Institut Puig Castellar
Santa Coloma de Gramenet

Wordpress Automatic Hosting

Administración de Sistemas Informáticos en Red

Andrés Bravo
Pratik Patel
2n ASIR
Curso 2020 - 2021

Administración de Sistemas Informáticos en Red
Andrés Bravo Ruiz
Pratik Kumar Patel

Aquesta obra està subjecta a una llicència de
Reconeixement-NoComercial-CompartirIgual 3.0 Espanya de Creative Commons

Resumen del proyecto:
La idea del proyecto es realizar un servicio hosting, donde los usuarios,con un registro e inicio de sesión previo, sean capaz de crear contenedores a través de un panel web (creado con HTML/CSS y PHP), y gracias esto, podrán tener su propia instancia Wordpress (Blog de Wordpress).

El objetivo final será tener un panel web, donde tendremos un formulario de registro, para que los usuarios se den de alta y puedan iniciar sesión. Cada usuario registrado podrá crear sus contenedores personalizados por ellos mismos, con su subdominio y su tema de wordpress, el cual podrán gestionar o destruir.

Palabras clave:
Podman, ansible, contenedores, automatización, php, hosting, wordpress

Abstract:
The idea of the project is to create a server that is capable of containing containers (pods) in which certain wordpress are installed, which in turn can be managed and configured through a web environment from which they can also be created and destroyed.
The objective is to simplify the action of creating containers that contain wordpress, since it is often a difficult task. Therefore, we try to facilitate the task for those users who want to have their own website (wordpress) and do not have the prior knowledge to do so.
Once users create their own container with wordpress, they will be able to perform the action of modifying the template, downloading the plugins, managing users, etc., all this the user will be managing in their own domain with a public IP.
The organization is done through a Gantt chart and the Asana service, which allows us to apply the SCRUM methodology.
The final goal would be to achieve a working server prototype that works locally to see the efficiency of this type of container hosting systems with wordpress and consider if the idea can be transferred to a public and online environment.

Keywords:
Podman, ansible, containers, automation, php, hosting, wordpress

Índice

Introducción

En la actualidad resulta muy difícil imaginar que administradores de una empresa, emprendedores o usuarios independientes compren servidores y los configuren para poder alojar su sitio web. En estos tiempos disponemos de servicios hosting que ofrecen sus servidores para que el usuario aloje su web a cambio de un Plan de Hosting Web (Gratuito, de Pago..etc).

Hay servicios como AWS (Amazon Web Service) que engloban una gran cantidad de servicios para poder realizar distintos tipos de actividades en la nube, uno de ellos es la creación de instancias (contenedores) que pueden ser utilizados como alojamiento de sitios web, base de datos, aplicaciones, almacenamiento de contenido, servicio cloud, entre otros.

Contexto y justificación del proyecto

W.A. Hosting (Wordpress Automatic Hosting) consiste en facilitar la creación de hosting (servicio de alojamiento para sitios web) y automatizar este servicio. Los usuarios que se den de alta, mediante un panel web, podrán crear su hosting gratuitamente solo rellenando un formulario con el nombre y las características que deseen en ese contenedor (hosting), además de eso dispondrán de un wordpress con un tema escogido por el usuario previamente.

Objetivos del trabajo

La idea del proyecto es realizar un servidor que sea capaz de crear contenedores, los cuales serán creados por los usuarios/clientes a través de un panel de control web, donde podrán crear su propia instancia Wordpress (Blog de Wordpress).

A la hora de cumplir con éxito el trabajo queremos conseguir los siguientes apartados :

1. Aprender como funciona Podman (Contenedor).
2. Aprender a utilizar la plataforma de nube Amazon Web Services.
3. Hacer que funcionen los servicios de los contenedores.
4. Conseguir que funcionen los contenedores de cada usuario sin errores.
5. Conseguir optimizar todo el proceso con nuestro formulario.
6. Descubrir el mundo del contenedor + hostings.
7. Ofrecer soporte al cliente.

Estrategia y planificación del proyecto

Tendremos en cuenta otros tipos de servicios hosting o de creación de servidores (contenedores), para poder desarrollar de una manera más eficiente las metas del proyecto.

Metodología de trabajo

La técnica para la planificación, realización y seguimiento del proyecto será "**SCRUM**".
Nos reuniremos dos veces en semana. Uno de los días será utilizado para hacer un seguimiento de las tareas que hemos llevado a cabo durante la semana. El otro día restante será para hacer un análisis sobre el desarrollo del proyecto, aprovechando el mismo para planificar las tareas pendientes de la semana venidera, hasta el fin del proyecto.

Utilizaremos un Diagrama de Gantt para hacer un seguimiento de todas las tareas que componen el proyecto y sus diferentes fases.

N	Tareas	Duración	Inicio	Final
1	**Desarrollo de un servidor hosting con Podman**	100H	01.04.21	30.05.21
2	**Abril**	42H	01.04.21	30.04.21
3	Analización + Configuración de Podman	12H	02.04.21	10.04.21
4	Crear un servidor LAMP + DNS en contenedor	6H	11.04.21	15.04.21
5	Configuración de dominios e sudominios	12H	16.04.21	22.04.21
6	Un panel web con PHP para los usuarios	12H	23.04.21	30.04.21
7	**Mayo**	58H	1.05.21	29.05.21
8	Decoración de panel PHP (CSS)	6H	03.05.21	09.05.21
9	Un playbook Ansible (Podman + Wordpress)	30H	10.05.21	29.05.21
10	Estudio de datos y conclusion (Memòria Principal)	22H	01.04.21	29.05.21

Enlace al Diagrama de Gantt

Estudio económico y presupuestario

Para la realización del proyecto, se ha tenido en cuenta los costes materiales, es decir equipo informático, software utilizado, coste de los recursos humanos y otros elementos:

Para llevar a cabo el proyecto se han considerado los siguientes elementos:

- Recursos Humanos: Salarios de cada uno de los roles participantes en el proyecto:
  - Andres Bravo: Administrador de sistema (1.300 € / 99H)
  - Pratik Kumar: Administrador de sistema (1.300 € / 99H)

- Material: Hardware y Software
  - 1 Instancia "Máquina Virtual" en AWS (50€)
  - Licencia de Software (Gratuito)
  - Renta de los servidores (500€ / mes)

- Presupuesto inicial total: 3150 €

Descripción del proyecto

Análisis de requisitos

- Profundizar y tener más conocimientos de los contenedores.
- Cómo funciona la conexión entre una página web y el servidor.
- Diseñar una página web estructurada correctamente.
- Tener éxito en la creación de contenedores desde la página web (Automatización).

Requisitos funcionales

- Dominio y subdominios de la página web.
- Servidor de contenedores wordpress.
- Conectar la página web con el servidor.

Requisitos no funcionales

- Página web
  - funcional
  - intuitiva
  - responsive

Tecnologías

A la hora de empezar el proyecto nos hemos tomado el tiempo necesario para poder investigar sobre las tecnologías y las métricas que utilizaremos durante el transcurso del mismo.

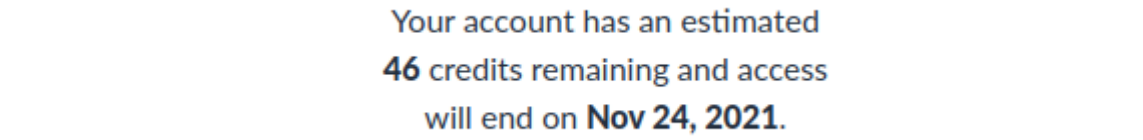
Amazon web service

Es una plataforma de servicios de nube que proporciona una variedad de servicios de infraestructura tales como almacenamiento, redes, bases de datos, servicios de aplicaciones, potencia de cómputo, mensajería, inteligencia artificial, servicios móviles, seguridad, identidad y conformidad....

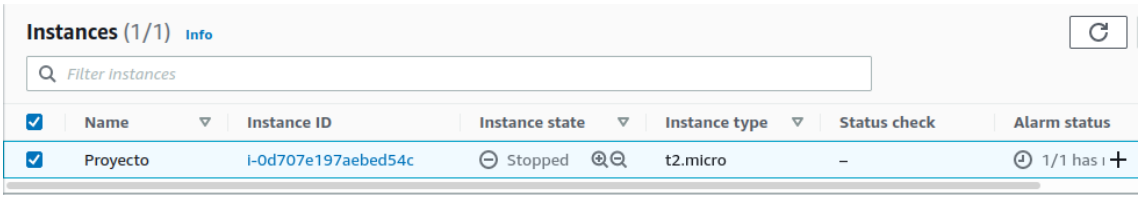
Nosotros en este proyecto estamos utilizando una plataforma que está dedicada para los estudiantes de 14 años o más.

AWS Educate es una iniciativa global de Amazon cuyo objetivo es proveer a los estudiantes recursos integrales para desarrollar habilidades vinculadas con la nube.

Nosotros tenemos creada una instancia (Virtual Host) en aws educate, en principio nos ofrecen bastantes créditos gratuitos. Juan (Profesor de redes), nos ofreció algunos créditos aparte para realizar este proyecto. Como podemos ver en la siguiente captura, ahora mismo disponemos de 46 créditos (dólares).



La instancia (Máquina virtual) creada del proyecto:



Para ver más información sobre AWS
Consulte Anexos 1.1

Podman
Es un motor de contenedores, que nos permite actualizar estos contenedores de una manera similar a Docker, pero con algunas diferencias fundamentales:

Rootless: Nos permite levantar contenedores sin necesidad de ser root (principal ventaja). Importante ya que se crearán los contenedores a través de un formulario web, el cual se visualizará gracias a nuestro servidor web (apache)

Daemon-less: Podman no necesita levantar un único demonio de muchos servicios para funcionar, es más bien algo parecido a la arquitectura de microservicios, levanta los servicios necesarios para cada contenedor de forma que se hace mucho más "tratable" la gestión de servicios en el anfitrión.

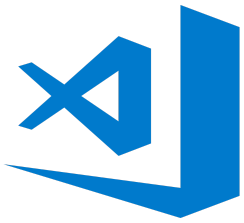
Pods: Podman acuña el término pod tal y como lo conocemos con Kubernetes de forma que podríamos levantar pods de uno o varios contenedores y aislarlos de otros pods.

Línea de comandos: Su sintaxis es similar a la de docker de forma que la curva de aprendizaje es igual a esta.

Ventajas de utilizar Podman

- Uno de los principales beneficios de usar Podman es su seguridad mejorada.
- Para los que comienzan, Podman se puede ejecutar sin privilegios de administrador, manteniendo casi toda su funcionalidad.
- Podman también utiliza un modelo de bifurcación/ejecución, en vez del modelo cliente/servidor que Docker utiliza.
- Este modelo permite el paso de sockets conectados desde systemd y la devolución de los avisos a través de la pila en la que Podman está listo para recibir tareas.

Para ver instalación y configuración que hemos estado realizando en el proyecto,
Consulte Anexos 1.2



Visual Studio Code

Es un gran editor para el desarrollo de PHP, dispone de características como el resaltado de sintaxis, concordancia de llaves en expresiones y snippets que pueden añadir funcionalidades a tu día a día. Sin embargo, esto no es suficiente para competir con otros sistemas IDE como PHPStorm o Eclipse entre otros. Así que además de estas características, podemos ampliar funcionalidades con las extensiones que encontraremos en la tienda de Visual Studio Code.

