Despliegue con docker

RECIPEDIUM

Justificación y descripción

Sara Eugenia Martínez Madruga 2ºDAW

Tabla de contenido

Introducción

El presente documento se ha creado con la intención de justificar y describir el empleo de Docker para desplegar la aplicación web Recipedium en la nube. Para ello, se ha realizado una simulación a través de una máquina virtual (simulando el entorno real de servidor) y nuestra propia máquina (como cliente), proceso que se describe en el apartado “Descripción del proceso” del presente documento.

Justificación

Los motivos que han llevado a elegir el despliegue en la nube a través de Docker, son varios, los cuales procedemos a describir a continuación:

1. ## Facilidad de uso
   Se simplifica mucho el uso y gestión de las aplicaciones, ya que Docker proporciona herramientas para construir, ejecutar, actualizar… los contenedores. Además, configurar y mantener las máquinas virtuales puede ser más complicado que los contenedores.

2. ## Microservicios
   Permite encapsular cada microservicio en un contenedor independiente

3. ## Portabilidad
   Se puede ejecutar en cualquier sistema que soporte Docker

4. ## Uso de recursos más eficiente
   Consume menos recursos que las máquinas virtuales, permite ejecutar más aplicaciones en el mismo servidor al compartir el Kernel los contenedores. Las máquinas virtuales consumen más recursos (CPU, RAM, almacenamiento) que los contenedores.

5. ## Despliegue rápido y replicable
   Agiliza el proceso de despliegue y reduce errores pues crea entornos de desarrollo consistentes. Los contenedores son más rápidos para arrancar y detener que las máquinas virtuales

6. ## Automatización
   Tiene buena integración con herramientas de automatización, esto permite automatizar procesos como despliegue, pruebas, etc.

7. ## Escalable
   Se puede cambiar el número de contenedores o la configuración de cada uno pues estos son ligeros y eficientes

8. ## Económico
   Su eficiencia permite reducir costes de infraestructura

En definitiva, Docker facilita la ejecución de microservicios al proporcionar una forma de empaquetar y distribuir estos servicios como contenedores. Los contenedores de Docker permiten que los microservicios se ejecuten en diferentes entornos de forma consistente y aislada.

Aunque Apache y PHPMyAdmin no son microservicios por sí solos (Apache es un servidor web, y PHPMyAdmin es una herramienta de gestión de bases de datos), pueden ser implementados como

parte de una arquitectura de microservicios, y se pueden ejecutar cada uno en su propio contenedor Docker, para que funcionen de manera independiente y coordinada.

Por todos estos motivos, se ha decidido que el uso de Docker era forma más acertada de desplegar nuestra aplicación.

Descripción del proceso

1. Dockerfile personalizado: ¿por qué no lo hemos usado?

En nuestro caso no hemos necesitado un Dockerfile personalizado, ya que hemos empleado imágenes oficiales de Docker como php:8.1-apache y mysql:8.0.

No hemos requerido configuraciones específicas adicionales más allá de lo que permiten las opciones de configuración de Docker Compose.

Además, las imágenes oficiales ya incluyen todo lo necesario para ejecutar Apache con PHP y MySQL por lo que usar imágenes oficiales nos ha simplificado el despliegue y evitado la necesidad de crear un Dockerfile personalizado.

Si en el futuro fuera necesario instalar dependencias o personalizar la imagen, entonces sí sería recomendable crear un Dockerfile.

2. Creación del archivo docker-compose.yml

Para definir y configurar los servicios necesarios para la aplicación, creamos este archivo, cuya estructura básica consta entre otros de los siguientes elementos:

- version: Versión de la sintaxis de Docker Compose
- services: Define cada uno de los servicios (contenedores) necesarios. Estos servicios a su vez, se configuran usando los siguientes campos (no todos son requeridos).
    - image: Especifica la imagen de Docker a utilizar.
    - ports: Mapea los puertos del host a los del contenedor.
    - volumes: Permite la persistencia de datos y el acceso al código fuente.
    - environment: Variables de entorno para la configuración de los servicios. En nuestro caso empleados con la base de datos MySQL y el gestor de bases de datos PHPMyAdmin
    - depends_on: Define dependencias entre servicios.
    - command: Sobrescribe el comando predeterminado que se ejecuta cuando se inicia un contenedor a partir de una imagen. En nuestro caso lo hemos empleado para instalar las extensiones PHP (mysqli y pdo_mysql que permiten que PHP se conecte y trabaje con bases de datos MySQL usando las librerías MySQLi y PDO MySQL.
    - networks: Nombre de la red interna para comunicación entre servicios
- networks: Permite la comunicación interna entre los servicios.
- volumes: Define volúmenes persistentes para los datos importantes (como la base de datos).

En líneas generales, lo que hemos definido en nuestro archivo docker-compose.yml ha sido:

- Uso de imágenes oficiales: php:8.1-apache, mysql:8.0 y phpmyadmin. Aclarar que, aunque phpMyAdmin es opcional, es muy útil para la gestión visual de la base de datos.
- Configuración de servicios, puertos, volúmenes y algunos comandos extra (como instalar extensiones PHP) definiendo el elemento command
- Definición de una red interna entre servicios para facilitar la comunicación entre ellos.

