podman

CREACIÓN DE ESCENARIOS MULTICONTENEDOR PODMAN CON PODMAN-COMPOSE

ANTONIO CASTRO DÍAZ

Índice de contenido

1.- Objetivos que se quieren conseguir y se han conseguido.

El objetivo de este proyecto es explicar la creación de escenarios multicontenedor podman con podman-compose

Este proyecto se enfoca en la creación de escenarios multicontenedor mediante el uso de podman-compose. El objetivo principal es proporcionar una guía detallada para crear y administrar escenarios multicontenedor de manera fácil y eficiente.

Para alcanzar este objetivo, se utilizará podman-compose, una herramienta que permite utilizar archivos Compose para crear y administrar contenedores en un sistema que utiliza podman en lugar de Docker. También se utilizará podman, una herramienta de línea de comandos similar a Docker, diseñada para ser segura y compatible con sistemas sin un demonio.

El proyecto incluye una guía detallada que cubre desde la instalación de las herramientas necesarias hasta la creación y administración de escenarios multicontenedor.

2.- Fundamentos teóricos y conceptos.

2.1.- ¿Que es podman?

Podman es una herramienta de línea de comandos que permite crear, administrar y ejecutar contenedores sin un demonio de fondo. A diferencia de Docker, que utiliza un demonio para manejar los contenedores, podman utiliza el sistema operativo para manejar los contenedores, lo que lo hace más seguro y fácil de usar en entornos de producción. Esta desarrollado por RedHat. Es una herramienta open source para desarrollar, gestionar y ejecutar contenedores en los sistemas Linux.

Su arquitectura inclusiva y sin daemons la convierte en una opción más segura y accesible para la gestión de los contenedores. Además, las funciones y las herramientas que la acompañan, como Buildah y Skopeo, permiten que los desarrolladores personalicen los entornos de contenedores según sus necesidades.

Podman está enfocado a contenedores mínimos que únicamente ejecutan un servicio: un servidor web, una base de datos, un servidor de correo...

2.1.1.-Que son los pods

Un pod es un concepto introducido por Kubernetes que se utiliza para agrupar uno o varios contenedores relacionados en una sola unidad lógica.

Un pod es la unidad más pequeña que puede ser desplegada en Kubernetes y representa un entorno de ejecución para uno o varios contenedores que comparten el mismo espacio de red y almacenamiento. En general, los contenedores dentro de un pod suelen estar estrechamente relacionados entre sí y trabajan juntos para proporcionar una funcionalidad específica de la aplicación.

2.1.2.-Que son los contenedores:

Un contenedor es un entorno de ejecución aislado y ligero para una aplicación y sus dependencias, que se basa en la tecnología de virtualización a nivel de sistema operativo. Un contenedor proporciona una forma consistente y portátil de empaquetar, distribuir y ejecutar una aplicación, sin las limitaciones de un sistema operativo completo.

2.1.3.-Como funciona podman

Podman funciona de manera similar a docker pero sin un daemon central. Esto significa que los contenedores se ejecutan directamente en el sistema host y no requieren una instancia adicional de Docker en segundo plano.

Con Podman, podemos crear, ejecutar y administrar contenedores de manera sencilla y segura. Algunas de las características clave incluyen:

-Creación de imágenes de contenedores a partir de archivos de configuración.

-Ejecución de contenedores en segundo o primer plano

-Interacción con contenedores en ejecución mediante la ejecución de comandos y la transferencia de archivos

-Gestión de volúmenes y redes asociadas con los contenedores

-Monitorización y depuración de contenedores en ejecución

Podman es compatible con la mayoría de las características y recursos de Docker, por lo que es una buena opción si buscamos una alternativa sin daemon a Docker que sea fácil de usar y ofrezca una gran compatibilidad.

2.1.4- Otras funciones de podman:

Podman además ofrece otras funciones mas complejas más alla de crear pods como son:

-Activación y gestión de pods con varios contenedores con Podman:

Podman es una alternativa a Docker que proporciona la misma funcionalidad de manejo de contenedores pero sin necesidad de un daemon y con características adicionales como el soporte para pods y la compatibilidad con la herramienta de orquestación de contenedores Kubernetes. Podman permite importar definiciones de Kubernetes usando el comando play.

Si creamos un pod:

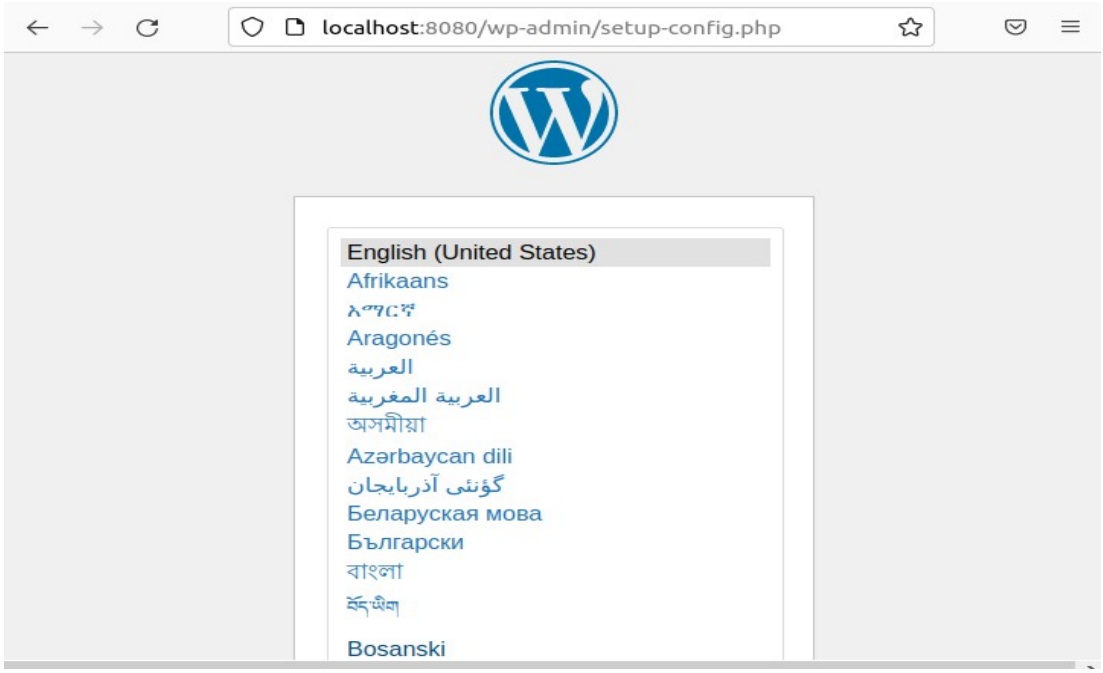
```
sudo podman pod create --name my-pod
-p 8080:80
```

Y Creamos los contenedores necesarios para desplegar wordpress:

```
sudo podman run \
-d --restart=always --pod=my-pod \
-e MYSQL_ROOT_PASSWORD="myrootpass" \
-e MYSQL_DATABASE="wp" \
-e MYSQL_USER="wordpress" \
-e MYSQL_PASSWORD="w0rdpr3ss" \
--name=wptest-db mariadb

sudo podman run \
-d --restart=always --pod=my-pod \
-e WORDPRESS_DB_NAME="wp" \
-e WORDPRESS_DB_USER="wordpress" \
-e WORDPRESS_DB_PASSWORD="w0rdpr3ss" \
-e WORDPRESS_DB_HOST="127.0.0.1" \
--name wptest-web wordpress
```

Y accedemos localhost:8080 podremos ver que efectivamente se ha creado



Ahora podemos generar un archivo .yaml en el que se guarden los contenedores que acabamos de crear, para ello ejecutaremos el siguiente comando:

sudo podman generate kube my-pod >> my-pod.yaml

Con este comando nos guardará las instrucciones para crear un podman con los contenedores necesarios para desplegar wordpress

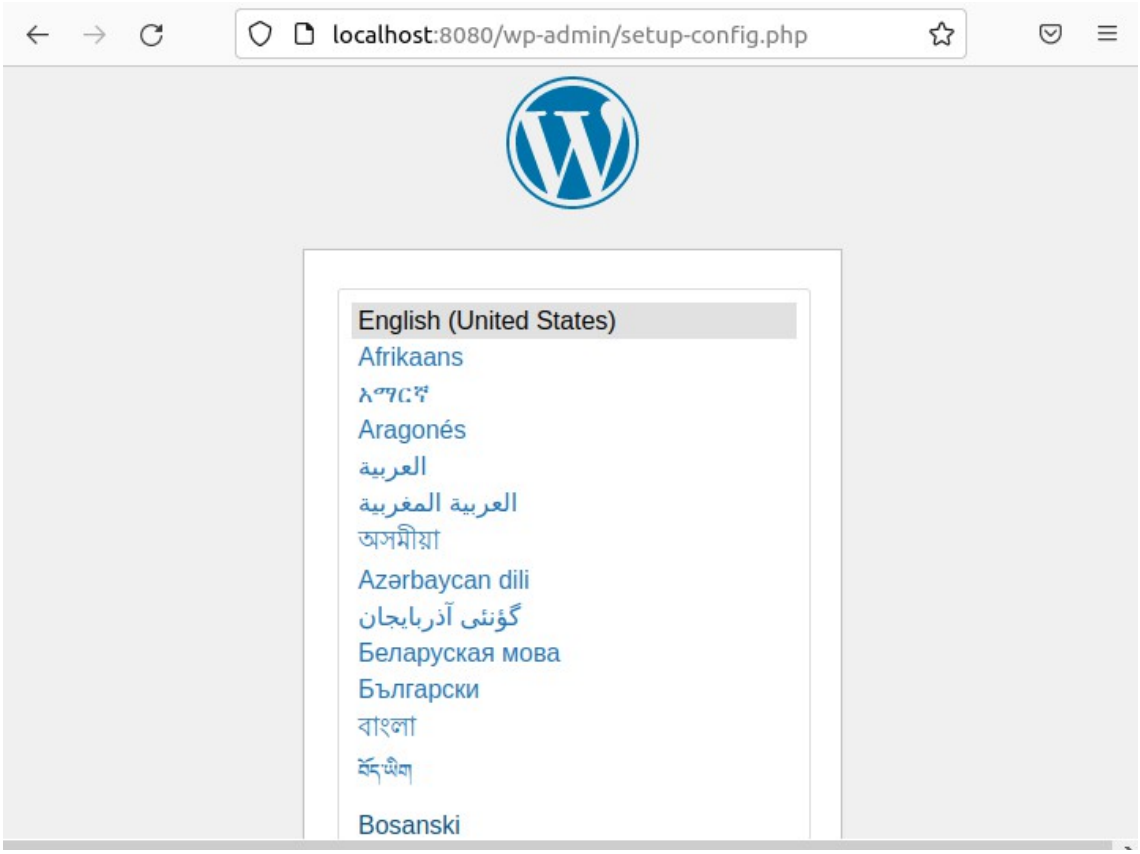
Borraremos el pod que hemos creado con este comando:

podman pod stop my-pod && podman pod rm my-pod

Después, para desplegar el archivo utilizaremos el siguiente comando:

sudo podman generate kube my-pod >> my-pod.yaml

Si ahora accedemos a **localhost:8080** podremos ver que efectivamente se ha creado



Cabe aclarar que esta es una función de podman que no tiene que ver con podman-compose. Aunque ambos utilicen un fichero .yaml para desplegar contenedores la sintaxis que utilizan ambos es diferente. Si intentamos desplegar este archivo con podman-compose nos mostrará el siguiente error:

```
root@usuario-VirtualBox:~/docker# podman-compose up podman-compose.yaml
podman-compose version: 1.0.4
['podman', '--version', '']
using podman version: 3.4.4
WARNING: No services defined
Traceback (most recent call last):
  File "/usr/local/bin/podman-compose", line 2862, in <module>
    main()
  File "/usr/local/bin/podman-compose", line 2858, in main
    podman_compose.run()
  File "/usr/local/bin/podman-compose", line 1385, in run
    cmd(self, args)
  File "/usr/local/bin/podman-compose", line 1712, in wrapped
    return func(*args, **kw)
  File "/usr/local/bin/podman-compose", line 1987, in compose_up
    excluded = get_excluded(compose, args)
  File "/usr/local/bin/podman-compose", line 1976, in get_excluded
    excluded -= compose.services[service]["_deps"]
KeyError: 'podman-compose.yaml'
root@usuario-VirtualBox:~/docker#
```

El cual nos indica que la sintaxis del archivo es incorrecta y no puede desplegar el escenarios

2.2 ¿Que es podman-compose?

Podman-compose es una herramienta que permite crear y administrar aplicaciones compuestas por múltiples contenedores utilizando un archivo de configuración en formato YAML, similar al formato utilizado por Docker Compose.

Con Podman-compose, podemos definir la configuración de los contenedores, las imágenes que deben utilizarse, las redes y volúmenes que deben conectarse, y mucho más, todo en un solo archivo.

Podman-compose permite simplificar el proceso de creación y gestión de aplicaciones compuestas por contenedores, al permitir describir toda la configuración en un único archivo y automatizar tareas como la creación, la ejecución y el mantenimiento de contenedores.

Con Podman-compose, podemos crear aplicaciones compuestas por contenedores de manera sencilla y eficiente, sin tener que escribir comandos complejos o manualmente configurar cada contenedor individualmente

2.2.1 ¿Como funciona podman-compose?

Podman-compose funciona leyendo un archivo de configuración en formato YAML que describe la aplicación compuesta por contenedores que se quiere crear y administrar.

Este archivo, que suele llamarse "podman-compose.yml", contiene información sobre las imágenes de contenedores que se deben utilizar, las configuraciones de red y volumen, los comandos que deben ejecutarse al iniciar los contenedores, etc..

Una vez que se tiene el archivo de configuración de Podman-compose, podemos ejecutar comandos para crear y administrar los contenedores de manera fácil y automatizada.

Algunos de los comandos más comunes incluyen:

-**podman-compose up**: Crea y ejecuta los contenedores descritos en el archivo de configuración.

-**podman-compose down**: Detiene y elimina los contenedores.

-**podman-compose ps**: Muestra el estado de los contenedores en ejecución.

-**podman-compose logs**: Muestra los registros de los contenedores en ejecución.

-**podman-compose exec**: Ejecuta un comando dentro de un contenedor en ejecución.

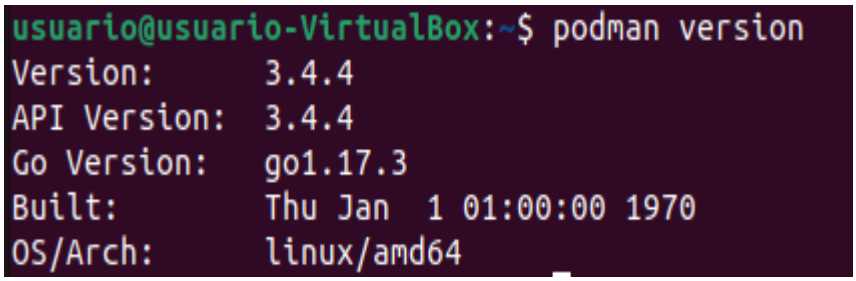
3.- Instalación y configuraciones

3.1.- Instalación de podman

Utilizare una máquina virtual con la ultima versión de ubuntu. Para su actualización introduciremos los siguientes comandos:

```
apt update && apt upgrade -y
apt install podman
```

Para comprobar que está instalado ejecutaremos el comando podman version, que nos mostrará lo siguiente:



Con esto ya tendríamos podman instalado y listo para usarse en debían 11.