Draw.io

es la herramienta google que te permite dibujar todo tipo de diagramas, ya sean diagramas UML o un dibujo cualquiera. Dentro del proyecto esta herramienta la utilizaremos para hacer el mock -ups de la aplicación.

Contenido

Tras un análisis del objetivo del proyecto, decidimos tener varios puntos en cuenta, uno de ellos era el público objetivo para el que iba a estar orientado esta herramienta, usuarios que quieren tener un wordpress y que no tienen la capacidad de realizar o crear un "pod" que está hecho con distintas imágenes que nos ofrece Podman.

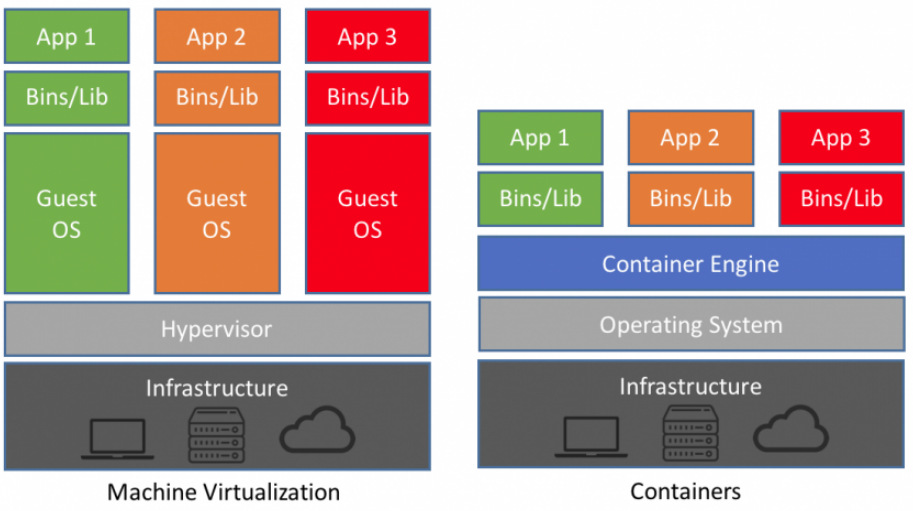
La solución que se llevó a cabo fue la virtualización de contenedores virtuales mediante la tecnología Linux Ubuntu Container, en este caso la máquina virtual la teníamos en la nube AWS Educate y en ella se encontrarán todos los contenedores e imágenes necesarias para que el cliente pueda tener un Wordpress solo rellenando el formulario que hemos implementado mediante php.

Además, si desea implementar el sistema en un servidor de mayor capacidad, la virtualización también puede lograr la portabilidad o migración.

Entorno de virtualización

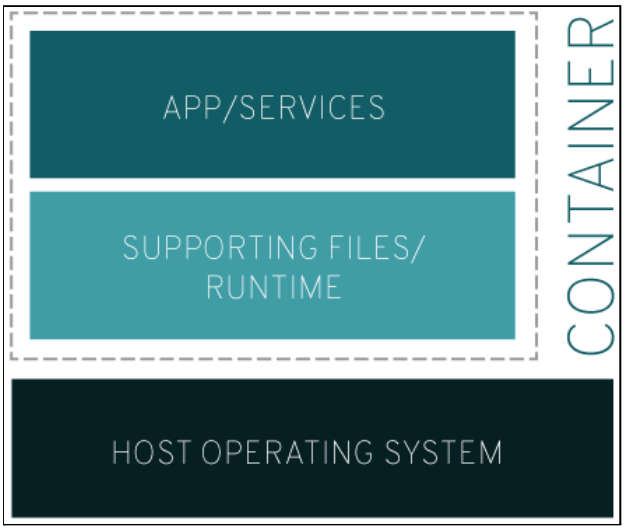
Para este proyecto lo que hemos utilizado para su implementación, ha sido la nube AWS Educate, nosotros tenemos una máquina contratada (alquilada) en la nube de AWS Educate y eso nos ayuda mucho a la hora de trabajar, ya que solo tenemos que disponer del par de claves de acceso de la máquina AWS. Eso significa que desde casa, trabajo o aula de clase tenemos acceso por terminal al servidor contratado.

El entorno de virtualización que hemos utilizado es Podman. Este método de virtualización nos permite ejecutar diferentes servicios o aplicaciones dentro de diferentes entornos virtuales denominados "contenedores", estos están aislados entre ellos en una sola máquina que los controla.

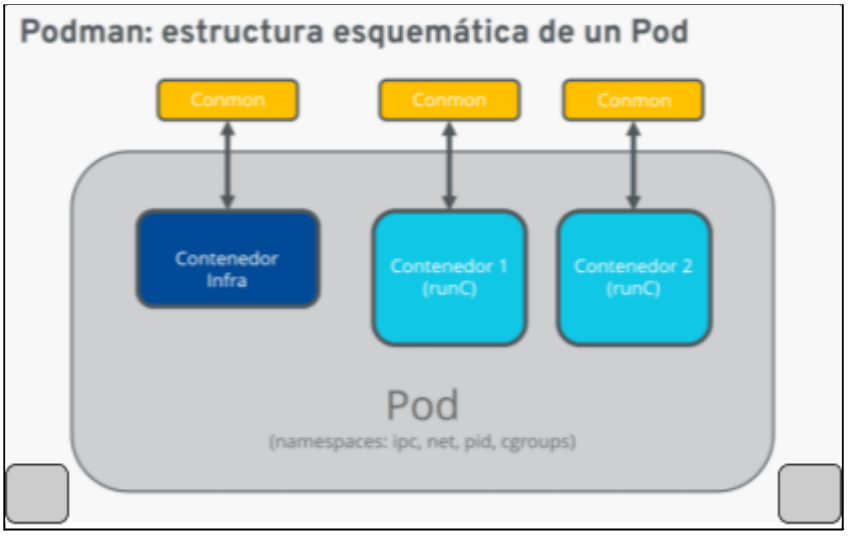


Podman

Sería mucha tarea para nosotros como administradores, tener que crear máquinas para los diferentes servicios a utilizar. Por cada usuario que se dé de alta y necesite un servidor web (con wordpress y una BD dedicada) tendríamos que instalar y configurar los servicios necesarios para el usuario correspondiente, esto podríamos hacerlo directamente en la máquina servidor, pero no sería óptimo, ya que sería un descontrol, y lo ideal sería tener estos procesos/servicios separados del resto del sistema (que sean independientes)



Podman nos será muy útil, ya que nosotros no tenemos que crear distintas máquinas para diferentes usos. Nos ofrece contenedores particulares de los servicios necesarios (Apache, Nginx, Mysql, etc...), Para ello debemos bajar las imágenes oficiales de Podman y asignarlas a nuestro contenedor.



Esto provoca que el sistema sea capaz de virtualizar un volumen mucho mayor de contenedores que el volumen que sería capaz de alcanzar mediante máquinas reales/virtuales, por los recursos que necesitan.

Otro punto importante es que todos los contenedores se ejecutan sobre el mismo kernel del sistema, lo que quiere decir que se puede iniciar, reiniciar, parar y apagar los

diferentes contenedores, esto se hace a través de un proceso muy veloz y sencillo, sin las complicaciones que podría llevar una máquina real o virtual.

La gestión de los contenedores Podman se realiza mediante comandos, esto permite la utilización de scripts, para en nuestro caso poder automatizar la creación de contenedores a través de la web.

Algunas ventajas de Podman a la hora de desarrollar:

- La gestión de los contenedores mediante comandos facilita mucho su uso.
- Se pueden realizar pruebas de los contenedores mediante la clonación o restablecimiento de los mismos.
- El funcionamiento de la propia herramienta es mediante el sistema de archivos de Linux, por lo que es muy sencillo acceder a los archivos.
- Es sencillo realizar copias de seguridad y restaurarlas.
- Da la posibilidad de ejecutar diferentes versiones de aplicaciones, experimentar o probar nuevo software antes de actualizar de manera definitiva un servidor.

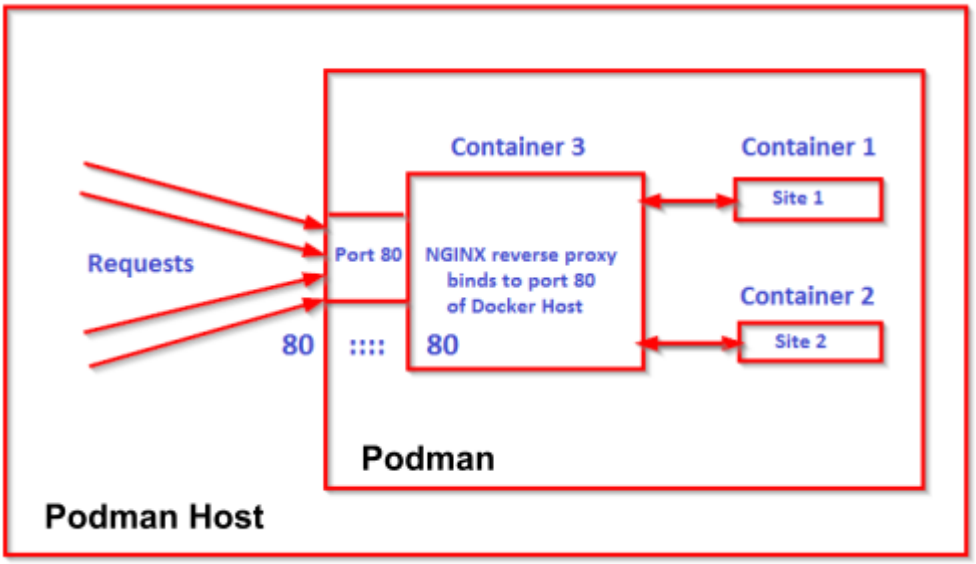
Una de las desventajas que nos encontramos al realizar este proyecto, es que no hay mucha información en Internet sobre este servicio, ya que es una nueva tecnología. Nos encontramos con muchos errores, como puede ser el tema de permisos, problemas al ejecutar comandos de podman con un usuario de servicio, como puede ser el de apache, y no hay mucha información en internet para guiarnos.

Nginx “Proxy Inverso”

Un servidor proxy inverso generalmente se coloca detrás del firewall de una red privada y recupera recursos en nombre de un cliente de uno o más servidores, equilibrio de carga, enrutamiento de solicitudes, almacenamiento en caché, compresión, etc.

Configuraremos un proxy inverso con NGINX delante de un par de servidores web que contienen Wordpress (en este proyecto los servidores webs son los Pods “contenedores”).

Nuestro proxy funcionará de la siguiente forma :



Dentro del contenedor, los puertos y las direcciones IP son privados y no se puede acceder a ellos externamente, a menos que estén vinculados por host. Por lo tanto, solo un contenedor puede vincularse al puerto 80 del host de la ventana acoplable.

¿cómo podemos acceder a varias aplicaciones web que se ejecutan en varios contenedores a través del puerto 80 del host de la ventana acoplable?

La respuesta es a través del proxy inverso y usaremos nginx dentro de un contenedor que vincula su puerto 80 al puerto 80 del host de la ventana acoplable y reenviará la solicitud a la aplicación web que se ejecuta en varios contenedores.

