Practica 1: Despliegue Docker

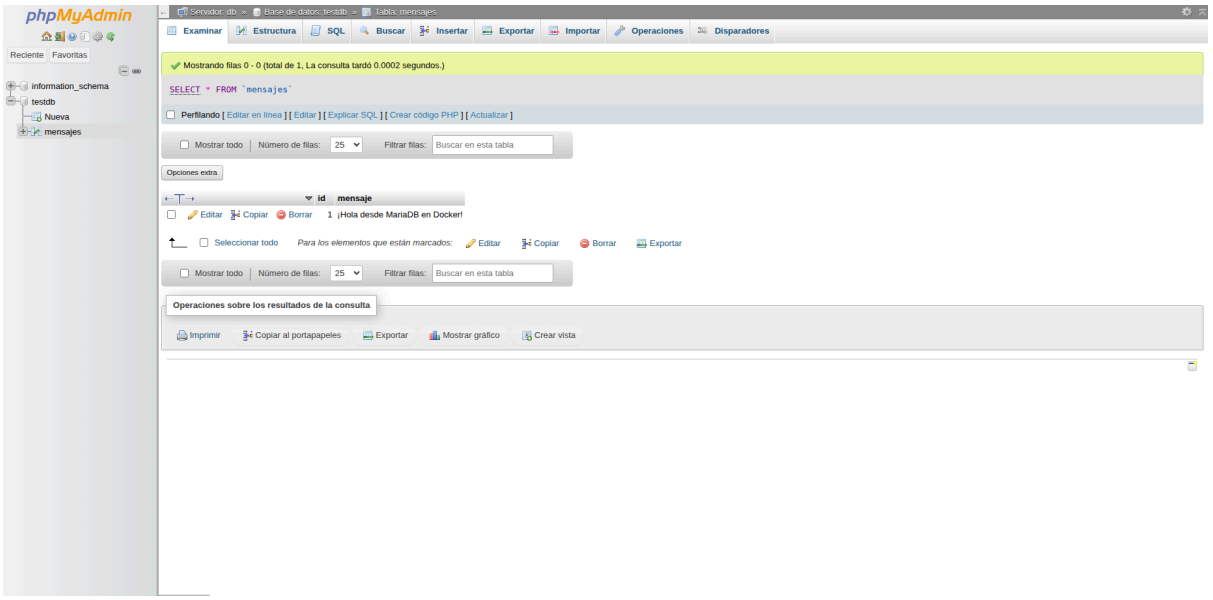
1. Despliegue de Docker

Se estableció la base de la aplicación definiendo los servicios web (PHP con Apache) y db (MariaDB) en docker-compose.yml. El principal beneficio de este paso es la Portabilidad y el Aislamiento, asegurando un entorno de desarrollo consistente. El problema inicial fue el clásico Error de Permisos (permission denied) al intentar comunicarme con el demonio de Docker en el host. La solución fue añadir mi usuario al grupo docker en Ubuntu y reiniciar la sesión de terminal.

Mensaje desde la BD: ¡Hola desde MariaDB en Docker!
Servidor: 63846a93af96

2. Añadimos Interfaz a la BD

Se incorporó el servicio phpmyadmin en el puerto 8081. El beneficio es la usabilidad, ya que permite la gestión, consulta y depuración de la base de datos mediante una interfaz de usuario gráfica, evitando la necesidad de operar únicamente por línea de comandos.



3. Añadimos Variables de Entorno (.env)

El beneficio fundamental es la Seguridad y la Flexibilidad, separando la configuración sensible del código fuente. No surgió ningún problema en esta parte.

4. Añadimos Persistencia al Contenedor

Se implementó un Volumen Nombrado (dbdata) que se mapea al directorio de datos de MariaDB (/var/lib/mysql). El beneficio principal es la Fiabilidad de Datos, garantizando que toda la información almacenada en la base de datos persista en el host incluso si el contenedor db es eliminado, actualizado o detenido. Fue también inmediato de implementar.