3.2.- Instalación de podman-compose

Para instalar podman-compose debemos primero instala curl para descargar el script de instalación de podman-compose:

```
sudo apt-get install -y curl git
```

Después descargaremos e instalaremos el script de instalación de Podman Compose:

```
curl -o /usr/local/bin/podman-compose
https://raw.githubusercontent.com/containers/
podman-compose/devel/podman_compose.py
```

Asignamos permisos de ejecución al script:

```
sudo chmod +x /usr/local/bin/podman-compose
```

También tendremos que instalar pip:

```
apt install python3-pip
```

Además del paquete dotenv:

```
pip3 install python-dotenv
```

Tendremos que configurar podman para que busque en dockerhub, para ello en el fichero */etc/containers/registries.conf*

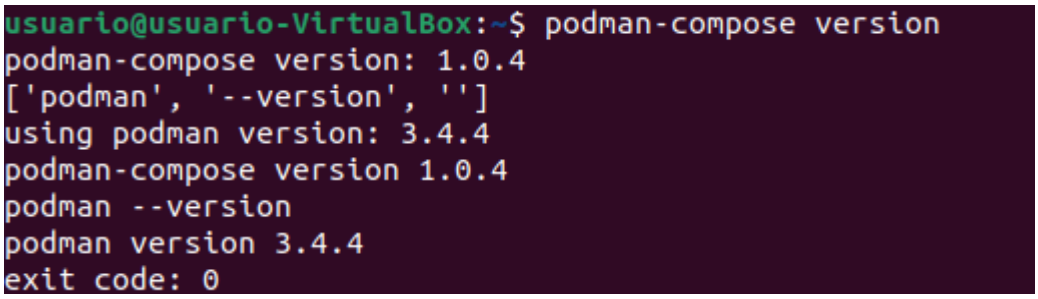
Para descomentar la siguiente linea

unqualified-search-registries = ["docker.io"]

Y por ultimo, verificaremos que la instalación ha sido correcta:

```
podman-compose version
```

Si todo se ha ejecutado correctamente, deberías ver la versión de Podman Compose que hemos instalado.



Una vez hecho esto, podemos comenzar a crear y gestionar aplicaciones de contenedores con Podman Compose.

4.-Descripción detalla de los que se ha realizado

Una vez instalado correctamente podman + podman-compose, crearemos varios escenarios multicontenedores:

4.1-Multicontenedor de base de datos + wordpress:

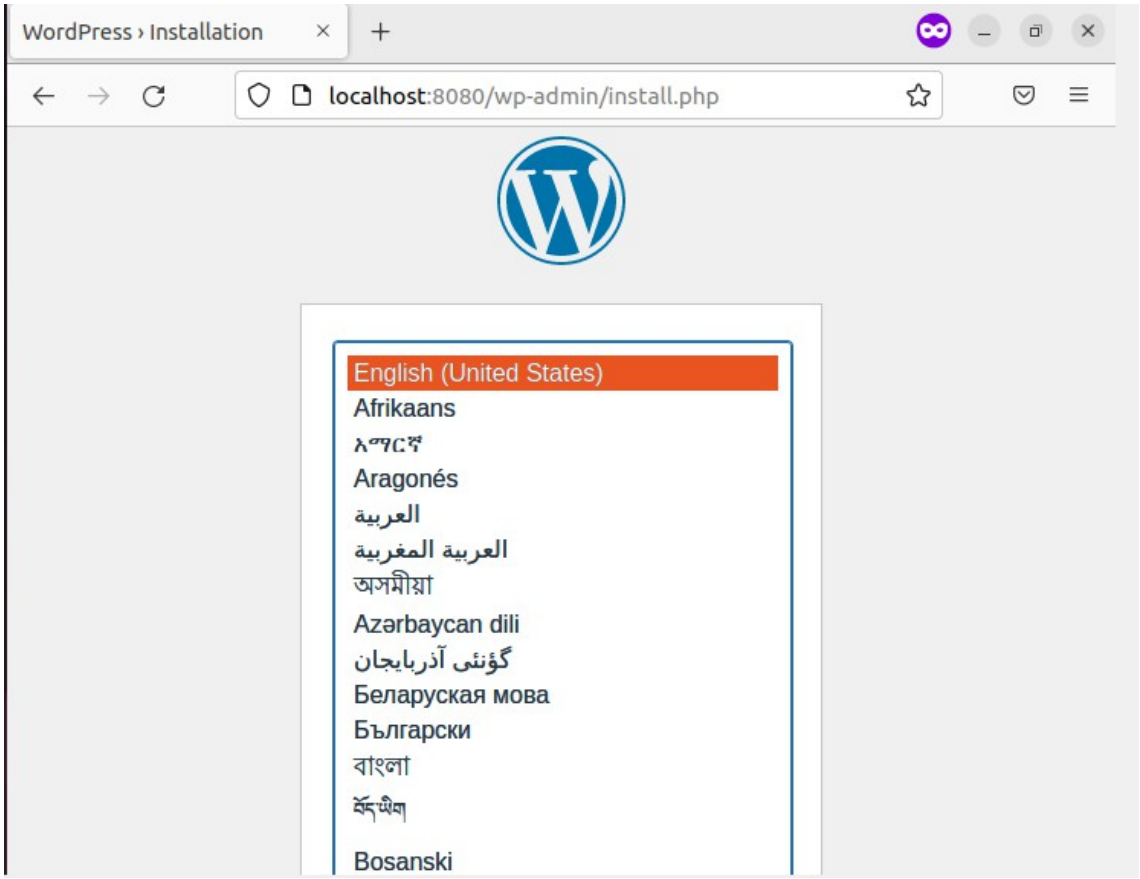
```
version: '3.1'
services:
  wordpress:
    container_name: servidor_wp
    image: wordpress
    restart: always
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: user_wp
      WORDPRESS_DB_PASSWORD: asdasd
      WORDPRESS_DB_NAME: bd_wp
    ports:
      - 8080:80
    volumes:
      - wordpress_data:/var/www/html/wp-
content
  db:
    container_name: servidor_mysql
    image: mariadb
    restart: always
    environment:
      MYSQL_DATABASE: bd_wp
      MYSQL_USER: user_wp
      MYSQL_PASSWORD: asdasd
      MYSQL_ROOT_PASSWORD: asdasd
    volumes:
      - mariadb_data:/var/lib/mysql
volumes:
```

```
wordpress_data:
mariadb_data:
```