Automatización de tareas

Uno de los puntos más importantes de un proyecto así y en general de un buen administrador de sistemas es el de la automatización de tareas. No tendría sentido que cuando un usuario necesite crear, iniciar o apagar un servidor se tenga que realizar manualmente con una persona pendiente las 24 horas para la realización de las diferentes tareas según el caso.

Es por esto que se ha realizado una automatización de las diferentes tareas mediante archivos Bash script, php y con la herramienta ansible se ha logrado automatizar los diferentes procesos para no necesitar la mano humana a la hora de realizar una de las posibles tareas que se le mande mediante la página web “formulario”.

Integración entre Ansible y Podman

Es un software de gestión de la configuración automática y remota, que nos permite centralizar la configuración de numerosos servidores, dispositivos de red y Cloud Providers de una forma sencilla y automatizada.

Podemos usar Podman y Ansible de forma separada pero también podemos combinarlas para beneficiarnos de las propiedades de cada una. En Ansible disponemos de muchos módulos que nos permite manejar las imágenes y los contenedores de Podman, por ejemplo:

- **podman_container** : Administrar contenedores de podman.
- **podman_image :** Descargar imágenes para que las use podman**.**
- **podman_login :** Inicie sesión en un registro de contenedores utilizando podman.
- **podman_logout :** Cerrar sesión en un registro de contenedor usando podman.
- **podman_pod :** Administrar pods de Podman.
- **podman_volume :** Administrar volúmenes de Podman.
- **podman_volume_info** : Recopilar información sobre los volúmenes de podman.

Usando tareas de ansible podemos automatizar el arranque del servicio de Podman, la construcción de las imágenes con los ContainerFile y el inicio o parada de los contenedores de Podman.

Principales características de Ansible

A Ansible se le agrupa como una herramienta de gestión de configuración, pero lo cierto es que no se le puede encasillar solamente en eso, pues con todos los módulos de los que dispone puede ser utilizado en otro tipo de escenarios.

Aprovisionamiento
Con Ansible se pueden aprovisionar hosts virtualizados por lo que se pueden realizar descargas de nuevo contenido.

Gestión de la configuración
Sus distintos módulos permiten modificar archivos de configuración de todo tipo, permitiendo una gestión automática de la mayoría de aplicaciones o servicios de un sistema.

Despliegue de aplicaciones
Se puede definir una aplicación con Ansible que permite llevar un control de todo el ciclo de vida de una aplicación.

Seguridad y Cumplimiento
Ansible permite definir la seguridad de los sistemas de forma sencilla. Utilizando los diferentes Playbook se pueden definir reglas de firewall, gestión de usuarios y grupos y políticas de seguridad personalizada en los sistemas que se estén gestionando poseyendo un gran número de módulos que ayudan en su labor.

Playbook Ansible

Ansible se puede utilizar de varias formas, ejecutando directamente el comando con las cosas que quieres realizar o mediante playbooks, esta segunda opción beneficia mucho la automatización, ya que la función de los playbooks es la de un archivo de extensión **.yml** en el que se podrá introducir de una forma muy cómoda las diferentes cosas que queremos que se realicen, es decir que se pueden realizar varias tareas una detrás de otra.

La creación de los contenedores se hará a partir de un fichero **.yml** genérico, los datos de este fichero irán cambiando conforme los datos del usuario al enviar el formulario.

Bash Script

Bash es un lenguaje de comandos y Shell de Unix lanzado por primera vez en 1989, que se ha utilizado ampliamente como el Shell de inicio de sesión predeterminada para la mayoría de distribuciones de Linux, es decir que Ubuntu server utiliza Bash para la interacción del administrador con el sistema. Una de las características de Bash es la opción de realizar archivos denominados scripts, estos archivos mediante el texto que se introduzca permite la programación de tareas, como ejecutar comandos para realizar alguna tarea en concreto.

Dado que es el lenguaje por defecto que utiliza Linux, es una tecnología muy potente a la hora de realizar ciertas tareas, por ello, se procedió a la utilización de este lenguaje que aún y no siendo tan potente como podrían haber sido otros lenguajes, es el nativo y era lo suficientemente útil para cubrir las necesidades que se deseaba.

PHP

Php es un lenguaje de código abierto, que nos permite generar páginas web dinámicas (páginas cuyo contenido no es el mismo siempre). En este caso, el contenido de nuestra página deberá ir cambiando en base a los cambios que haya en nuestra base de datos.

El lenguaje PHP enviará peticiones al servidor, el cual recibe todas las peticiones, reúne la información necesaria consultando en la base de datos, y responde enviando una página web “normal” (estática) pero cuya creación ha sido dinámico (realizando procesos de modo que la página web devuelta no siempre es igual).

Desarrollo

Debemos realizar una herramienta a nivel local que permita crear contenedores que contengan un Wordpress limpio con una plantilla por defecto y apartir de ahi ya cada cliente pueda modificar el wordpress a sus gustos, todo esto gestionado por una página web diseñada también por el grupo desde cero programando con PHP aprovechando la experiencia adquirida en antiguos estudios por los dos participantes.

Análisis de necesidades

El primer paso antes de empezar a desarrollar el proyecto ha sido un análisis de las necesidades básicas del proyecto, se ha buscado información de las diferentes tecnologías que se podrían utilizar, priorizando en todo momento tecnologías/herramientas con conocimiento ya adquirido, con esto ahorraremos tiempo de aprendizaje. Fue entonces cuando acabamos seleccionando las diferentes herramientas:

- ➔ **AWS Educate** para la creación del entorno servidor
- ➔ **Podman** para la creación de contenedores
- ➔ **Script Bash** y **Ansible** para la gestión de los contenedores, modificar los ficheros, etc...

por otra parte utilizamos **Visual studio** para la creación y edición de la página web tanto los .HTML, .CSS, y .PHP

Los servicios que utilizaremos para los contenedores:

1. Apache y PHP para la visualización de la página y gestionar los usuarios.
2. Mysql para la creación de una bases de datos “wordpress”.
3. Nginx como proxy inverso

Gestion de tareas

Desde un primer momento se ha dividido el proyecto en dos partes, por una parte el desarrollo de la página web, la creación de la misma, la obtención de recursos gráficos y la estructuración de formularios para poder configurar los archivos de configuración de los contenedores.

Y por otro lado la configuración de distintos servicios creando contenedores y comprobando el comportamiento del mismo ante los servidores que consumen sus recursos para funcionar. Para ello se han realizado pruebas con varios contenedores para comprobar su funcionamiento.

Servidor

Como hemos explicado anteriormente, hemos apostado por Podman como tecnología a utilizar en este proyecto. Primero de todo, hemos tenido que estudiar su funcionamiento, los comandos, diferentes parámetros, etc ... y después de todos estos conocimientos previos hemos creado una "Instancia" en AWS Educate.

Requisitos de la Instancia:

Info	Ubuntu Server 20.04
CPU virtual	1
Tipo Instancia	t2.small
Almacenamiento	Solo EBS
Crédito por hora de CPU	12$
Red	De Bajo a Moderado
Memoria GiB	2

Como podemos ver en la tabla anterior, hemos elegido el tipo de la instancia t2.small, que es suficiente para realizar este proyecto y así ahorrar los créditos (dinero) que nos ofrecen por ser estudiantes.

Amazon también nos ofrece más tipos de instancia, para más información Anexo 1.3.

Una vez que tengamos creada la instancia, vamos a modificar o habilitar algunas cosas que son importantes para este proyecto.

1. Habilitar los puertos de los servicios que vamos a utilizar.
2. Asociar una IP pública estática para vincular nuestro dominio "proyecto.ccff.site".

En este servidor es donde tendremos el formulario que ayuda a los clientes a crear los contenedores que contendrán toda la configuración hecha para tener su propio Wordpress, y eso solo rellenando un formulario que pueden llegar a tardar como mínimo 1 minuto.

Contenedores

Primero de todo, entramos a través de un control remoto ssh a nuestro servidor "Instancia", e instalamos el Podman. Creamos las carpetas **www-html** y **data.mariadb** en nuestro $HOME (puede ser una dirección específica para cada contenedor y cada usuario), esas dos carpetas serán enlazadas a los contenedores de podman, la cual tendrá los datos de la base de datos y el contenido de la página web de wordpress.

Descargamos la nueva versión de wordpress y descomprimimos su contenido dentro de la carpeta www-html.

```
ubuntu@ip-172-31-44-6:~$ ls
apaga.txt  data-mariadb  librerias  listaInstancias.csv  test.csv  www-html
ubuntu@ip-172-31-44-6:~$ ls www-html/wordpress/
index.php         wp-admin             wp-config.php  wp-links-opml.php  wp-settings.php
license.txt       wp-blog-header.php   wp-content     wp-load.php        wp-signup.php
readme.html       wp-comments-post.php wp-cron.php    wp-login.php       wp-trackback.php
wp-activate.php   wp-config-sample.php wp-includes    wp-mail.php        xmlrpc.php
ubuntu@ip-172-31-44-6:~$
```

Una vez que tengamos la configuración anterior, descargamos las imágenes oficiales de Podman para luego poder crear dos contenedores, las imágenes son **mariadb** y **php:7.4-apache**. Básicamente la imagen de mariadb es para la gestión de la base de datos y la otra contiene el servicio apache que nos servirá para Wordpress, a través de php gestionaremos algunas tareas como insertar usuarios en la base de datos, eliminar y crear contenedor.

```
ubuntu@ip-172-31-44-6:~$ sudo podman image ls
REPOSITORY                  TAG         IMAGE ID      CREATED       SIZE
localhost/php-mysql-apache  latest      bedb8367a219  8 days ago    424 MB
docker.io/library/mariadb   latest      2a2c18b8e036  13 days ago   412 MB
docker.io/library/php       7.4-apache  7ad4200bb99a  2 weeks ago   423 MB
k8s.gcr.io/pause            3.5         ed210e3e4a5b  2 months ago  690 kB
```

Contenedor Principal

A partir de aquí, creamos el contenedor principal, el cual podrá visualizar, crear, detener y eliminar a través del panel web. Hacemos que el tráfico dirigido al puerto 8000 del servidor, se redirija al puerto 80 de este contenedor, esto es muy importante ya que cuando el cliente se dirija a nuestro dominio con el puerto 8000 "proyecto.ccff.site:8000", automaticamente este tráfico se dirigirá al puerto 80 del contenedor y recogerá los datos necesarios para mostrar el Wordpress en el navegador con el puerto 8000 (se trata de una redirección de peticiones y respuestas).

```
ubuntu@ip-172-31-44-6:~$ sudo podman pod ls 2>/dev/null
POD ID        NAME    STATUS   CREATED             INFRA ID      # OF CONTAINERS
30851102b9db  admin   Running  About a minute ago  7a45a9ab66dd  3
ubuntu@ip-172-31-44-6:~$
```

