

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS,
ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES



COMUNICACIONES AVANZADAS

Octavo 'A'

CONSULTA

Tema:

DOCKER Y DOCKER COMPOSE EN LA CREACIÓN DE
CONTENEDORES CON WORDPRESS, MARIADB Y
RESPALDOS

Integrantes:

Yugcha Edison

Fecha de Envío:

Miercoles 25 de Octubre 2023

Fecha de Entrega:

Jueves 26 de Octubre de 2023

Docente: Ing. Mg. Santiago Manzano

Septiembre 2023 - Febrero 2024

AMBATO-ECUADOR

2023

Índice

1. TEMA	3
2. OBJETIVOS	3
2.1. Objetivo General	3
2.2. Objetivos Específicos	3
3. Resumen	3
4. FUNDAMENTACIÓN TEÓRICA	3
4.1. DOCKER	3
4.2. DOCKER COMPOSE	3
4.3. MARIADB	4
4.4. WORDPRESS	4
4.5. BACKUP	5
4.6. INSTALACION DE CONTENEDORES	6
4.7. Desarrollo	6
5. CONCLUSIONES	17
6. BIBLIOGRAFÍA	17
7. ANEXOS	18

Índice de figuras

1. Docker Compose	4
2. MariaDB	4
3. Wordpress	5
4. Backup	6
5. máquina virtual mediante PUTTY	6
6. Descargamos la imagen de MariaDB	6
7. Verificamos la imagen descargada de MariaDb	7
8. Instalamos wodrpess	7
9. Verificamos que se haya instalado	7
10. Creamos los contenedores dentro de la misma red	7
11. Creamos el contenedor de maría db	8
12. Creamos el contenedor de servidor _{ey}	8
13. Verificamos los contenedores creados	8
14. Apagamos la ejecución de los contenedores por el momento	8
15. INSTALAMSO DOCKER COMPOSE	9
16. Le damos permisos	9
17. Verificamos la version	9
18. Creamos un volumen para maria db y backup-data	9
19. Creamos un directorio	9
20. Ingresamos a este directorio	9
21. Creamos un archivo yml	10
22. Ingresamos y ponemos la siguiente configuración	10
23. Ingresamos y ponemos la siguiente configuración	11
24. creamos un archivo DockerFile	11
25. crear respaldos cada 5 minutos	12
26. Creamos un archivo bin/sh	12
27. Ejecutamos los contenedores	13
28. Ejecutamos los contenedores	13
29. los contenedores se esten ejecutando correctamente	14
30. verificar el respald cada 5 minutos ingresamos	15
31. pagina de wordpress	16
32. pagina de wordpress	17

33. ANEXO 1 18

34. ANEXO 2 19

35. ANEXO 3 20

36. ANEXO 4 21

1. TEMA

Docker y Docker Compose en la creación de contenedores con WordPress, MariaDB y respaldos

2. OBJETIVOS

2.1. Objetivo General

- Utilizar Docker y Docker Compose para crear un entorno de desarrollo con WordPress, MariaDB y respaldos.

2.2. Objetivos Específicos

- Conocer los conceptos básicos de Docker y Docker Compose.
- Crear un contenedor con WordPress que cumpla con su funcionalidad.
- Desarrollar un contenedor de MariaDB junto con un contenedor de respaldos que realice los mismos cada 5 minutos .

3. Resumen

Docker es una plataforma de contenedores que te permite empaquetar y distribuir aplicaciones en entornos aislados llamados contenedores. Proporciona una forma eficiente y consistente de ejecutar aplicaciones en diferentes sistemas operativos. Por otro lado, Docker Compose es una herramienta que complementa a Docker, permitiéndote definir y orquestar múltiples contenedores para crear un entorno de desarrollo completo.

Mientras que Docker se centra en la creación y ejecución de contenedores individuales, Docker Compose se encarga de coordinar y gestionar múltiples contenedores que trabajan juntos para formar una aplicación. Docker Compose es especialmente útil cuando necesitas configurar un entorno de desarrollo que involucre múltiples servicios o contenedores. Te permite definir la estructura de tu aplicación y sus dependencias en un solo archivo YAML, simplificando la creación y gestión de tu entorno de desarrollo.

Al utilizar Docker Compose en lugar de Docker solo, obtienes beneficios como la capacidad de definir volúmenes y redes personalizadas, la posibilidad de escalar servicios y la facilidad de compartir y replicar tu entorno de desarrollo con otros miembros del equipo. Docker Compose ofrece una solución más completa y eficiente para configurar y orquestar aplicaciones multi-contenedor en tu entorno de desarrollo.

4. FUNDAMENTACIÓN TEÓRICA

4.1. DOCKER

Docker es una plataforma de software que le permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución. Con Docker, puede implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno con la certeza de saber que su código se ejecutará. La ejecución de Docker en AWS les ofrece a desarrolladores y administradores una manera muy confiable y económica de crear, enviar y ejecutar aplicaciones distribuidas en cualquier escala.

4.2. DOCKER COMPOSE

Docker Compose es una herramienta de la plataforma dedicada a la orquestación local de dockers, es decir, se utiliza con el objetivo de definir y ejecutar aplicaciones Docker de varios contenedores de forma fácil y rápida.

Esta definición y orquestación se lleva a cabo de forma local al interior de los containers, quienes, además, se encontrarán unidos a través de una red de Docker.

Para su funcionamiento, Docker Compose emplea un archivo tipo YAML que le permite realizar la configuración de los diferentes servicios pertenecientes a la aplicación. Después de esto, un comando cumplirá la función de crear e iniciar estos servicios a partir de sus ajustes.