5. Distinguimos entre Servicios (Redes)

Se crearon dos redes internas, frontend y backend. El servicio db fue asignado exclusivamente a backend. El beneficio es la Segmentación de Tráfico y Aislamiento, ya que la base de datos queda inaccesible directamente desde la red pública. Después tuve que crear las redes al final del archivo .yml.

6. Check de la Base de Datos (Healthcheck)

Se añadió un healthcheck al servicio db y se configuró al servicio web para que esperara la condición healthy. El beneficio es la estabilidad del despliegue, ya que la aplicación web no intenta conectarse a la base de datos hasta que esta no está completamente lista y funcional, evitando fallos de conexión al inicio.

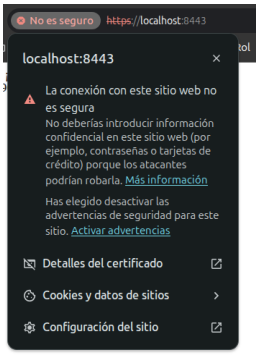
```
usua5pc15@A5PC15:~/Escritorio/Victor/Despliegue/Practica1/ALUMNOS$ sudo docker compose ps
NAME                IMAGE          COMMAND                  SERVICE      CREATED         STATUS                   PORTS
alumnos-db-1        mariadb:10.5   "docker-entrypoint.s…"   db           3 minutes ago   Up 3 minutes (healthy)   3306/tcp
alumnos-phpmyadmin-1   phpmyadmin:5   "/docker-entrypoint.…"   phpmyadmin   3 minutes ago   Up 3 minutes             0.0.0.0:8081->80/tcp, [::]:8081->80/
p
alumnos-tomcat-1    tomcat:9.0     "catalina.sh run"        tomcat       3 minutes ago   Up 3 minutes             8080/tcp
alumnos-web-1       alumnos-web    "docker-php-entrypoi…"   web          3 minutes ago   Up 3 minutes             80/tcp
```

Se puede ver que el proceso de la db pone (healthy)

7. Certificado Autofirmado (TLS) y Proxy Inverso

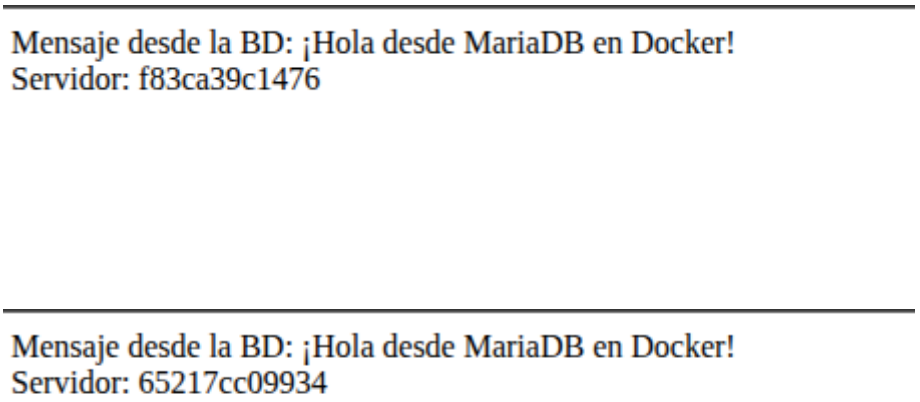
Se introdujo el servicio proxy (Nginx) para gestionar el tráfico y los certificados autofirmados, ruteando de HTTP a HTTPS (8443). El beneficio es la Seguridad de la Comunicación (cifrado TLS) y la Centralización del Tráfico. Tuvimos dos problemas recurrentes:

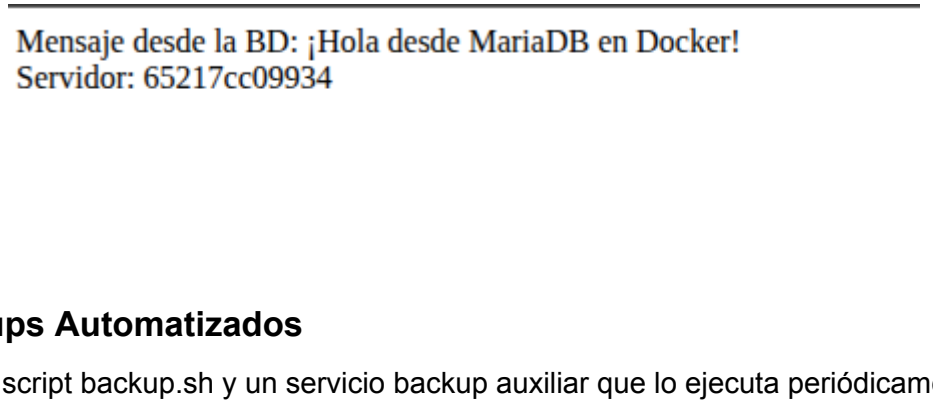
1. Conflicto de Puertos: El puerto 80 del host estaba ocupado, impidiendo el inicio de Nginx. La solución fue detener el servicio que usaba en el host (ej. Apache).
2. Ruteo Fallido: Error 502 Bad Gateway al intentar acceder, porque Nginx no podía establecer conexión con Apache. La solución fue cambiar la declaración de una variable, no entiendo porque daba fallo ya que la sintaxis era correcta.



8. Balanceo de Carga

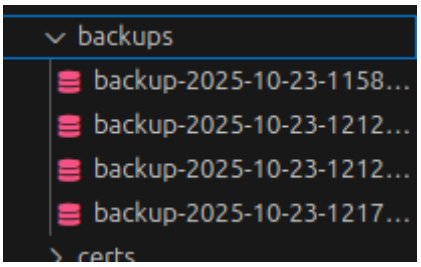
Se escaló el servicio web a dos servidores (--scale web=2). El beneficio es el aumento de la disponibilidad y la tolerancia a fallos. La carga se distribuye entre los dos servidores y, si uno cae, el otro toma el relevo. El principal problema fue que el nombre del servidor no cambiaba al recargar la página. Pero esto era porque no baje el servidor antes de subirlo de nuevo y no se actualizó bien.





9. Backups Automatizados

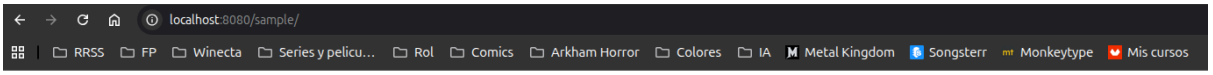
Se creó el script backup.sh y un servicio backup auxiliar que lo ejecuta periódicamente (cada 60 segundos) para hacer un backup de la base de datos, guardando los archivos SQL en un volumen. El beneficio es la Recuperación de Desastres y el Mantenimiento Automatizado, garantizando copias de seguridad consistentes de los datos.



Estos backups se han generado de forma automatica

10. Servidor de Aplicaciones (Tomcat)

Se integró un nuevo servicio, tomcat:9.0, para desplegar aplicaciones Java (.war) y se configuró Nginx para rutear el tráfico a este servicio a través del puerto 8080. El beneficio es que la misma arquitectura de despliegue soporte múltiples tecnologías (PHP y Java). El último problema fue un fallo de ruteo (404 Not Found) por parte de Nginx. La solución fue añadir el puerto 8080 a la lista de puertos del proxy en docker-compose.yml y configurar un bloque server { listen 8080; ... } separado en nginx.conf para el ruteo específico de la aplicación Tomcat.



Sample "Hello, World" Application

This is the home page for a sample application used to illustrate the source directory organization of a web application utilizing the principles outlined in the Application Developer's Guide.

To prove that they work, you can execute either of the following links:

- To a JSP page.
- To a servlet.