Contenedor de Base de datos y apache-php

Crearemos dos contendores a partir de las imágenes que hemos descargado anteriormente (**mariadb - php:7.4-apache**) y aquí conjuntamente vamos a mapear las carpetas **www-html** y **data-mariadb**, hacemos esto para modificar los ficheros desde nuestro servidor "instancia", así no hará falta entrar en cada contenedor para modificar sus ficheros de configuración, se modifican desde afuera y los cambios se aplican automáticamente dentro de los contenedores porque las carpetas y los ficheros están mapeados. Estos dos contenedores pertenecerán al pod (contenedor principal) **admin**

php:7.4-apache:

```
ubuntu@ip-172-31-44-6:~$ sudo podman run -d --pod=admin -v /home/usuario/www-html/wordpress:/var/www/
html --name miapache_php docker.io/library/php:7.4-apache
```

mariadb:

```
ubuntu@ip-172-31-44-6:~$ sudo podman run -d --pod admin -v /home/usuario/data-mariadb:/var/lib/mysql
-e MYSQL_ROOT_PASSWORD="admin" -e MYSQL_DATABASE="wordpress" -e MYSQL_USER="andres" -e MYSQL_PASSWORD
="Andres_10" --name mimariadb docker.io/library/mariadb
```

Captura de las carpetas mapeadas:

```
ubuntu@ip-172-31-44-6:~$ ls data-mariadb/
aria_log.00000001  ib_logfile0  multi-master.info   wordpress
aria_log_control   ibdata1      mysql
ib_buffer_pool     ibtmp1       performance_schema
```

```
ubuntu@ip-172-31-44-6:~$ ls www-html/
wordpress
```

Iniciar contenedor y mostrar lista de contenedores:

```
ubuntu@ip-172-31-44-6:~$ sudo podman pod start admin
30851102b9db83d5a20a28dcb639c937ce736402456021311a8d1f335dddd4a6
ubuntu@ip-172-31-44-6:~$ sudo podman pod ls 2>/dev/null
POD ID        NAME    STATUS   CREATED         INFRA ID      # OF CONTAINERS
30851102b9db  admin   Running  15 minutes ago  7a45a9ab66dd  3
ubuntu@ip-172-31-44-6:~$
```

Ahora solo nos faltará iniciar el contenedor principal, entrar en el contenedor y agregar algunas librerías que son necesarias.

Automatización .yml

Podman nos ofrece algunos parámetros para generar archivos YAML de Kubernetes a partir de los pods existentes, gracias a esto podremos automatizar la creación de nuestro **contenedor principal**, este fichero incluirá todas las configuraciones y tareas que hemos realizado anteriormente.

Con el siguiente comando grabamos la configuración y servicios necesarios para la creación del contenedor (en este caso, el pod "admin") en un fichero **.yml**

```
ubuntu@ip-172-31-44-6:~$ sudo podman kube -f wordpress.yml admin
```

Ahora cuando un cliente accede a nuestro formulario para crear un contenedor, en ves de realizar toda la configuración e implementar los servicios, lo que haremos es ejecutar el fichero **.yml**, con esto logramos automatizar todas las tareas y configuraciones necesarias para tener su página web wordpress funcionando en los contenedores.

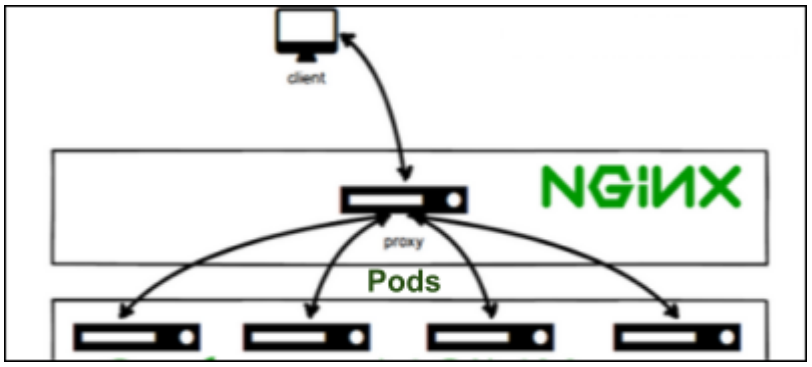
Con la siguiente orden creamos un contenedor a aprtir de un fichero .yml:

```
ubuntu@ip-172-31-44-6:~$ sudo podman play kube wordpress.yml
```

Para ver la configuración más detalladamente Anexo 1.2

Contenedor NGINX

Vamos a utilizar NGINX como proxy inverso, en este caso nuestro contenedor proxy se ubicará delante de todos los contenedores creados por los clientes y gestionará el envío de peticiones. En vez de enviar las peticiones a un puerto específico, se enviarán al servicio de NGINX, el cual a través del dominio (host) sabrá a qué contenedor enviar cada petición.



Para la configuración del contenedor proxy, necesitaremos la imagen oficial de NGINX.

Crearemos una carpeta **"nginxProxy"** en nuestra instancia, esa carpeta la mapeamos "**sincronizamos**" con la carpeta de configuración del servicio NGINX (contenedor) y así la modificación se hará desde la máquina real.

Dentro de esta carpeta creamos un fichero para los **logs** (visualizar los futuros errores), y un archivo llamado **default.conf** (no debemos cambiar el nombre de este fichero).

Al crear el contenedor NGINX, tenemos que sincronizar la carpeta creada anteriormente y la del log.

```
ubuntu@ip-172-31-44-6:~$ sudo podman run -d -p 9000:80 -v /home/ubuntu/nginxProxy:/etc/nginx/conf.d -
v /home/ubuntu/nginxProxy/logs:/var/log/nginx --name nginxproxy nginx
cafd5d4ddc92e9a1fcd22c11f8ec98d6ac597a5bca07b26dabc77250ffc62b49
ubuntu@ip-172-31-44-6:~$
```

Test en el contenedor proxy:
Enviamos una petición a un contenedor "pod" con su puerto específico

```
ubuntu@ip-172-31-44-6:~$ sudo podman container exec -it nginxproxy bash
root@cafd5d4ddc92:/#
root@cafd5d4ddc92:/# curl http://proyecto.ccff.site:8000/
<!doctype html>
<html lang="en-US" >
<head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <title>miweb &#8211; Just another WordPress site</title>
<meta name='robots' content='max-image-preview:large' />
<link rel='dns-prefetch' href='//s.w.org' />
<link rel="alternate" type="application/rss+xml" title="miweb &raquo; Feed" href="http://proyecto.ccf
f.site:8000/index.php/feed/" />
<link rel="alternate" type="application/rss+xml" title="miweb &raquo; Comments Feed" href="http://pro
yecto.ccff.site:8000/index.php/comments/feed/" />
```

El fichero **default.conf** nos servirá como fichero de configuración del servicio NGINX
***Nota**: podemos copiar la configuración que está por defecto en Nginx. File*

Directiva de ejemplo para nuestro dominio en el fichero default.conf :

```
    location ^proyecto.ccff.site {
        proxy_pass http://127.0.0.1:8000;
#       rewrite ^proyecto.ccff.site(.*)$ /$1 break;
    }
```

Con esto logramos que cuando nos llegue una petición de un cliente, y además tiene como dominio **proyecto.ccff.site**, envía la solicitud a un contenedor web específico, gracias a la directiva **proxy_pass.**

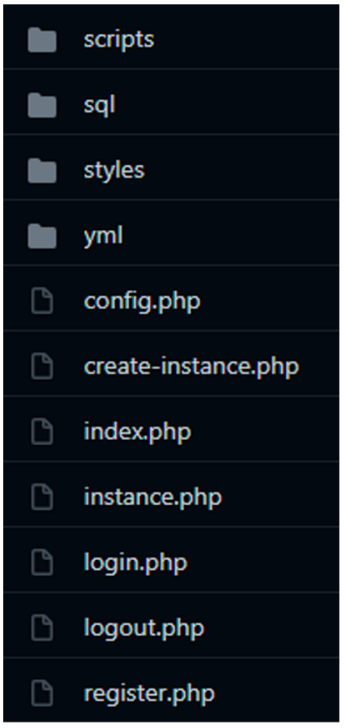
Todo lo explicado anteriormente se ha realizado de forma manual. Estamos añadiendo a mano la localización de cada contenedor y que dirija las peticiones por su dominio. Esto lo podemos automatizar a través del formulario web, el cual nos crea automáticamente el contenedor pod, pero tanto el proceso manual como automatizarlo no nos ha funcionado. Error - NGINX

Página web

La página web es muy importante, ya que es el intermediario entre el cliente y el servidor, es la parte visual del servicio que ofreceremos a los usuarios, ya que a través de este panel web, el usuario podrá crear, modificar y eliminar sus contenedores.

Hemos visto la necesidad de hacer la página web accesible para todos los formatos, que sea completamente responsive, gracias a esto se podrá visualizar de una forma correcta, tanto en un ordenador como en un smartphone.

La web será sencilla e intuitiva, tendremos las páginas con las que el usuarios podrá registrarse, iniciar sesión, crear o eliminar sus contenedores, una página de inicio y una página de información, sin olvidar las carpetas que guardarán los scripts, ficheros .sql, hojas de estilo y ficheros .yml.



Dir. para guardar los **scripts**.

Dir. para guardar los ficheros .**sql**.

Dir. para los ficheros **.css** de la web.

Dir. para almacenar los **.yml**.

Página para la **conexión** con la BD.

Página de **creación** de contenedores.

Página de **inicio** de la web.

Página de **gestión** de los contenedores.

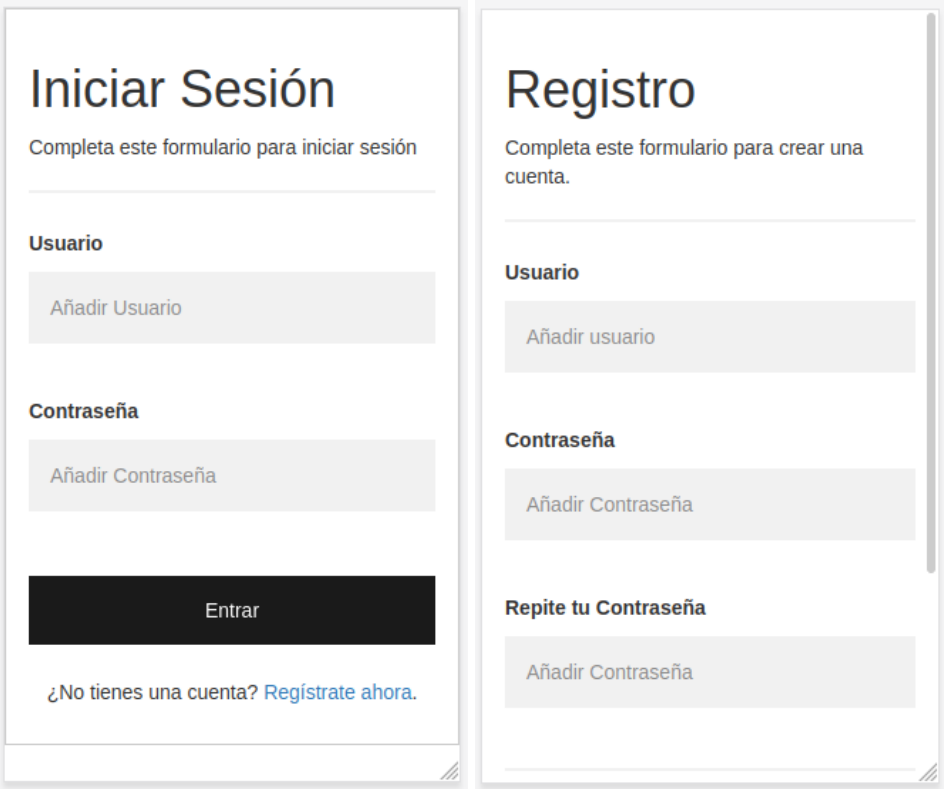
Página para **iniciar sesión** en la web.