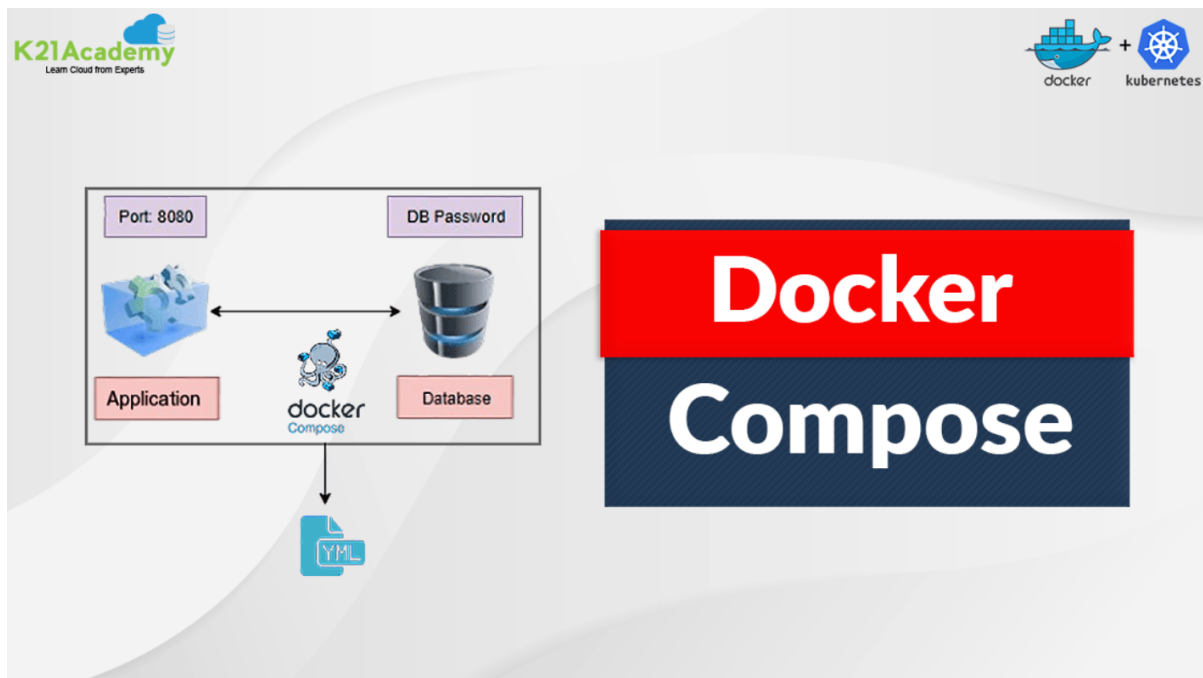


Figura 1: Docker Compose

4.3. MARIADB

MariaDB es un sistema de gestión de bases de datos relacionales (RDBMS) gratuito y de código abierto. Fue creado por los desarrolladores originales de MySQL por la preocupación de que MySQL pasara a ser comercializado después de que Oracle lo adquiriera en 2009.

MariaDB está escrito en C y C++ y es compatible con varios lenguajes de programación, incluidos C, C, Java, Python, PHP y Perl. MariaDB también es compatible con todos los principales sistemas operativos, incluidos Windows, Linux y macOS.

Aunque es una base de datos relacional, MariaDB ofrece funciones similares a las de NoSQL en la versión 10. El motor Connect permite acceder fácilmente a datos no estructurados desde MariaDB, mientras que las columnas dinámicas permiten el almacenamiento de tipo NoSQL de diferentes tipos de objetos en la misma fila.



Figura 2: MariaDB

4.4. WORDPRESS

WordPress es un sistema de gestión de contenidos web (CMS o content management system), que en pocas palabras es un sistema para publicar contenido en la web de forma sencilla. Tan común es ya, que es el líder absoluto a nivel mundial para la creación de webs desde hace muchísimos años.

Es un software de código abierto (se puede tener acceso a todo el código) que además podemos tratar de mejorar dentro de su comunidad. Por lo tanto WordPress es gratuito y podemos descargarlo desde sus servidores <https://wordpress.org/> para distintos fines.

WordPress llegó para democratizar la web, como otros CMS. Desde el año 2003, es un sistema de gestión de contenidos que hace que la creación de contenido web no dependa sólo de programadores y de personas de alto conocimiento técnico. Ahora, cualquier persona puede crear una web.



Figura 3: Wordpress

4.5. BACKUP

Aunque existen diferentes métodos para realizar copias de seguridad de bases de datos MySQL o MariaDB, el más común y eficiente se basa en el uso de una herramienta nativa que tanto MySQL como MariaDB ponen a nuestra disposición para este cometido: el comando `mysqldump`. Como su propio nombre indica, se trata de un programa ejecutable desde la línea de comandos que permite realizar una exportación completa (dump) de todo el contenido de una base de datos o incluso de todas las bases de datos presentes en una instancia de MySQL o MariaDB en ejecución. Por supuesto también permite realizar copias de seguridad parciales, es decir, sólo de algunas tablas concretas.

El comando `mysqldump` ofrece multitud de parámetros distintos que lo hacen muy potente y flexible. Dado que disponer de tal cantidad de opciones puede llegar a ser confuso, en este post voy a recoger varios de los ejemplos de uso más frecuente usando los parámetros más habituales y que resultan de más utilidad en el día a día del administrador de sistemas.