En este archivo podman-compose crearemos dos pods, uno de ellos con wordpress y el otro con mariadb.

En este archivo especificamos el nombre del pod que se va a crear, la base de datos que se va a crear, las variables de entorno necesarias para la conexión de la base de datos, además del usuario y la contraseña y el puerto al que debemos conectarnos para acceder a la aplicación. Además, se crean dos volúmenes para almacenar los datos persistentes de Wordpress y MariaDB.

Una vez ejecutemos el podman-compose up y accedamos a http://localhost:8080 podremos ver que tenemos la aplicación lista para su instalación:



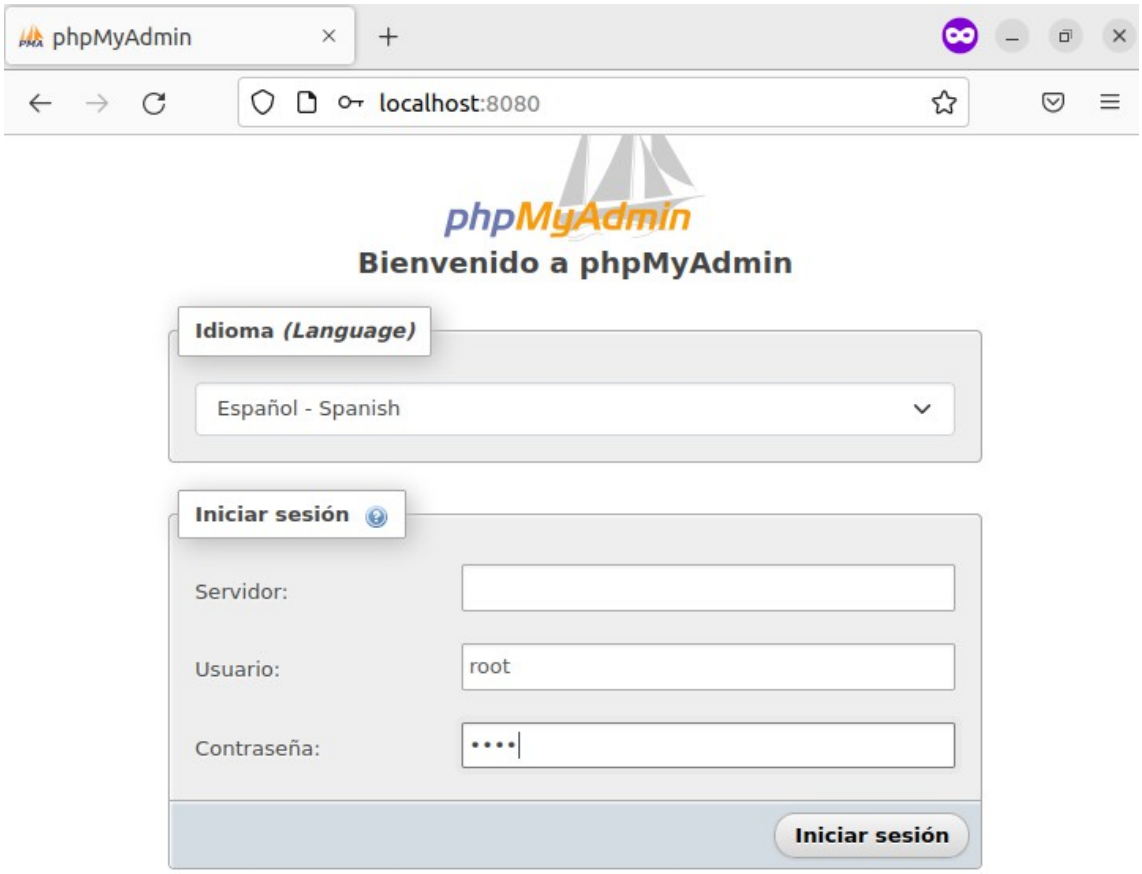
4.2-Multicontenedor de base de datos + phpmyadmin:

```
version: '3.7'
services:
  db:
    image: mariadb
    environment:
      MYSQL_ROOT_PASSWORD: password
      MYSQL_DATABASE: tiger
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    ports:
      - "8080:80"
    environment:
      PMA_HOST: db
      PMA_PORT: 3306
      PMA_ARBITRARY: 1
    depends_on:
      - db
```