Página para **cerrar sesión**.

Página para **darse de alta** en la web.

Hemos desarrollado la página web utilizando el editor de código visual studio a través de los lenguajes **html** y **css** para la construcción, maquetación y diseño de la web (los estilos css son de la página w3school), y **php** para la comunicación con la base de datos MySQL del servidor de aws.

Página de Registro / Inicio de sesión
código fuente: login.php, register.php



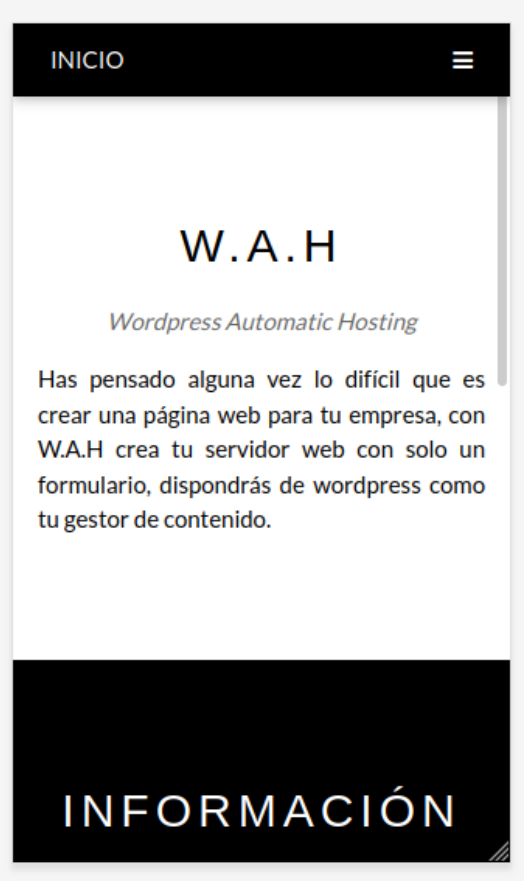
Registro: Antes de crear un usuario, se comprueba en la BD que el nombre de usuario no exista, esto lo haremos con un **select** que nos devuelva la lista de usuarios. A partir de aquí, si no hay ningún usuario registrado con ese nombre, **insertará** los valores en la tabla users de la base de datos.

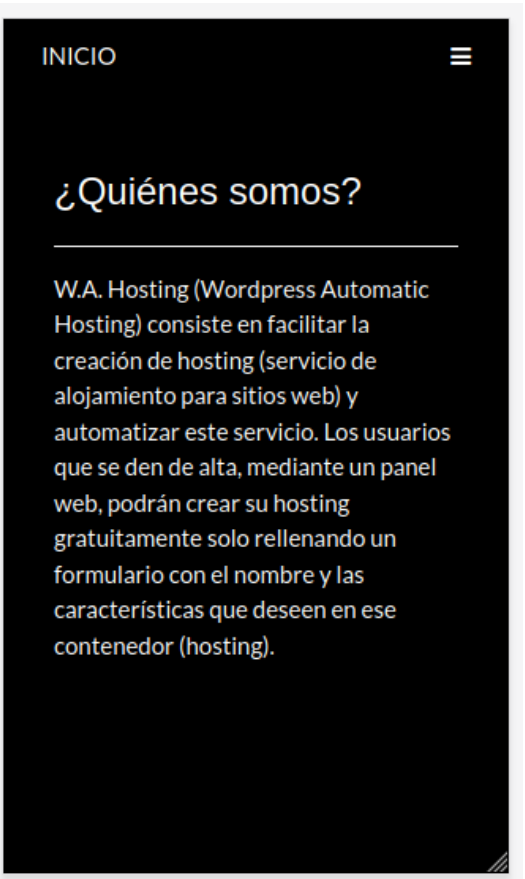
Iniciar Sesión: Para que el usuario pueda iniciar sesión correctamente, primero haremos un select a partir del nombre de usuario añadido, si el select nos devuelve un valor, pasaremos a comprobar si la contraseña añadida por el usuario es igual a la que está en el campo password de la tabla users. A partir de aquí, si la contraseña es correcta, iniciamos sesión y nos dirigimos a la página de inicio.

Una vez iniciada la sesión los usuarios tendrán acceso al resto de páginas y podrán navegar por el sitio web. Tendrán el acceso restringido al menos que dispongan de un usuario y hayan iniciado sesión.

```php
// chequear si está la sesión iniciada
if(!isset($_SESSION["loggedin"]) || $_SESSION["loggedin"] !== true){
    header("location: login.php");
    exit;
}
```

Página de inicio
código fuente: [index.php]

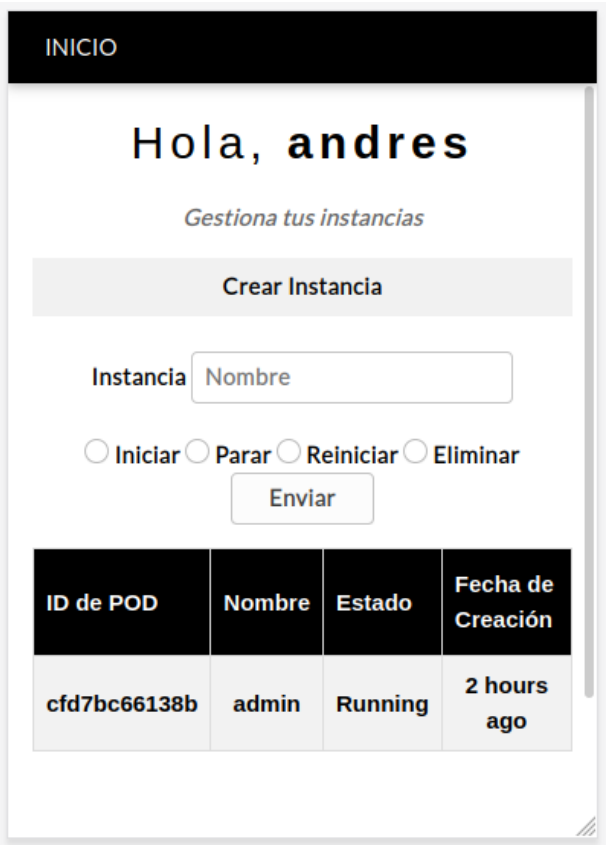




Una vez registrados e iniciamos la sesión, seremos dirigidos automáticamente a esta página,

Página de inicio o de información, donde explicamos qué es lo que hacemos y qué servicio ofrecemos al usuario/cliente dado de alta.

Página para gestionar contenedores
código fuente: instance.php



Esta página será la más importante de la web, ya que será la más utilizada por el usuario. Disponemos de un botón que nos redirigirá a la página de creación del contenedor (Sólo a partir de esta podremos acceder).

También se visualizará la lista de contenedores ya creados anteriormente por él, y a partir de su nombre se podrá iniciar, parar, reiniciar y eliminar dicha instancia.

Lo ideal hubiera sido que la gestión sea automática (no añadiendo el nombre de la instancia a modificar), que nos muestre botones que con un simple click se inicie, detenga, reinicie o elimine el contenedor. Por falta de tiempo no se pudo mejorar el código para implementar esta gestión.

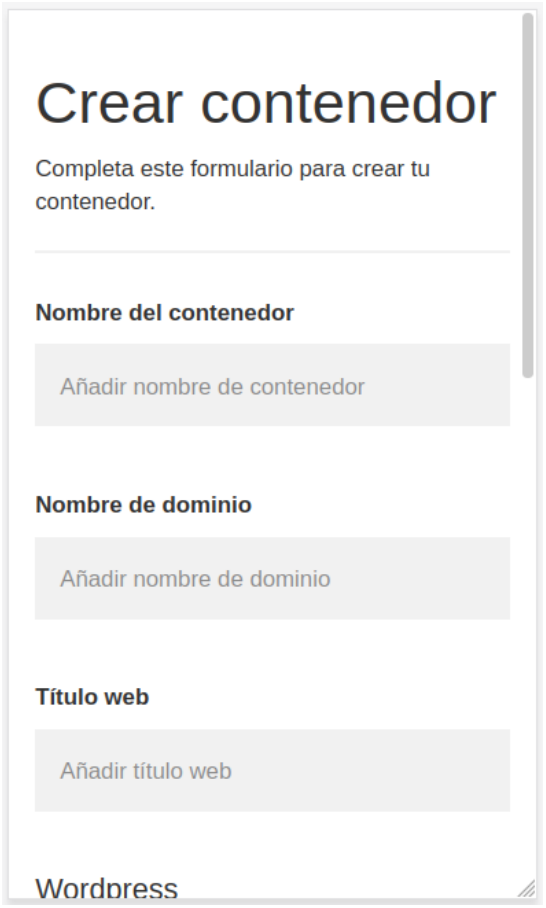
Para disponer de una lista de contenedores actualizada, lo que hacemos es crear un fichero temporal .csv con los datos personalizados (a partir de expresiones regulares) que nos muestra el comando *podman pod ls* (comando para listar los contenedores "pod").

```php
<?php
    $fila = 1;
    shell_exec("sudo podman pod ls | sed 's/  /,/g' | sed '1d' | cut -d',' -f1,2,3,4 > /home/ubuntu/listaInstancias.csv");

    if (($gestor = fopen("/home/ubuntu/listaInstancias.csv", "r")) !== FALSE) {
        echo '<table id="customers">';
        echo '<tr><th>ID de POD</th><th>Nombre</th><th>Estado</th><th>Fecha de Creación</th></tr>';
        while (($datos = fgetcsv($gestor, 1000, ",")) !== FALSE) {
            $numero = count($datos);
            $fila++;
            echo "<tr>";
            for ($c=0; $c < $numero; $c++) {
                echo "<td>".$datos[$c] ."</td>";
            }
            echo "</tr>";
        }
        echo '</table>';

        fclose($gestor);
    }
    ?>
```

Página para crear contenedor
código fuente: [create-instance.php]



La creación del contenedor se hará a partir de un fichero .yml, este nos creará un pod con 2 contenedores (mariadb y apache) necesarios para que el usuario tenga su página web con wordpress.

A través de un script haremos una actualización de valores en la BD de wordpress, como por ejemplo: actualizar el título de la web y el nombre y contraseña del usuario de administración, para que el usuario pueda iniciar sesión en wordpress y editar/crear dentro de su página web.

Tanto la BD y los ficheros wordpress de cada contenedor se encuentran enlazados en dos carpetas externas en el servidor.

Crear contenedor

Creamos una copia del fichero .yml genérico y en la copia temporal reemplazamos la palabra **nombreContainer** por el nombre introducido por el usuario en el formulario.