Figura 4: Backup

4.6. INSTALACION DE CONTENEDORES

4.7. Desarrollo

Proceso

1. Ingresamos a nuestra máquina virtual mediante PUTTY

```
root@localhost:~  
login as: prueba  
prueba@192.168.149.131's password:  
Last login: Tue Oct 24 10:53:45 2023 from 192.168.149.1  
[prueba@localhost ~]$ su -  
Password:  
Last login: Tue Oct 24 13:19:22 -05 2023 on tty1  
[root@localhost ~]#
```

Figura 5: máquina virtual mediante PUTTY

2. Descargamos la imagen de MariaDB

```
root@localhost ~]# docker pull mariadb  
Using default tag: latest  
latest: Pulling from library/mariadb  
Digest: sha256:2403cc521634162f743b5179ff5b35520daf72df5d9e7e397192af685d9148fd  
Status: Image is up to date for mariadb:latest  
docker.io/library/mariadb:latest
```

Figura 6: Descargamos la imagen de MariaDB

3. Verificamos la imagen descargada de MariaDb

```
[root@localhost ~]# docker images
REPOSITORY      TAG          IMAGE ID       CREATED        SIZE
apache/ubuntu    v1           d8006a701511   5 days ago    231MB
git/ubuntu       v1           f046cdeb5306   7 days ago    198MB
wordpress        latest       bd918e5d2324   10 days ago    666MB
mariadb          latest       f35870862d64   11 days ago    404MB
ubuntu           latest       e4c58958181a   2 weeks ago    77.8MB
alpine           latest       8ca4688f4f35   3 weeks ago    7.34MB
[root@localhost ~]#
```

Figura 7: Verificamos la imagen descargada de MariaDb

4. Instalamos wordpress

```
[root@localhost ~]# docker pull wordpress
Using default tag: latest
latest: Pulling from library/wordpress
Digest: sha256:549987ef8d878e6a9f360ee0cc033f03ca68c1e86aa9af7a206bdae4a98b0486
Status: Image is up to date for wordpress:latest
docker.io/library/wordpress:latest
[root@localhost ~]#
```

Figura 8: Instalamos wordpress

5. Verificamos que se haya instalado

```
[root@localhost ~]# docker images
REPOSITORY      TAG          IMAGE ID       CREATED        SIZE
apache/ubuntu    v1           d8006a701511   5 days ago    231MB
git/ubuntu       v1           f046cdeb5306   7 days ago    198MB
wordpress        latest       bd918e5d2324   10 days ago    666MB
mariadb          latest       f35870862d64   11 days ago    404MB
ubuntu           latest       e4c58958181a   2 weeks ago    77.8MB
alpine           latest       8ca4688f4f35   3 weeks ago    7.34MB
[root@localhost ~]#
```

Figura 9: Verificamos que se haya instalado

6. Creamos los contenedores dentro de la misma red

```
[root@localhost ~]# docker network create red_ey
ffcf2d3484be53d83f49e8b623037bb9ae1fe8df2a42f420c1ceb0823c3c33b1
[root@localhost ~]#
```

Figura 10: Creamos los contenedores dentro de la misma red

7. Creamos el contenedor de maria db


```
[root@localhost ~]# docker run -d --name servidor_mariadb \
> --network red_ey \
-v /opt/mysql_ey:/var/lib/mysql \
-e MYSQL_DATABASE=bd_ey \
-e MYSQL_USER=user_ey \
-e MYSQL_PASSWORD=asdasd \
-e MYSQL_ROOT_PASSWORD=asdasd \
mariadb
54852c1943fc8aef239a841b3e49973ad12d4914fe0cce2feedc4261f7e2dcf1
```

Figura 11: Creamos el contenedor de maría db

8. Creamos el contenedor de servidor ey

```
[root@localhost ~]# docker run -d --name servidor_ey \
> --network red_ey \
-v /opt/wordpress:/var/www/html/ey-content \
-e WORDPRESS_DB_HOST=servidor_mariadb \
-e WORDPRESS_DB_USER=user_ey \
-e WORDPRESS_DB_PASSWORD=asdasd \
-e WORDPRESS_DB_NAME=bd_ey \
-p 80:80 \
wordpress
8c1ede79a7f111ad408ca5d5f2a42fbc9023d543d744b3af939185d53e6fb6d2
```

Figura 12: Creamos el contenedor de servidor_{ey}

9. Verificamos los contenedores creados

```
[root@localhost ~]# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
8c1ede79a7f1	wordpress	"docker-entrypoint.s..."	3 minutes ago	Up 3 minutes	0.0.0.0:80->80/tcp, :::80->80/tcp
54852c1943fc	mariadb	"docker-entrypoint.s..."	5 minutes ago	Up 5 minutes	3306/tcp

```
servidor_ey
servidor_mariadb
[root@localhost ~]#
```

Figura 13: Verificamos los contenedores creados

10. Apagamos la ejecución de los contenedores por el momento

```
docker stop servidor_ey
docker stop servidor_mariadb
```

```
[root@localhost ~]# docker stop servidor_ey
servidor_ey
[root@localhost ~]# docker stop servidor_mariadb
servidor_mariadb
[root@localhost ~]#
```

Figura 14: Apagamos la ejecución de los contenedores por el momento

11. INSTALAMOS DOCKER COMPOSE

```
[root@localhost ~]# curl -L "https://github.com/docker/compose/releases/download/v2.12.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/bin/docker-compose
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             %             Dload  Upload  Total   Spent    Left   Speed
  0     0    0     0    0     0      0     0  --:--:--  --:--:--  --:--:--    0
34 42.8M  34 14.9M    0     0 2153k      0  0:00:20  0:00:07  0:00:13 2633k
```

Figura 15: INSTALAMOS DOCKER COMPOSE

12. Le damos permisos

```
[root@localhost ~]# chmod +x /usr/bin/docker-compose
[root@localhost ~]#
```

Figura 16: Le damos permisos

13. Verificamos la version

```
[root@localhost ~]# docker-compose --version
Docker Compose version v2.23.0
[root@localhost ~]#
```

Figura 17: Verificamos la version

14. Creamos un volumen para maria db y backup-data

```
[root@localhost ~]# docker volume create mariadb-data
mariadb-data
[root@localhost ~]# docker volume create backup-data
backup-data
```

Figura 18: Creamos un volumen para maria db y backup-data

15. Creamos un directorio