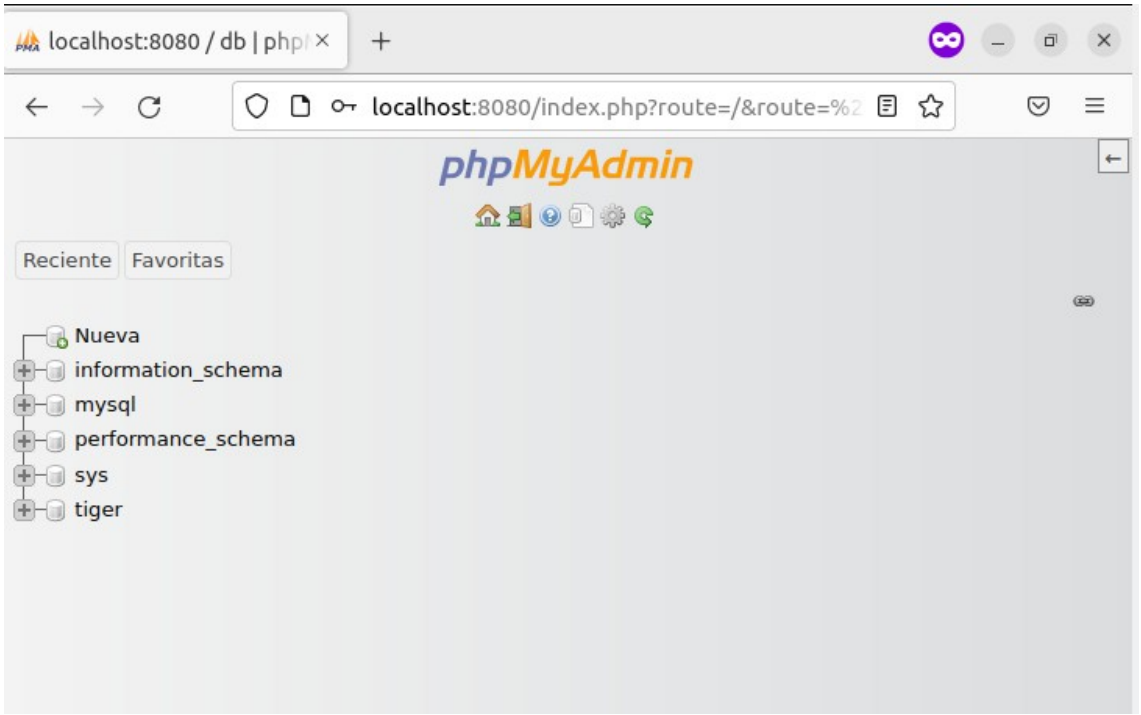
En este archivo podman-compose crearemos dos pods, uno de ellos con phpmyadmin y el otro con mariadb.

En este archivo especificamos el nombre del pod que se va a crear, la base de datos que se va a crear (tiger), las variables de entorno necesarias para la conexión de la base de datos, además del usuario y la contraseña de la base de datos y de phpmyadmin y el puerto al que debemos conectarnos para acceder a la aplicación. Además, se crean dos volúmenes para almacenar los datos persistentes de phpmyadmin y MariaDB.