A continuación creamos el contenedor “pod” a partir de este fichero .yml con el comando *podman play kube*

```
// reemplazar el nombre del pod por defecto
copy($file_sample, $copy);
file_put_contents($copy,str_replace('nombreContainer',"$nombre_contenedor",file_get_contents($copy)));

// Creación de pod con 3 contenedores
shell_exec("sudo podman play kube /var/www/site/html/yml/container.yml");
shell_exec("rm -rf yml/container.yml");
```

Actualizar datos del contenedor

Con esta consulta sql estamos realizando un update sobre la tabla **wp_options** (en la cual WordPress guarda los ajustes de configuración). En este caso solo queremos cambiar el título de la web por el valor introducido por el usuario, esto lo hacemos filtrando por el valor "blogname" dentro de la casilla **option_name**

```
$titulo_web = $_POST["titulo_web"];
// preparando sql para actulizar el título de la web de wordpress
$sql = "UPDATE wp_options SET option_value = '$titulo_web'
        WHERE option_name = 'blogname';" . "\n";
```

Generamos otra consulta sql a partir de los datos de usuario de wordpress introducidos por el usuario a través del formulario. Hacemos un update sobre la tabla **wp_users** (en la cual WordPress guarda los datos de los usuarios registrados). En este caso queremos actualizar los valores del administrador, para lograr esto filtramos por el id "1", el cual nos indica el primer usuario dado de alta, y por lo mismo el administrador de la página web de wordpress.

```
$username = $_POST["username"];
$nombre_dominio = $_POST["nombre_dominio"] . ".proyecto.ccff.site:8080";
$email = $_POST["email"];
$password = $_POST["password"];

// preparando sql para actualizar el usuario de administrador de wordpress
$sql = "UPDATE wp_users SET user_login = '$username',
        user_pass = MD5('$password'), user_email = '$email',
        user_url = '$nombre_dominio'
        WHERE ID = 1;" . "\n";
```

Por último escribimos en un fichero .sql las dos consultas creadas y ejecutamos un script, el cual ejecutará un comando dentro del contenedor de la BD que actualizará los datos de las dos tablas a partir del fichero .sql

```
fwrite($fileSQL, $sql);
fclose($fileSQL);

sleep(2);
shell_exec("sudo sh scripts/script.sh $nombre_contenedor");
```

```
podman exec -i $1-mariadb bash -c 'exec mysql -uandres -p"Andres_10"' < sql/update.sql
```

Menú de navegación

El menú es importante, ya que permite que los usuarios puedan seleccionar la parte de nuestro sitio que desean visitar. El usuario deberá moverse con soltura y facilidad por las distintas páginas del sitio. Aunque no hay mucho en lo cual navegar, debe encontrar lo que busca rápidamente.

Disponemos de un menú responsive, el cual nos ofrece navegación a las páginas:

- **Inicio**: es importante que el usuario pueda volver al inicio en un click.
- **Instancias**: página importante, ya que a través de ella, el usuario podrá gestionar y crear sus contenedores
- **Salir**: con un click el usuario podrá cerrar su cesión.

Nota: ¿Por qué los comandos podman se ejecutan como root en el formulario php? Errores - Podman.

Problema y Errores

Problemas

El gran problema que hemos tenido en el proyecto, es la falta de tiempo, esto fue debido a los errores que hemos tenido con el servicio podman, por cada error corregido nos salía uno nuevo y hemos estado varias semanas junto a oscar (tutor de proyecto) investigando la manera de resolver estos errores. Gracias a esto nos hemos visto obligados a recortar nuestras expectativas y conformarnos.

Errores

Podman

Hemos tenido varios problemas a la hora de ejecutar los comandos de podman desde la página web.

El cliente que se loguee en nuestra página web, al intentar crear un contenedor (instancia) lo hará a través del usuario de apache (www-data), este usuario al ejecutar los comandos podman para la creación del contenedor, no podrá hacerlo, ya que podman no se puede ejecutar a través de un usuario nologin (que no se haya logueado en el sistema). Con el siguiente comando permitimos al usuario de apache ejecutar comandos sin haber iniciado sesión.

```
loginctl enable-linger www-data,
```

Ahora ejecutamos un comando de podman como php y con usuario de apache (www-data).

```
sudo -u www-data php -r "system('podman version');"
```

Salida del comando:

```
ubuntu@ip-172-31-44-6:~$ sudo -u www-data php -r "system('podman version');"
Error: error creating runtime static files directory: mkdir /var/www/.local/share: permission denied
```

Este error significa que no somos propietarios o que no tenemos los permisos necesarios para ejecutar los comandos de podman con php.

Solución: Dar permisos necesarios para poder avanzar el proceso.

```
sudo chmod -R o+rwx /usr/www/.local/share
```

Cuando volvemos a ejecutar el comando, nos aparece un nuevo error

```
Error : chown /etc/containers/strorage.conf: operation not permitted
```

Solución: Habrá que modificar el fichero "storage.conf", cambiar la ruta de almacenamiento de los contenedores, esta ruta será propiedad del usuario apache y pueda ver, modificar y ejecutar dentro de esta.

```
#rootless_storage_path = "$HOME/.local/share/containers/storage"

rootless_storage_path = "/opt/apache"
```

Ejecutamos el comando de podman con el usuario de apache, y comprobamos que se ha solucionado el error.

```
ubuntu@ip-172-31-44-6:~$ sudo -u www-data php -r "system('podman version');"
Version:      3.0.1
API Version:  3.0.0
Go Version:   go1.15.2
Built:        Thu Jan  1 01:00:00 1970
OS/Arch:      linux/amd64
ubuntu@ip-172-31-44-6:~$
```

Para comandos simples como el anterior no habrá ningún problema, pero al intentar bajar imágenes y crear pods o contenedores con podman (a través del usuario de apache) nos seguirá dando error (estos errores no son específicos).

Debemos instalar el paquete **uidmap** y debemos configurar un **subuid** y un **subgid** para el usuario de apache (usuario que utilizará podman), esta información deberemos almacenarla en /etc/subuid y /etc/subgid

```
ubuntu@ip-172-31-44-6:~$ cat /etc/subgid
ubuntu:100000:65536
profe:165536:65536
u1:231072:65536
www-data:296609:65536
```

```
ubuntu@ip-172-31-44-6:~$ cat /etc/subuid
ubuntu:10000:65536
www-data:75536:65536
```

Para que estos cambios se guarden ejecutamos el comando:

podman system migrate

A partir de aquí nos permitirá bajar imágenes, pero al crear contenedores a partir de una imagen bajada, nos seguirá dando errores de permisos o operaciones no permitidas. Investigando nos dimos cuenta que la única manera de solucionar con totalidad todos los errores que nos daba, era darle privilegios de root al usuario de apache (Dicho de otra forma, darle permisos en todo el sistema.)

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
www-data ALL=(ALL:ALL) NOPASSWD: ALL
```

Esta solución no es nada recomendable pero necesaria para el buen funcionamiento de podman con el usuario www-data.

Los comandos podman de prueba que hemos ido ejecutando para solucionar los errores, se han ejecutado a través de un terminal, pero ¿qué pasa si se hacen a través

de un formulario web? en teoría no debería ser diferente, ya que aunque ejecutemos el comando podman por terminal, lo hacemos con php a través del usuario de apache.

A través de un formulario web, creamos un pod con 3 contenedores, a partir de un fichero .yml

```
shell_exec("sudo podman play kube /var/www/site/html/yml/container.yml");
```

Si intentamos visualizar el pod y los contenedores a través del terminal, nos dará las siguientes líneas de errores.

```
ubuntu@ip-172-31-44-6:~$ sudo podman pod ls
ERRO[0000] error joining network namespace for container 81e15050ced5b0d7e74849688d30f3bdc53b779c308f01c8074e00bed62b7f77: error retrieving network namespace at /run/netns/cni-a398b889-ee30-d7ff-ad57-d61358656385: unknown FS magic on "/run/netns/cni-a398b889-ee30-d7ff-ad57-d61358656385": 1021994
ERRO[0000] error joining network namespace for container 81e15050ced5b0d7e74849688d30f3bdc53b779c308f01c8074e00bed62b7f77: error retrieving network namespace at /run/netns/cni-a398b889-ee30-d7ff-ad57-d61358656385: unknown FS magic on "/run/netns/cni-a398b889-ee30-d7ff-ad57-d61358656385": 1021994
ERRO[0000] error joining network namespace for container 81e15050ced5b0d7e74849688d30f3bdc53b779c308f01c8074e00bed62b7f77: error retrieving network namespace at /run/netns/cni-a398b889-ee30-d7ff-ad57-d61358656385: unknown FS magic on "/run/netns/cni-a398b889-ee30-d7ff-ad57-d61358656385": 1021994
POD ID        NAME    STATUS   CREATED         INFRA ID      # OF CONTAINERS
750547343dd3  prueba  Running  18 seconds ago  81e15050ced5  3
```

```
ubuntu@ip-172-31-44-6:~$ sudo podman container ls
ERRO[0000] error joining network namespace for container 81e15050ced5b0d7e74849688d30f3bdc53b779c308f01c8074e00bed62b7f77: error retrieving network namespace at /run/netns/cni-a398b889-ee30-d7ff-ad57-d61358656385: unknown FS magic on "/run/netns/cni-a398b889-ee30-d7ff-ad57-d61358656385": 1021994
Error: error joining network namespace of container 81e15050ced5b0d7e74849688d30f3bdc53b779c308f01c8074e00bed62b7f77: error retrieving network namespace at /run/netns/cni-a398b889-ee30-d7ff-ad57-d61358656385: unknown FS magic on "/run/netns/cni-a398b889-ee30-d7ff-ad57-d61358656385": 1021994
```

Nota: estos errores no aparecen si visualizamos los comandos desde el formulario web.

```
CONTAINER ID  IMAGE                         COMMAND               CREATED        STATU
81e15050ced5  k8s.gcr.io/pause:3.5                                5 minutes ago  Up 5
271a08c8d35a  localhost/php-mysql-apache    apache2-foregroun...  5 minutes ago  Up 5
afacabc4c75b  docker.io/library/mariadb     mysqld                5 minutes ago  Up 5
```

```
POD ID        NAME    STATUS   CREATED        INFRA ID      # OF CONTAINERS
750547343dd3  prueba  Running  6 minutes ago  81e15050ced5  3
```

Esto puede ser debido al **stderr** (Standard Error), es el canal por el que se envía un mensaje de error en caso de que la ejecución del comando falle. Esto quiere decir que los mensajes de error se muestran solo por el terminal, si ejecutamos el comando por el formulario, a través de una página web, estos mensajes de error no se visualizarán.

Para la solución de estos problemas, se creó una incidencia en el github oficial de podman, por el cual obtuvimos la mayoría de soluciones descritas anteriormente.
https://github.com/containers/podman/issues/10308

Solución final

4 días antes de presentar el proyecto, se logró solucionar todos los errores que nos daba podman al ejecutar sus comandos a través de un usuario nologin (como apache).

***IMPORTANTE**: La solución para el proyecto ha sido dar privilegios de root al usuario apache, ya que por falta de tiempo no decidimos implementar esta nueva solución. Para futuros proyectos con podman similares a este, la solución perfecta será la explicada a continuación.*

Pasos (provisionales) hechos en un fedora

1. PROBABLEMENTE NO NECESARIO. Desactivar SELinux poniendo la línea:
   SELINUX = disabled en *"/etc/selinux/config"*
2. Instalar los paquetes httpd php
3. Añadir la siguiente línea en los ficheros *"/etc/subuid"* y *"/etc/selinux/config"*
   apache: 165537: 65536
4. Cambiamos los permisos de la carpeta personal del usuario de apache, en este caso *"/usr/share/httpd"* con *sudo chown -R apache: apache /usr/share/httpd*. Para comprobar cuál es la carpeta personal de un usuario, podemos visualizar el fichero *"/etc/passwd"*
5. Ejecutamos *sudo chsh -s /bin/bash apache* (de "util-linux-user" package) para dar al usuario de apache un intérprete de órdenes (muy mala idea, pero no es importante para nosotros, funciona)
6. Con el comando *sudo loginctl enable-linger apache* le decimos a systemd que inicie una sesión para el usuario apache, con esto el usuario dejará de ser nologin y podman no tendrá problemas al ejecutar sus comandos, ya que solo los usuarios con sesión iniciada podrán ejecutar comandos de este servicio.
7. Reiniciar el sistema.

Si seguimos la guía tal cual y no nos da ningún error, podremos ejecutar cualquier comando podman. Esto lo podemos probar ejecutando el comando *sudo -i -u apache php -r "system ('versió podman');".* También podremos ejecutar órdenes como *"podman run..."* y funcionará correctamente.

NGINX

Al implementar la configuración explicada en el apartado de Desarrollo - NGINX nos damos cuenta que no funciona