```
[root@localhost ~]# mkdir practicadocker1
[root@localhost ~]# ls
anaconda-ks.cfg  docker-wordpress  ifcfg-ens256  practicadocker1  scripts
```

Figura 19: Creamos un directorio

16. Ingresamos a este directorio

```
[root@localhost practicadocker1]# nano docker-compose.yml
[root@localhost practicadocker1]#
```

Figura 20: Ingresamos a este directorio

17. Creamos un archivo yml

```
[root@localhost practicaDocker1]# nano docker-compose.yml
[root@localhost practicaDocker1]#
```

Figura 21: Creamos un archivo yml

18. Ingresamos y ponemos la siguiente configuración

```
services:
  mariadb-container:
    image: mariadb
    environment:
      MYSQL_DATABASE: bd_ey
      MYSQL_USER: user_ey
      MYSQL_PASSWORD: asdasd
      MYSQL_ROOT_PASSWORD: asdasd

    volumes:
      - mariadb-data:/var/lib/mysql

  wordpress-container:
    image: wordpress
    environment:
      WORDPRESS_DB_HOST: mariadb-container
      WORDPRESS_DB_USER: user_ey
      WORDPRESS_DB_PASSWORD: asdasd
      WORDPRESS_DB_NAME: bd_ey

[ Read 37 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Figura 22: Ingresamos y ponemos la siguiente configuración

```

version: '3'
services:
  mariadb-container:
    image: mariadb
    environment:
      MYSQL_DATABASE: bd_ey
      MYSQL_USER: user_ey
      MYSQL_PASSWORD: asdasd
      MYSQL_ROOT_PASSWORD: asdasd

  volumes:
    - mariadb-data:/var/lib/mysql

  wordpress-container:
    image: wordpress
    environment:
      WORDPRESS_DB_HOST: mariadb-container
      WORDPRESS_DB_USER: user_ey
      WORDPRESS_DB_PASSWORD: asdasd
      WORDPRESS_DB_NAME: bd_ey
    ports:
      - "80:80"
    depends_on:
      - mariadb-container

  backup-container:
    build:
      context: .
      dockerfile: Dockerfile-backup
    volumes:
      - backup-data:/backups
    depends_on:
      - mariadb-container

volumes:
  mariadb-data:
  backup-data:

```

Figura 23: Ingresamos y ponemos la siguiente configuración

19. Luego creamos un archivo DockerFile nano Dockerfile-backup

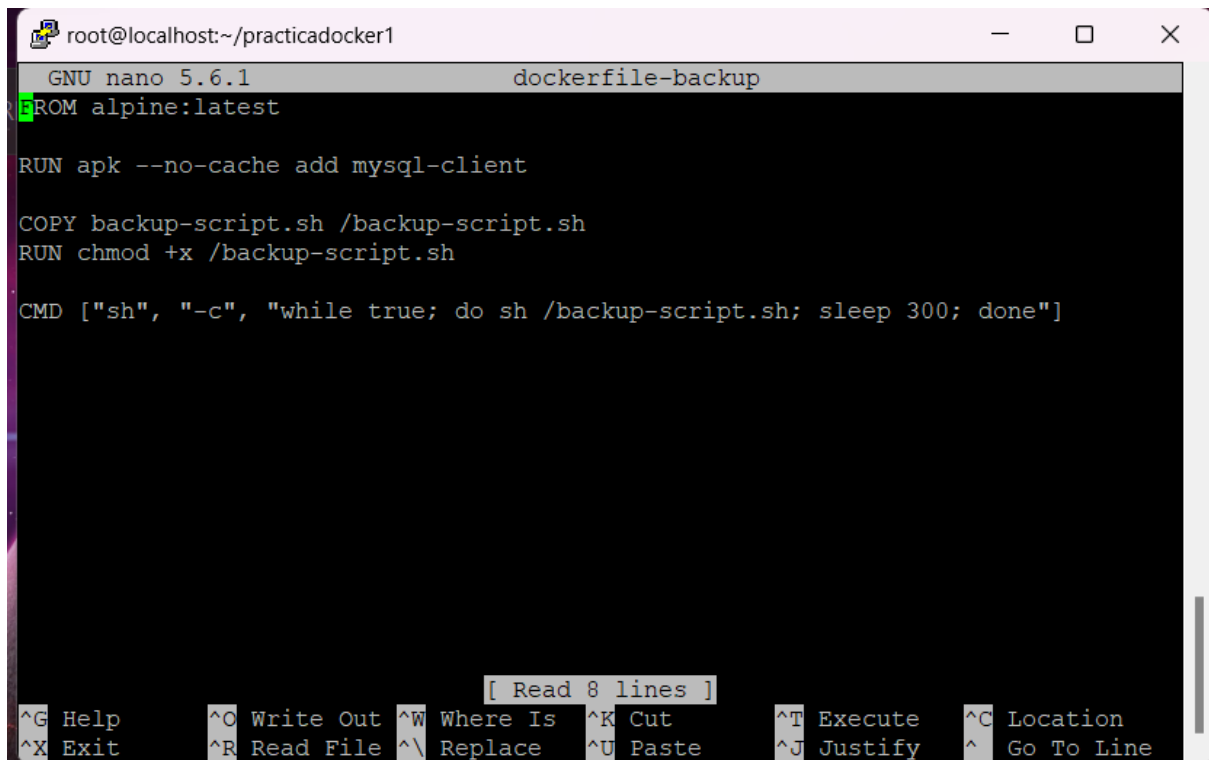
```

[root@localhost practicaDocker1]# nano dockerfile-backup
[root@localhost practicaDocker1]#