Una vez ejecutemos el podman-compose up y accedamos a http://localhost:8080 podremos ver que tenemos la aplicación lista para acceder con el usuario root y la contraseña que especificamos en el archivo:



Y vemos que podemos acceder sin problemas y ver que tenemos la tabla tiger ya creada



4.3-Multicontenedor de base de datos postgres + grafana :

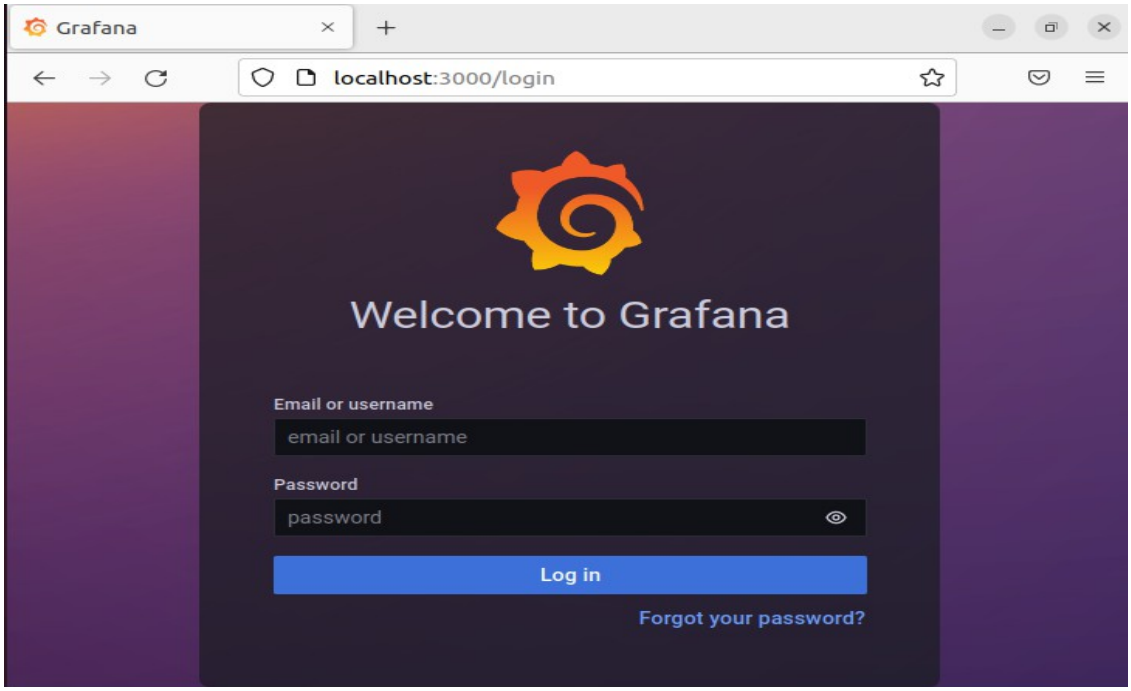
```
version: '3'
services:
  grafana:
    image: grafana/grafana
    ports:
      - "3000:3000"
    environment:
      GF_SECURITY_ADMIN_PASSWORD: "secret"
      GF_INSTALL_PLUGINS: grafana-clock-
panel,grafana-piechart-panel,grafana-simple-
json-datasource

    volumes:
      - grafana-data:/var/lib/grafana
  db:
    image: postgres
    environment:
      POSTGRES_DB: "grafana"
      POSTGRES_USER: "grafana"
      POSTGRES_PASSWORD: "secret"
    volumes:
      - db-data:/var/lib/postgresql/data
volumes:
  grafana-data:
  db-data:
```

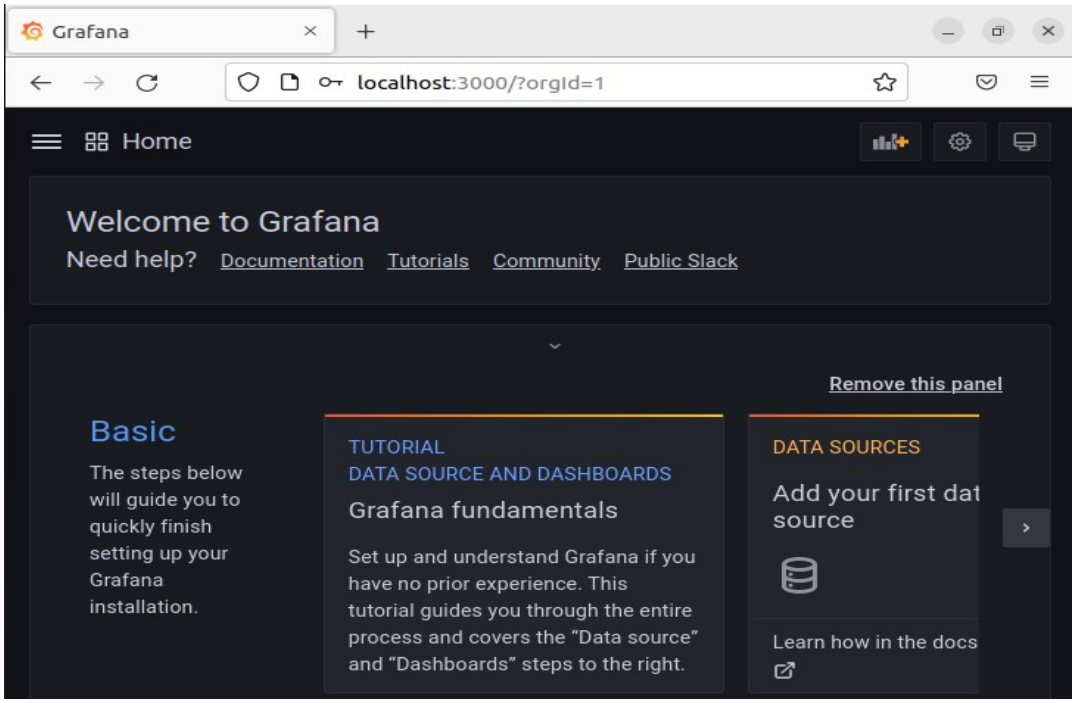
En este archivo podman-compose crearemos dos pods, uno de ellos con grafana y el otro con postgres.