```
ubuntu@ip-172-31-44-6:~$ sudo podman container ls -a
ERRO[0000] error joining network namespace for container ae8d7188663d997aac78fd91d2ba263f8950c
7cff2ad30: unknown FS magic on "/run/netns/cni-b27ba9ea-537d-35ac-2912-a147cff2ad30": 1021994
Error: error joining network namespace of container ae8d7188663d997aac78fd91d2ba263f8950c5441b
ad30: unknown FS magic on "/run/netns/cni-b27ba9ea-537d-35ac-2912-a147cff2ad30": 1021994
ubuntu@ip-172-31-44-6:~$
```

Al crear el contenedor proxy y configurarlo, todos los contenedor “pods” se ponen en estado **degraded** (estado crítico).

ID de POD	Nombre	Estado	Fecha de Creación
cfd7bc66138b	admin	Degraded	15 minutes ago

Esto puede ser debido a las muchas modificaciones que se hicieron en podman para tratar de resolver los errores de permisos e intentar ejecutar correctamente los comandos de podman con el usuario de apache. Esto se encuentra mejor explicado en Errores - Podman.

No tuvimos el tiempo suficiente para investigar y realizar las pruebas necesarias. La implementación de NGINX se añadirá a la lista de futuros propósitos. Mejoras.

Mejoras

Al final no conseguimos completar al 100% todos los objetivos del proyecto. Para el futuro lo primero que se hará es completar la siguiente lista de propósitos:

- Implementar correctamente **NGINX** como proxy inverso para que envíe las peticiones a los contenedores a partir de su dominio (host), con esto conseguiremos que todas las peticiones a todos los contenedores no vayan a un puerto específico, si no que actúe NGINX como servicio que envía las petición a cada subdominio establecido.
    - Con esto lograremos crear varios contenedores y que las peticiones no se pierdan o confundan, ya que van al mismo puerto y no hay ninguna regla establecida para dirigir las peticiones por su subdominio (cada contenedor tendrá su subdominio ***.proyecto.ccff.site**).
    - Configurar la zona DNS de nuestro dominio. El contenedor creado por el usuario deberá enlazarse a un subdominio. Cuando aplicamos el proxy inverso, independientemente de a qué subdominio se dirigen las peticiones, todas irán al servidor NGINX.
    - Seguramente habrán muchas otras cosas que configurar y tener en cuenta, por eso se irá investigando conforme la implementación.
- Mejorar la creación de contenedores automáticamente. Aunque no obtenemos ningún problema al realizar la creación a partir del formulario, creemos que es posible la mejora y optimización de este proceso.
    - Primero podríamos implementar la creación de las **carpetas mysql y apache** para el contenedor principal (el cual crea el usuario a partir del .yml). A fecha de doy el .yml no crea ni copia las carpetas para cada contenedor, si no que utiliza dos carpetas ya creadas anteriormente por nosotros, es importante saber que para cada contenedor tiene que haber dos carpetas mapeadas, estas guardarán los datos de la BD y el apache del contenedor. No nos vimos con la necesidad de hacer esto, ya que para realizarlo deberíamos implementar correctamente el proxy inverso. + info
    - Mejorar el .yml para que **los contenedores se creen con una red interna**, la cual nos será útil para la implementación de varios contenedores a la vez, ya que estas IPs estarán enlazadas a un subdominio, y gracias a esto el proxy inverso podrá enviar los paquetes a un contenedor específico.

    - A la hora de crear un nuevo pod el formulario web tendrá que **comprobar que no haya ya un pod con el mismo nombre** (subdominio), en ese caso se le ofrece al usuario poder escribir otro nombre.

    - Optimizar el hecho de que por cada pod se tenga que copiar una réplica de las carpetas mysql y apache. Investigaremos la manera de hacer algún tipo de carpeta **"compartida única" entre pods** para que cada pod nuevo partiera del mismo contenido inicial común y entonces solo adaptara los cambios propios. De esta manera se tendría un uso mucho más eficiente del espacio en disco.

- Mejoras en el código fuente de la web

    - Aunque la web no nos da problemas, y a priori todo funciona correctamente, si pasamos nuestro código por un validador de html, css y php nos saldrían varias alertas. Tendremos que validar al 100% todo el código que por necesidades, errores y problemas mayores no tuvimos en cuenta.

    - En php estamos ejecutando **comandos como sudo** (usuario con privilegios de root), esto es una brecha de seguridad importante, pero lo hicimos debido a los múltiples errores de podman. Al final se obtuvo una solución que no implementamos en el proyecto por falta de tiempo. Solución

    - La lista de contenedores que nos muestra la página web no está enlazada al usuario, lo que quiere decir que nos muestra la lista general de pods disponibles en el servidor. Esto es fácil de solucionar, podríamos crear un fichero de registro que nos guarde la lista de pods que crea cada usuario y recorrer este registro dependiendo del usuario que haya iniciado la sesión (solución simple pero que funciona). Se investigará mejores opciones.

Al realizar toda esta larga lista de requisitos, obtendremos un servicio de hosting automático completo, pero con pequeños detalles que pulir. Una vez realizado todo esto mejoraremos las características del servidor o migraremos a otro mejor, para que los usuarios ya dados de alta tengan una mejor experiencia, en cuanto a velocidad y rendimiento.

Como final y última mejora a realizar, vemos con buenos ojos que el usuario pueda mejorar las características de los contenedores en caliente o cambiar algún ajuste en concreto del pod, pero para realizar todo esto debemos investigar más en profundidad el servicio podman, porque al ser una tecnología nueva, puede que no sea posible realizar estas mejoras.

Conclusiones

Conclusiones generales del proyecto

Podemos asegurar que este proyecto ha servido en primer lugar, para mejorar la habilidad de trabajar en equipo, hemos realizado tareas diferentes pero unidas en un mismo proyecto, los dos hemos aportado todo lo que sabemos hacer hasta ahora e incluso hemos aprendido muchas cosas gracias a la investigación de nuevos servicios.

Teníamos conocimientos previos para crear una página web e incluso pensamos en la opción de crear nuestra web a través de WordPress, pero vimos la necesidad de hacerlo nosotros mismos, para investigar y aprender más del tema. Investigamos cómo crear una página de registro e inicio de sesión, y restringir el acceso a usuarios no registrados previamente. Todo esto se logró gracias al lenguaje PHP, el cual nos ayudó a crear páginas dinámicas y también a conectar la web con la base de datos.

Donde más aprendimos fue en la creación de contenedores con podman, ya que es un servicio moderno y que en un futuro puede que supere a docker como motor de contenedores. Esta investigación nos cambió la manera de trabajar y nos dimos cuenta que este tipo de servicio se puede implementar en cualquier trabajo o tarea. Por ejemplo, muchas veces como administradores vemos la necesidad de realizar pruebas, las cuales si hacemos en nuestra máquina real puede que afecte al sistema, para eso utilizamos máquinas virtuales (VM). Como las VM, los contenedores nos permiten empaquetar el servicio o aplicación que implementemos dentro y nos ofrece un entorno aislado para ejecutarlos, pero la gran diferencia es que los contenedores nos ofrecen una unidad mucho más liviana para que trabajemos los administradores o desarrolladores, además de una gran cantidad de beneficios. Otro beneficio es la automatización al crearlos, ya que si nosotros configuramos un contenedor para que nos ofrezca un servidor web, podemos exportar todo lo hecho (servicios instalados, configuración) a un fichero .YML que podemos ejecutar en otro sistema operativo y disponer del mismo contenedor, con esto no haría falta volver a implementar los servicios y configurar los ficheros necesarios para su buen funcionamiento.

Como reflexión y crítica a nosotros mismo, hemos de decir que algunos de los objetivos planteados inicialmente se hicieron con mucho optimismo, creemos que hemos hecho un gran trabajo pero algunas tecnologías, como NGINX (proxy inverso), no han sido implementadas por falta de tiempo y práctica.

En conclusión, el tiempo dedicado al proyecto no ha sido en vano, ya que hemos aprendido como crear una página web semi-profesional, crear contenedores y automatizarlos a través de un formulario web y además de eso que ofrezcan servicios, como puede ser un servidor web con wordpress para el usuario. No hubiéramos logrado esto sin todos los conocimientos adquiridos en el grado, sobretodo en este último curso.

Se continuará trabajando para que la página web acabe siendo aquello que soñamos al iniciar el proyecto y que por poco no hemos conseguido, de igual manera quedamos contentos porque en poco tiempo hemos podido realizar un servidor hosting sencillo y robusto, el cual continuaremos mejorando.

Anexos

Anexo 1.1

Guía para registrarse en AWS Educate

Como crear una instancia en AWS Educate

Anexo 1.2

1. Instalación (Ubuntu 20.04)

Paso 1 : Asegúrese de que algunos datos de identificación del sistema estén disponibles (usaremos VERSION_ID). Al ejecutar este comando, no nos tiene que salir ningún error.

```
$ source /etc/os-release
```

Paso 2 : Agregue el repositorio de paquetes de Podman a Apt.

```
$ sudo sh -c "echo 'deb
http://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/xUbuntu_${V
ERSION_ID}/ /' > /etc/apt/sources.list.d/devel:kubic:libcontainers:stable.list"
```

Paso 3 : Bajamos los contenedores estables que están asociados con podman.

```
$ wget -nv
https://download.opensuse.org/repositories/devel:kubic:libcontainers:stable/xUbuntu_${VE
RSION_ID}/Release.key -O- | sudo apt-key add -
```

Paso 4 : Actualizar los repositorios e instalamos.