```

Figura 24: creamos un archivo DockerFile

20. Ingresamos el siguiente código que nos permite crear respaldos cada 5 minutos



```
root@localhost:~/practicadocker1
GNU nano 5.6.1 dockerfile-backup
FROM alpine:latest

RUN apk --no-cache add mysql-client

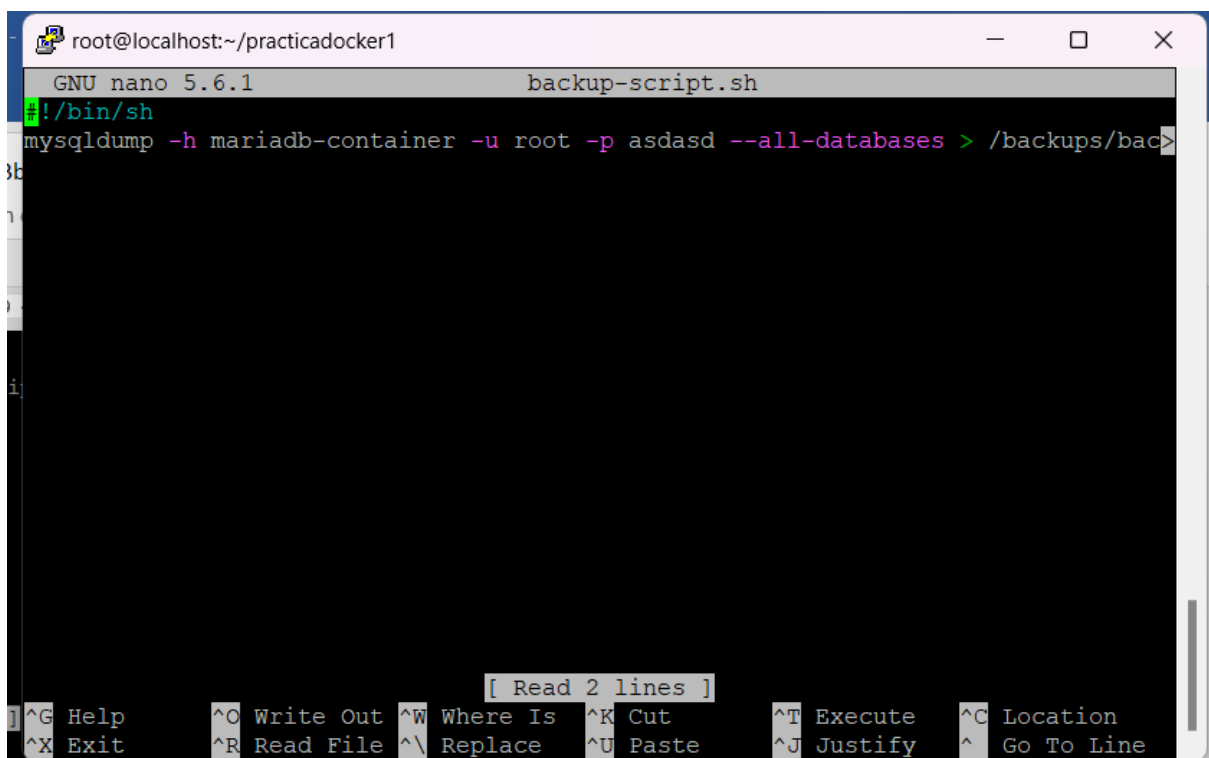
COPY backup-script.sh /backup-script.sh
RUN chmod +x /backup-script.sh

CMD ["sh", "-c", "while true; do sh /backup-script.sh; sleep 300; done"]

[ Read 8 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Figura 25: crear respaldos cada 5 minutos

21. Creamos un archivo bin/sh

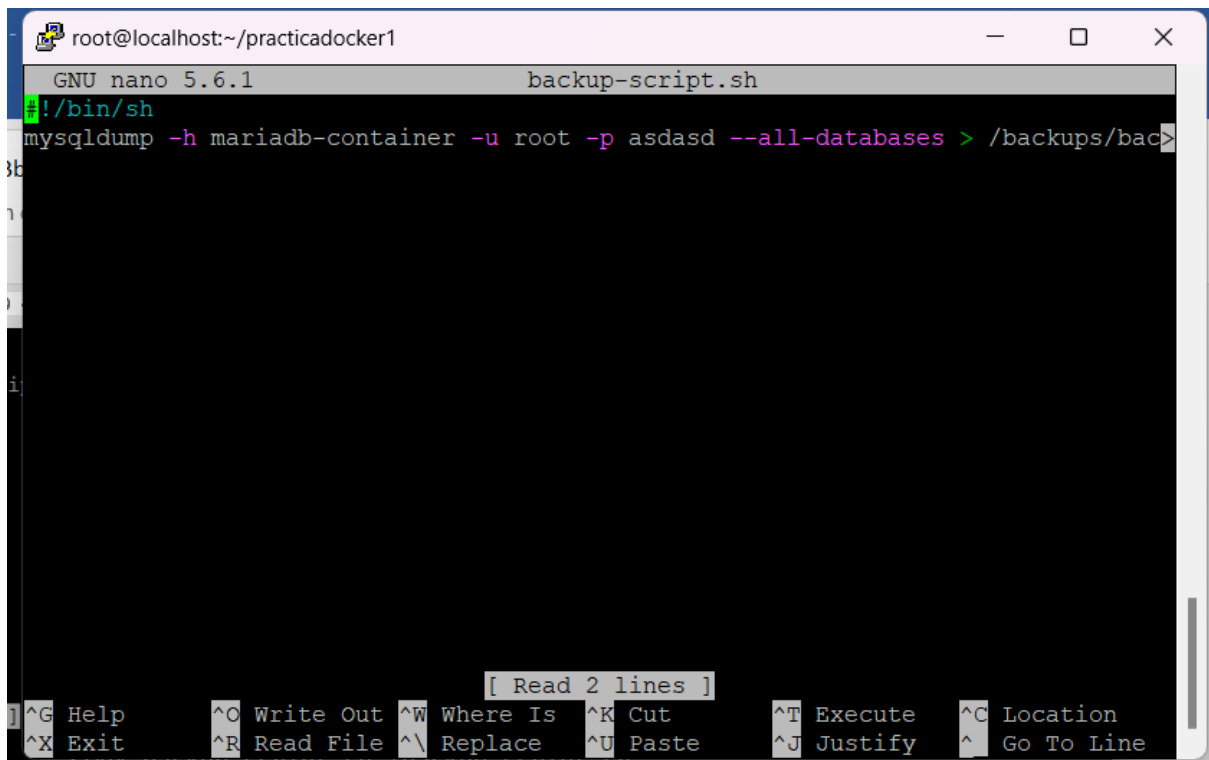