En este archivo especificamos el nombre del pod que se va a crear, la base de datos que se va a crear , las variables de entorno necesarias para la conexión de la base de datos, además del usuario y la contraseña de la base de datos y de grafana y el puerto al que debemos conectarnos para acceder a la aplicación. Además, se crean dos volúmenes para almacenar los datos persistentes de grafana y MariaDB. En diferencia a los dos escenarios anteriores en este podemos añadir los plugins que queremos que grafana tenga instalado especificando en la variable de entorno “ GF_INSTALL_PLUGINS”.

Una vez ejecutemos el podman-compose up y accedamos a http://localhost:3000 podremos ver que tenemos la aplicación lista para acceder con el usuario admin y la contraseña que especificamos en el archivo, en este caso secret:



Ponemos la credenciales y entraremos sin ningún problema



5.- Conclusiones y propuestas para seguir trabajando sobre el tema

Es importante seguir investigando y aprendiendo sobre podman y podman-compose para estar al tanto de las últimas características y mejoras que vayan añadiendo.

Después de haber tratado tanto con podman y podman-compose podemos ver que es una herramienta muy potente que permite la fácil y rápida creación de distintos escenarios sin apenas utilizar recursos.

Mis propuestas para seguir trabajando son:

-Automatización: Implementación de scripts para el despliegue y gestión de escenarios multicontenedor

-Integración con otras herramientas: Se pueden integrar otras herramientas, como Git o Jenkins, para mejorar la gestión de versiones y la automatización de despliegue.

Además hay varios escenarios multicontenedores que he tenido que descartar en los que podría seguir trabajando como son:

-Escenario multicontenedor que instala una base de datos y el cms moodle
-Escenario multicontenedor que instala servidor de correos con una plataforma donde pueda enviar y recibir correos (roundcube)

En conclusión, el proyecto de creación de escenarios multicontenedor con podman y podman-compose es una excelente oportunidad para aprender sobre los conceptos y técnicas de virtualización de contenedores y para mejorar habilidades en el desarrollo de aplicaciones y servicios. Además, es un proyecto con una amplia gama de aplicaciones en diversos sectores y una gran demanda en la industria.

6.- Dificultades que se han encontrado

En un principio, la idea era instalar podman en debian. Utilice una máquina debian que tenia ya creada anteriormente y al instalar el paquete podman me devolvía el siguiente error:

"Error: kernel does not support overlay fs: 'overlay' is not supported over extfs at "/var/lib/containers/storage/overlay": backing file system is unsupported for this graph driver"

Esto se debe porque el sistema operativo y el kernel que estás usando no tienen soporte para el sistema de archivos overlay, que es una característica esencial de Podman. Podman utiliza un sistema de archivos overlay para crear contenedores aislados con sus propios sistemas de archivos. Si el kernel no tiene soporte para overlay fs, no es posible usar Podman.

Viendo que no funcionaba, pase a crear una máquina vagrant con debian y ahí si pude instalar podman pero al descargar el script de podman-compose y darle el permiso de ejecución me muestra el siguiente error cuando ejecuto el comando: podman-compose version

```
Traceback (most recent call last):
  File "/usr/local/bin/podman-compose", line 39, in
<module>
    from dotenv import dotenv_values
ModuleNotFoundError: No module named 'dotenv'
```

Este error lo soluciono ejecutando el siguiente comando:

```
pip3 install python-dotenv
```

Por ultimo, una vez que ya me funcionaba podman y podman-compose, lancé el siguiente archivo:

```
version: '3.7'

services:
  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: example
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
    volumes:
      - db_data:/var/lib/mysql

  wordpress:
    image: wordpress:latest
    ports:
      - "8080:80"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    depends_on:
      - db
    volumes:
      - wordpress_data:/var/www/html

volumes:
  db_data:
  wordpress_data:
```

Pero al lanzarlo me lanza el siguiente error:

Error: the network name vagrant_default is already used
subprocess.CalledProcessError: Command '['podman', 'network', 'create', '--label', 'io.podman.compose.project=vagrant', '--label', 'com.docker.compose.project=vagrant', 'vagrant_default']' returned non-zero exit status 125.

El error indica que hay un pod con el nombre de pod_vagrant y una red con el nombre de vagrant_default que ya están creados y crea conflicto, pero antes de lanzar el podman-compose up no había nada creado.

Lanzo el siguiente comando para borrar la red y el pod y que automáticamente se inicie el podman-compose:

```
podman pod rm -f pod_vagrant && podman network rm vagrant_default && podman-compose up
```

El problema es que la versión disponible de podman en debian es muy antigua, este problema lo solucionamos al usar la ultima versión de ubuntu donde si es compatible podman y podman-compose