```
$ sudo apt-get update
$ sudo apt-get install podman
```

2. Comandos (Contenedores)

Para realizar este proyecto no hace falta saber todas las instrucciones que nos dispone el Podman, en sí hay muchos parámetros, pero os dejo una Guia con significados de cada parámetros.

Guia Oficial : http://docs.podman.io/en/latest/Commands.html

Podman Wordpress

Carpetas (mkdir) :

Importante : crear sin sudo

```
$ mkdir www-html
$ mkdir data-mariadb
```

cd www-html

```
$ curl -o wordpress.tar.gz https://wordpress.org/wordpress-latest.tar.gz
```

```
$ tar -xzf wordpress.tar.gz
```

Bajamos los contenedores de mysql y php :

```
$ sudo podman pull docker.io/library/mariadb
$ sudo podman pull docker.io/library/php:7.4-apache
```

Creamos un pod dentro de la carpeta www-html:

```
$sudo  podman pod create -n admin -p 8000:80
```

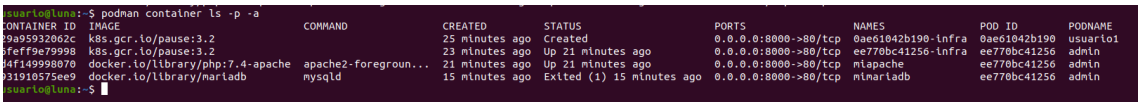
Ahora añadimos las imágenes anteriores en nuestro pod :

```
$ sudo podman run -d --pod=admin -v /home/usuario/www-html/wordpress:/var/www/html
--name miapache_php docker.io/library/php:7.4-apache
```

```
$ sudo podman run -d --pod admin -v /home/usuario/data-mariadb:/var/lib/mysql -e MYSQL_ROOT_PASSWORD="admin" -e MYSQL_DATABASE="wordpress" -e MYSQL_USER="andres" -e MYSQL_PASSWORD="Andres_10" --name mimariadb docker.io/library/mariadb
```

Confirmamos que ha añadido sin errores :

```
$ podman container ls -p -a
```

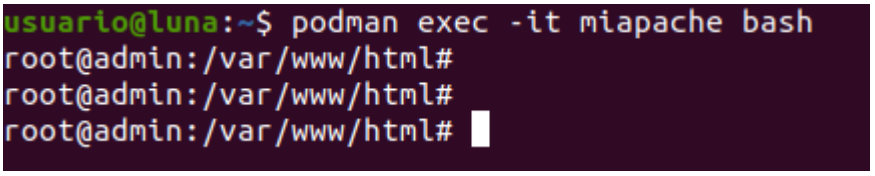


Arrancamos el pod :

```
$ sudo podman pod start admin
```

Ejecutamos bash del pod :

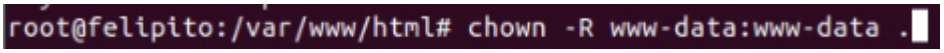
```
$ sudo podman exec -it miapache_php bash
```



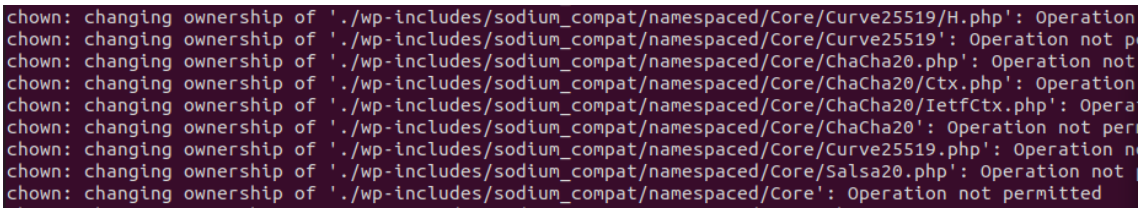
Ejecutamos el siguiente comando donde añadiremos las librerías que nos faltan :

```
$ docker-php-ext-install mysqli pdo pdo_mysql
$ docker-php-ext-enable mysqli pdo pdo_mysql
```

Damos permisos a la carpeta (/var/www/html) :



Si os sale este error **(no operating), son los permisos de las carpetas de fuera :**



Solución : Salimos del pod y a las dos carpetas de fuera, le cambiamos el propietario, en caso que salga error parecido:

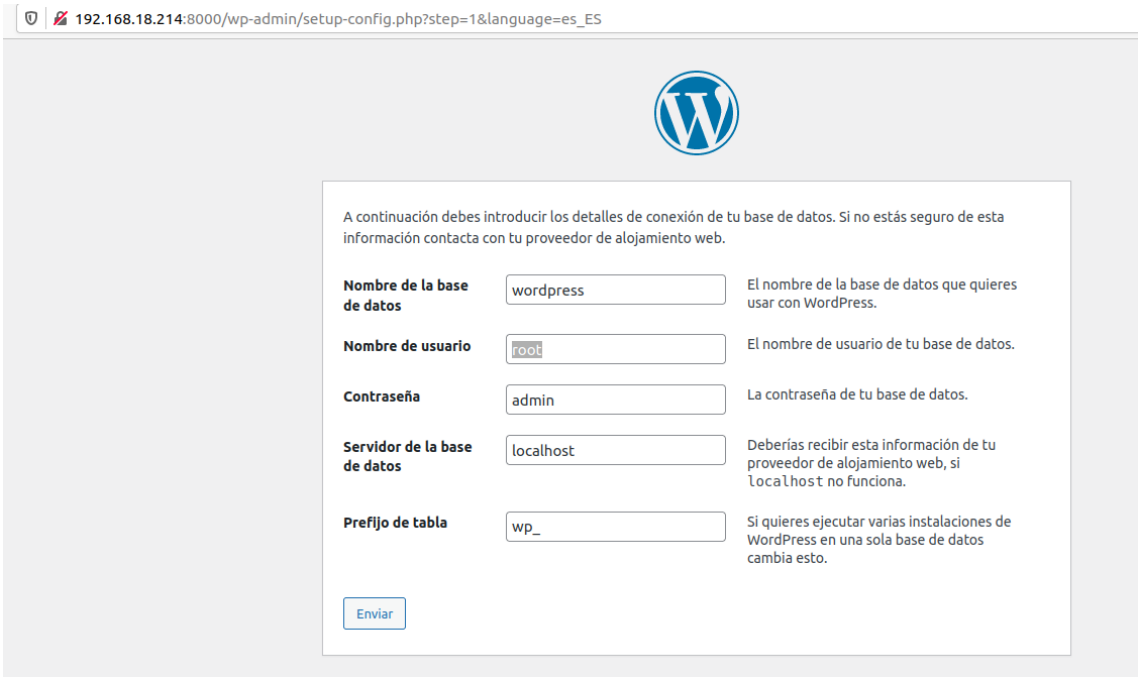
```
usuario@luna:~$ ls
data-mariadb  www-html
usuario@luna:~$ ls -l
total 8
drwxr-xr-x 2 usuario usuario 4096 abr  9 14:55 data-mariadb
drwxr-xr-x 3 usuario usuario 4096 abr  9 14:58 www-html
usuario@luna:~$ ls -l www-html/
total 4
drwxr-xr-x 5 100032 100032 4096 mar  9 20:19 wordpress
usuario@luna:~$ sudo chown -R usuario:usuario www-html
[sudo] password for usuario:
usuario@luna:~$ ls -l www-html/wordpress/
```

COPIAS DE SEGURIDAD :

Generar una copia de seguridad del pod con sus contenedores:

$ **podman generate kube -f miprueba.yml admin**

```
usuario@luna:~$ podman generate kube -f miprueba.yml admin
usuario@luna:~$
```



Guia de instalación :

https://www.linuxhowto.net/how-to-install-podman-on-ubuntu-20-04/

Anexo 1.3

Instancia	CPU virtual*	Créditos por hora de CPU	Memoria (GiB)	Almacenamiento	Rendimiento de red
t2.nano	1	3	0,5	Solo EBS	Bajo
t2.micro	1	6	1	Solo EBS	De bajo a moderado
t2.small	1	12	2	Solo EBS	De bajo a moderado
t2.medium	2	24	4	Solo EBS	De bajo a moderado
t2.large	2	36	8	Solo EBS	De bajo a moderado
t2.xlarge	4	54	16	Solo EBS	Moderado
t2.2xlarge	8	81	32	Solo EBS	Moderado

Glosario

Contenedores

Sistema virtual que utiliza el mismo kernel de la máquina donde se crea (máquina real)

Podman

Servicio utilizado para crear y gestionar contenedores.

Amazon educate

Herramienta de amazon que nos ofrece recursos informáticos (servidores, servicios, instancias, máquinas virtuales) para nuestras actividades y proyectos de curso.

YML

Lenguaje para realizar tareas automatizadas.

Ubuntu server

Sistema operativo Linux de código abierto, utilizado en nuestro servidor real.

Servidor

máquina que atiende las peticiones de los clientes las 24h del día

Hosting

Alojamiento web, utilizamos nuestro servidor para alojar un sitio web determinado.

XAMPP

Unión entre un sistema de base de datos MySQL, el servidor web Apache y los lenguajes de script PHP y Perl.

WordPress

Sistema de gestión de contenidos, enfocado a la creación de cualquier tipo de página web

Bibliografía

Se han utilizado con mucha frecuencia las siguiente páginas webs como soporte para cualquier incidencia y falta de conocimiento.

Contenedores

https://podman.io/
http://docs.podman.io/en/latest/Commands.html
https://github.com/containers/podman
https://github.com/containers/podman/issues
https://www.linuxhowto.net/how-to-install-podman-on-ubuntu-20-04/
https://www.youtube.com/watch?v=l6Bgd7Y8pgE

https://elpuig.xeill.net/Members/q2dg/aso-mp06/uf2

https://www.nginx.com/
https://docs.nginx.com/
http://jasonwilder.com/blog/2014/03/25/automated-nginx-reverse-proxy-for-docker/

Automatización

https://www.ansible.com/
https://www.ansible.com/integrations/containers/operators

https://elpuig.xeill.net/Members/q2dg/aso-mp06/uf2

https://www.reviversoft.com/es/file-extensions/yml
https://oracle-base.com/articles/linux/podman-generate-and-play-kubernetes-yaml-files

Página web

https://www.w3schools.com/
https://www.w3schools.com/css/
https://www.w3schools.com/w3css/w3css_templates.asp
https://www.w3schools.com/php/default.asp

https://www.w3schools.com/sql/default.asp
https://www.w3schools.com/mysql/default.asp
https://www.w3schools.com/html/html_tables.asp

https://www.php.net/
https://www.php.net/manual/es/function.shell-exec.php
https://www.configuroweb.com/registro-y-login-de-usuarios-con-php-y-mysql/
https://github.com/configuroweb/login
https://www.youtube.com/watch?v=JQsj6_w6MyU&t=216s
https://www.php.net/manual/es/function.mysql-connect.php

Amazon

https://aws.amazon.com/es/console/
https://aws.amazon.com/es/education/awseducate/
https://www.pragma.com.co/academia/lecciones/paso-a-paso-para-una-cuenta-aws-de-estudiante
https://aws.amazon.com/es/getting-started/hands-on/deploy-wordpress-with-amazon-rds/3/

Repositorio de proyecto

https://github.com/sbravor64/asixProyecto

*Gracias a **Oscar Torrente** - Profesor de Administración de Sistemas en Redes*
Sin él no hubiera sido posible

Pasos para iniciar sesión al servidor de aws

Dar permisos 400 al fichero .pem
$ **chmod 400 proyecto.pem**

Comando para conectar a la máquina (AWS) por ssh:
$ **ssh -i "proyecto.pem" ubuntu@ec2-52-45-219-23.compute-1.amazonaws.com**

Datos:

clave pem:
https://drive.google.com/file/d/11kfgB8lntebT3jCUDiLq0jt9ylbWV7lh/view?usp=sharing

IP Publica: **52.45.219.23**

Dominio web: http://proyecto.ccff.site/

Github de página web: https://github.com/sbravor64/asixProyecto