```
root@localhost:~/practicadocker1
GNU nano 5.6.1 backup-script.sh
#!/bin/sh
mysqldump -h mariadb-container -u root -p asdasd --all-databases > /backups/bac>

[ Read 2 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

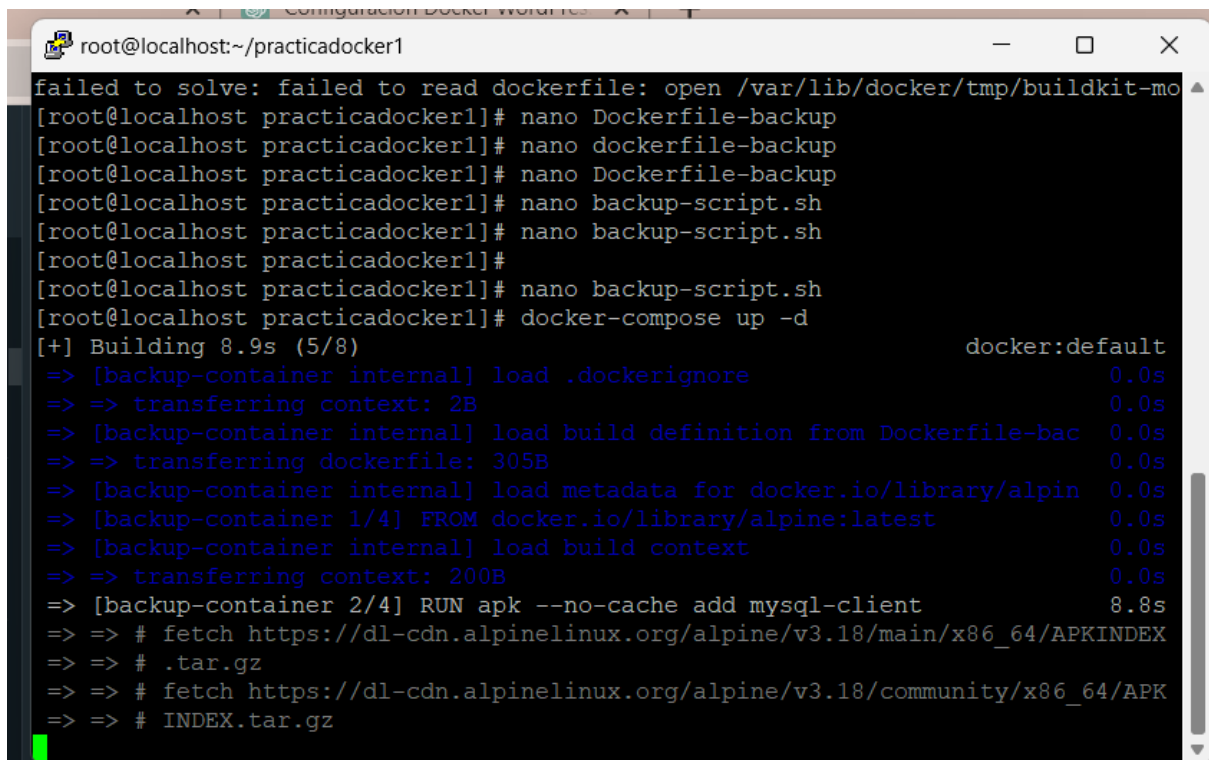
Figura 26: Creamos un archivo bin/sh

22. Ejecutamos los contenedores docker-compose up -d



The screenshot shows a terminal window titled 'root@localhost:~/practicadocker1'. Inside, the GNU nano 5.6.1 editor is open, editing a file named 'backup-script.sh'. The editor's status bar at the top indicates 'GNU nano 5.6.1' and 'backup-script.sh'. The main text area shows the command: `mysql_dump -h mariadb-container -u root -p asdasd --all-databases > /backups/bac`. The bottom status bar shows '[Read 2 lines]' and a series of keyboard shortcuts: `^G Help`, `^O Write Out`, `^W Where Is`, `^K Cut`, `^T Execute`, `^C Location`, `^X Exit`, `^R Read File`, `^_ Replace`, `^U Paste`, `^J Justify`, and `^_ Go To Line`.

Figura 27: Ejecutamos los contenedores



The screenshot shows a terminal window titled 'root@localhost:~/practicadocker1'. It displays the output of a `docker-compose up -d` command. The output starts with 'failed to solve: failed to read dockerfile: open /var/lib/docker/tmp/buildkit-mo'. Then, it shows several `nano` commands being executed: `nano Dockerfile-backup`, `nano dockerfile-backup`, `nano Dockerfile-backup`, `nano backup-script.sh`, `nano backup-script.sh`, `nano backup-script.sh`, and `nano backup-script.sh`. Finally, it shows the `docker-compose up -d` command being executed, which results in 'Building 8.9s (5/8)' and 'docker:default'. The output then shows the progress of building the container, including 'load .dockerignore', 'transferring context: 2B', 'load build definition from Dockerfile-bac', 'transferring dockerfile: 305B', 'load metadata for docker.io/library/alpin', 'FROM docker.io/library/alpine:latest', 'load build context', 'transferring context: 200B', and 'RUN apk --no-cache add mysql-client'. The final output shows the container being built successfully.

