Aplicaciones Multiplataforma. Si todo sale bien y consigo el título el año que viene, espero que los conocimientos adquiridos me ayuden a ser mejor profesional. Ese podría ser mi objetivo a medio plazo: mejorar mi perfil profesional con otra titulación que me ayude a adoptar nuevos roles y nuevos retos, ya sea en esta empresa o en otra nueva con algún proyecto ilusionante.

Porque soy consciente de que esta empresa por la que pase al graduarme no será mi última empresa. Por suerte o por desgracia, para avanzar hay que dejar atrás todo lo conocido. Quizás ocurre con más frecuencia de lo deseado, pero después de todo lo que he conseguido y, lo más importante, las circunstancias en las que lo he conseguido, tengo claro que es otro nuevo reto para el que estoy preparado.

Por lo tanto, si todo va bien, en el medio plazo espero haber podido especializarme y ser administrador de bases de datos, o parte de un equipo de servicios *cloud*, o programador con experiencia capaz de resolver problemas. En este momento no es algo que me pueda plantear, pero sí tengo claro que el futuro está en la especialización.

Sobre el largo plazo, tengo claro que para mejorar hay que seguir formándose. Por lo que será algo que siempre tendré presente. Podría ser con certificaciones, no es algo en lo que haya pensado exhaustivamente, pero la idea de la universidad y la ingeniería está presente. Esto me permitiría afrontar retos aún mayores y aprender tecnologías y metodologías hasta ahora desconocidas para mí. Elucubrando mucho, me permitiría liderar proyectos y asumir responsabilidades. El tiempo dirá, pero si no es algo que acabe consiguiendo seguiré estando contento sólo con lo que he conseguido hasta ahora.

Como última opción en el largo plazo, y terminando de barajar mis posibles trayectorias profesionales de futuro, cabe la posibilidad de emprender e intentar crear una empresa de servicios informáticos con el objetivo de buscar nuevas oportunidades.

Bloque 3

Este proyecto de síntesis se engloba dentro del ámbito de los servicios en la nube y el acceso a Internet. Utilizando varias tarjetas Raspberry Pi hemos tratado de crear un *cluster* que utilice Kubernetes, Docker y WordPress para implementar un *hosting* gratuito y de bajo coste para favorecer el acceso a Internet y a la creación de contenidos en partes del planeta que no lo tienen tan fácil como nosotros.

Principalmente, la funcionalidad del *cluster* se puede dividir en dos: *hosting* basado en Kubernetes y WordPress y punto de acceso a Internet. En ambos casos, se usa un *cluster* como sistema para aumentar la potencia de cálculo utilizando sistemas de bajo coste que permiten llevar tecnologías interesantes y punteras a partes del mundo que todavía están en vías de desarrollo.

Como *hosting*, el sistema funciona con 4 tarjetas Raspberry Pi que actúan como nodos Kubernetes. Uno de los nodos toma el rol de maestro o *master* y los otros tres actúan como trabajadores o *workers*. El nodo maestro reparte

contenedores y servicios entre los nodos trabajadores. Para ello utiliza una serie de servicios que trabajan juntos llevando a cabo tareas específicas.

El funcionamiento esperado por este servicio de *hosting* es el de permitir a los usuarios registrarse en la página WordPress creada al efecto. Cuando los usuarios se registran se les permite crear un subdominio WordPress en el que puedan crear y publicar sus propias páginas web utilizando todas las posibilidades de esta tecnología.

En el momento que un usuario registrado solicita un subdominio, se crearía un despliegue de WordPress + Apache + PHP + MySQL en Kubernetes con los datos recabados. El sistema de orquestación crearía los contenedores automáticamente, provisionando el subdominio solicitado por el cliente. Kubernetes en este caso asignaría a un nodo, seleccionado por su carga de trabajo y recursos disponibles, cada uno de los contenedores que contienen los elementos mencionados anteriormente. Para finalizar, y de manera resumida, Kubernetes expondría el espacio WordPress del usuario en el subdominio elegido para que el cliente pueda acceder desde fuera del *cluster*.

Otra funcionalidad deseada es la de enviar al cliente automáticamente sus credenciales para que pueda acceder a su espacio sin necesidad de que medie un administrador si todo es correcto.

Como punto de acceso a internet, las 4 Raspberry Pis proveen de acceso a internet a pequeños espacios públicos como colegios, institutos u otros edificios públicos, facilitando el acceso a internet en zonas en las que es muy caro implementar un sistema similar.

De esta manera, el *cluster* es capaz de atender las conexiones de los usuarios, así como de atender y responder sus peticiones.

Este cambio entre funcionalidades se puede llevar a cabo cambiando la tarjeta micro SD que contiene el sistema operativo de cada una de las Raspberry Pis, que previamente han sido configuradas para que la funcionalidad sea *plug & play*.

En cuanto a la aplicación en la empresa, el hecho de haber investigado cómo crear un *cluster* Kubernetes usando pequeños ordenadores cuando el servicio está pensado para grandes servicios en la nube ya establecidos como Amazon Web Services, Google Cloud o Microsoft Azure, hace que se parta con ventaja frente a otros profesionales que no han tenido la oportunidad de experimentar con la tecnología.

Esto otorga una base que puede ser beneficiosa para la carrera laboral, ya que Kubernetes y Docker no son tecnologías que se estudien normalmente en un Ciclo Formativo de Grado Superior, y menos utilizando un *cluster* real, aunque económico, como es nuestro caso.

La inclusión de la funcionalidad de punto de acceso también puede ser útil para asentar las bases de la creación y gestión de redes. Además, toda la gestión se ha llevado a cabo en sistemas Raspbian Lite sin entorno gráfico, lo que proporciona también tablas en la gestión de servicios mediante terminal.

En conclusión, este proyecto puede ser un valor añadido para mí como alumno, ya que puedo entrar a una empresa con una cierta ventaja gracias a los conocimientos adquiridos durante la investigación y desarrollo del proyecto.