Figura 28: Ejecutamos los contenedores

23. Verificamos que los contenedores se esten ejecutando correctamente

```

[root@localhost practicacontainer1]# docker-compose up -d
[+] Building 223.2s (9/9) FINISHED                                docker:default
=> [backup-container internal] load .dockerignore                 0.0s
=> => transferring context: 2B                                    0.0s
=> [backup-container internal] load build definition from Dockerfile-bac 0.0s
=> => transferring dockerfile: 305B                               0.0s
=> [backup-container internal] load metadata for docker.io/library/alpin 0.0s
=> [backup-container 1/4] FROM docker.io/library/alpine:latest     0.0s
=> [backup-container internal] load build context                 0.0s
=> => transferring context: 200B                                   0.0s
=> [backup-container 2/4] RUN apk --no-cache add mysql-client      220.6s
=> [backup-container 3/4] COPY backup-script.sh /backup-script.sh  0.1s
=> [backup-container 4/4] RUN chmod +x /backup-script.sh           0.8s
=> [backup-container] exporting to image                           1.6s
=> => exporting layers                                           1.6s
=> => writing image sha256:692a30cfd08d22a85934a2def58782edc0d243d17a095 0.0s
=> => naming to docker.io/library/practicacontainer1-backup-container 0.0s
[+] Running 6/6
✔ Network practicacontainer1_default                            Created      0.6s
✔ Volume "practicacontainer1_mariadb-data"                      Created      0.0s
✔ Volume "practicacontainer1_backup-data"                       Created      0.0s
✔ Container practicacontainer1-mariadb-container-1             Started      0.1s
✔ Container practicacontainer1-wordpress-container-1           Started      0.1s
✔ Container practicacontainer1-backup-container-1              Started      0.1s
[root@localhost practicacontainer1]# docker ps
CONTAINER ID   IMAGE                                COMMAND                  NAMES                                CREAT
ED            STATUS          PORTS                               COMMAND                  NAMES                                CREAT
8615a7008a5e   wordpress                               "docker-entrypoint.s..." 24 se
conds ago    Up 21 seconds    0.0.0.0:80->80/tcp, :::80->80/tcp practicacontainer1-
wordpress-container-1
5b5ec133a19b   practicacontainer1-backup-container  "sh -c 'while true; ..." 24 se
conds ago    Up 21 seconds                                practicacontainer1-
backup-container-1
6ae705a08da4   mariadb                               "docker-entrypoint.s..." 24 se
conds ago    Up 23 seconds    3306/tcp                                practicacontainer1-
mariadb-container-1
[root@localhost practicacontainer1]#

```

Figura 29: los contenedores se estan ejecutando correctamente

24. Para verificar el respald cada 5 minutos ingresamos

```
root@localhost:~/practicadocker1
conds ago Up 21 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp practica
wordpress-container-1
5b5ec133a19b practica "sh -c 'while true; ...'" 24 se
conds ago Up 21 seconds practica
backup-container-1
6ae705a08da4 mariadb "docker-entrypoin..." 24 se
conds ago Up 23 seconds 3306/tcp practica
mariadb-container-1
[root@localhost practica]# docker exec -it practica-backup-contain
er-1 /bin/sh
/ # ls
backup-script.sh home proc sys
backups lib root tmp
bin media run usr
dev mnt sbin var
etc opt srv
/ # cd backups
/backups # ls -l
total 0
-rw-r--r-- 1 root root 0 Oct 24 20:31 backup.sql
/backups # ls -l
total 0
-rw-r--r-- 1 root root 0 Oct 24 20:31 backup.sql
/backups #
```

Figura 30: verificar el respald cada 5 minutos ingresamos

25. Comprobación de la instalación y pagina de wordpress

← → ↻ ⚠ No seguro | http://192.168.149.131/wp-admin/install.php?s... 🔑 🔗 ☆ ⚙ □ 👤 ⋮

Hola

¡Bienvenido al famoso proceso de instalación de WordPress en cinco minutos! Simplemente completa la información siguiente y estarás a punto de usar la más enriquecedora y potente plataforma de publicación personal del mundo.

Información necesaria

Por favor, proporciona la siguiente información. No te preocupes, siempre podrás cambiar estos ajustes más tarde.

Título del sitio


MI BLOG

Nombre de usuario

EDISON

Los nombres de usuario pueden tener únicamente caracteres alfanuméricos, espacios, guiones bajos, guiones medios, puntos y el símbolo @.

Contraseña

DAVID0998542722 

Strong

Importante: Necesitas esta contraseña para acceder. Por favor, guárdala en un lugar seguro.

Figura 31: pagina de wordpress

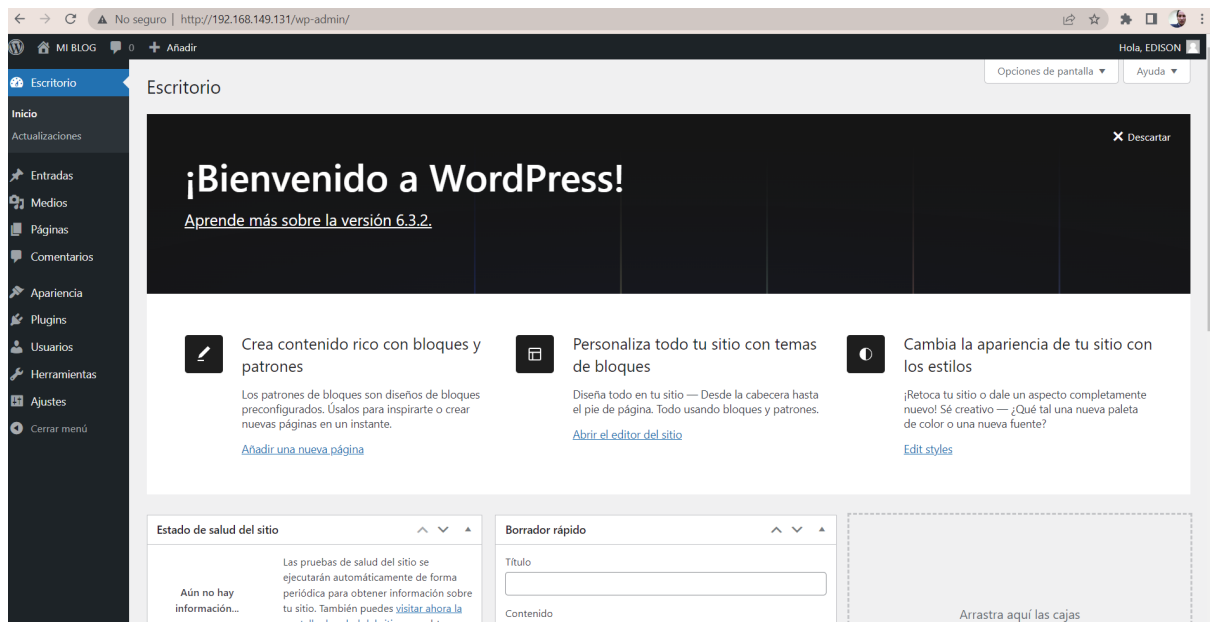


Figura 32: pagina de wordpress

5. CONCLUSIONES

- Se realizó la instalación de los 3 contenedores correctamente como es el caso del primer contenedor de WordPress, MariaDB y Backup.
- Cada 5 minutos se crean respaldos de la base de datos de MariaDb.
- De la misma forma se realizó la instalación de la página de WordPress con la utilización de un contenedor.

6. BIBLIOGRAFÍA

[1] <https://voidnull.es/instalacion-de-docker-en-almalinux-9-1/>
<https://docs.docker.com/compose/>
<https://mariadb.org/>

7. ANEXOS

```
mi_red

docker pull mariadb
docker images
docker pull wordpress
docker images

docker network create red_ey
docker run -d --name servidor_mariadb \
--network red_ey \
-v /opt/mysql_ey:/var/lib/mysql \
-e MYSQL_DATABASE=bd_ey \
-e MYSQL_USER=user_ey \
-e MYSQL_PASSWORD=asdasd \
-e MYSQL_ROOT_PASSWORD=asdasd \
mariadb

docker run -d --name servidor_ey \
--network red_ey \
-v /opt/wordpress:/var/www/html/ey-content \
-e WORDPRESS_DB_HOST=servidor_mariadb \
-e WORDPRESS_DB_USER=user_ey \
-e WORDPRESS_DB_PASSWORD=asdasd \
-e WORDPRESS_DB_NAME=bd_ey \
-p 80:80 \
wordpress

docker stop servidor_ey
docker stop servidor_mariadb

INSTALAMOS DOCKER COMPOSE
curl -L "https://github.com/docker/compose/releases/download/v2.12.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/bin/docker-compose

chmod +x /usr/bin/docker-compose

docker-compose --version

docker volume create mariadb-data
```

Figura 33: ANEXO 1

```

docker volume create mariadb-data
docker volume create backup-data

mkdir practicacontainer1
ls
cd practicacontainer1/
nano docker-compose.yml

version: '3'
volumes:
  mariadb-data:
  backup-data:

services:
  mariadb-container:
    image: mariadb
    environment:
      MYSQL_DATABASE: bd_ey
      MYSQL_USER: user_ey
      MYSQL_PASSWORD: asdasd
      MYSQL_ROOT_PASSWORD: asdasd

    volumes:
      - mariadb-data:/var/lib/mysql

  wordpress-container:
    image: wordpress
    environment:
      WORDPRESS_DB_HOST: mariadb-container
      WORDPRESS_DB_USER: user_ey
      WORDPRESS_DB_PASSWORD: asdasd
      WORDPRESS_DB_NAME: bd_ey
    ports:
      - "80:80"
    depends_on:
      - mariadb-container

  backup-container:
    build:
      context: .

```

Figura 34: ANEXO 2

```

backup-container:
  build:
    context: .
    dockerfile: Dockerfile-backup
  volumes:
    - backup-data:/backups
  depends_on:
    - mariadb-container

nano Dockerfile-backup

FROM alpine:latest
RUN apk --no-cache add mysql-client
COPY backup-script.sh /backup-script.sh
RUN chmod +x /backup-script.sh
CMD ["sh", "-c", "while true; do sh /backup-script.sh; sleep 300; done"]

nano backup-script.sh

#!/bin/sh
mysqldump -h mariadb-container -u root -p asdasd --all-databases > /backups/backup.sql
=====
-----levantamos los contenedores
docker-compose up -d
docker ps
PARA VERIFICAR EN EL BACKUP Y QUE SEA CADA 5 MINUTOS =====
docker exec -it practicadocker1-backup-container-1 /bin/sh
ls
cd backups
ls -l

PARA INGRESAR A MARIA DB

docker exec -it practicadocker1-mariadb-container-1 /bin/bash

```

Figura 35: ANEXO 3

```
PARA INGRESAR A MARIA DB

docker exec -it practicacontainer1-mariadb-container-1 /bin/bash

mysql -u user_ey -p

contraseña ===== asdasd

MYSQL_DATABASE: bd_ey
MYSQL_USER: user_ey
MYSQL_PASSWORD: asdasd
MYSQL_ROOT_PASSWORD: asdasd

ingresamos a nuestro navegador y colocamos nuestra dirección ip para verificar la instalación de wordpress

http://192.168.149.131

para ver si están en la misma red

docker network ls

para ver mis contenedores
docker ps -a

detener los contenedores
docker stop $(docker ps -q)

activar los contenedores
docker start $(docker ps -a -q)
```

Figura 36: ANEXO 4