Obsérvese captura inferior con ejemplo de un fragmento del código de docker-compose.yml que contiene la definición del servicio Apache.

```
apache:
  image: php:8.1-apache
  container_name: apache
  ports:
    - "80:80"      You, 20 hours ago • docker-compose mod
  volumes:
    - ./:/var/www/html
  depends_on:
    - mysql
  networks:
    - recipedium-network
  command: /bin/bash -c "apt-get update && apt-get
  install -y libpq-dev && docker-php-ext-install mysqli
  pdo_mysql && apache2-foreground"
```


3. Preparación del entorno

Creamos una máquina virtual con Ubuntu Server instalado para simular un entorno real de servidor a la que instalamos Docker y Docker Compose para gestionar contenedores de forma sencilla y reproducible. Con lo que simulamos el entorno de producción y aislamos el despliegue del entorno de desarrollo.

Para saber si tenemos Docker y Docker-compose usamos los comandos:

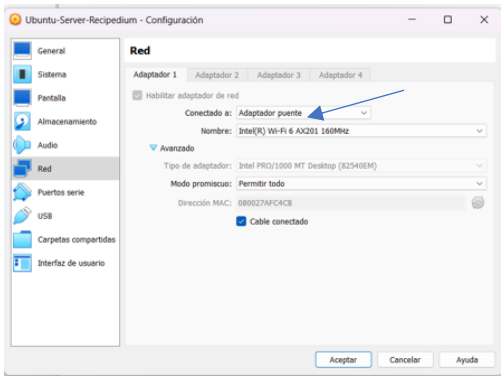
- docker –version
- docker-compose --version

Si no nos aparece la versión, significa que no lo temenos instalado, por lo que debemos instalarlos y habilitarlos, para ello nos valemos de los comandos:

- sudo apt install -y docker.io docker-compose (instala)
- sudo systemctl enable --now docker (habilita)
- sudo usermod -aG docker $USER (permite usar docker sin anteponer sudo)

Esta máquina virtual debe tener una conexión tipo adaptador puente (está en la misma red que mi PC físico y otros dispositivos por lo que puedo acceder a ella desde cualquier otro equipo de la red local. Esto permite aislar el entorno de desarrollo y simular cómo funcionaría el proyecto en producción).

En captura inferior se muestra la configuración de red de la máquina virtual que simulará el entorno real de servidor



4. Gestión del código de la aplicación

Debemos instalar git para poder clonar el proyecto usando el comando:

- sudo apt install -y git

Clonamos el proyecto desde GitHub en la máquina virtual (previamente lo hemos subido a un repositorio, pues intentamos replicar un procedimiento real, ya que el uso de Git está muy extendido porque facilita el control de versiones, la colaboración y asegura que el código en el servidor esté siempre actualizado.) para ello usamos el comando:

- git clone https://github.com/eumardev/recipedium donde eumardev es el nombre de usuario y recipedium el del repositorio

Para sincronizar cambios entre el repositorio local y el servidor usando el comando y descargarse la última versión del repositorio, nos valemos del comando: git pull origin main.

5. Configuración de Docker

Durante la configuración del archivo docker-compose.yml, se han tomado varias decisiones técnicas relevantes:

- Selección de puertos:
  Mapear el puerto 80 del contenedor Apache al puerto 80 del host ("80:80"), en lugar de usar un puerto alternativo como 8080. Esto permite acceder a la aplicación web directamente desde el navegador usando la URL estándar (http://IP-del-servidor/), simulando un entorno de producción real y facilitando el acceso para los usuarios.
- Comprobación de puertos libres:
  Antes de asignar el puerto 80, se ha verificado que no estuviera ocupado por otro servicio en la máquina virtual. Para ello, sea utilizado el comando:
  - sudo netstat -tlnp | grep 8080 (nos muestra qué programa está usando el puerto)
- Red interna personalizada:
  Definimos una red interna (recipedium-network) para aislar la comunicación entre los servicios y evitar conflictos con otros contenedores o servicios externos.
- Persistencia de datos:
  Se ha empleado un volumen nombrado (mysql-data) para asegurar que los datos de la base de datos MySQL se mantengan, aunque el contenedor se reinicie o elimine.

Estas decisiones garantizan un despliegue más profesional, seguro y cercano a un entorno real de producción.

6. Modificación archivo de hosts (solo para simulación)

Este paso lo hemos implementado para posibilitar el acceso a la prueba en red local a través de www.recipedium.com en vez de tener que usar la ip, ya que en un entorno de producción con DNS real no es necesario.

Hemos modificado el archivo hosts en C:\Windows\System32\drivers\etc añadiendo esta línea:

192.168.1.49 www.recipedium.com

7. Despliegue y pruebas

- Levantamos los contenedores con el comando:
  - docker-compose up -d (la orden se ejecuta en segundo plano).
- Hemos probado el acceso desde otros dispositivos de la red local, usando la IP y el nombre personalizado (gracias a la modificación anterior del archivo hosts).

En captura superior, se aprecia el acceso a la aplicación a través de ip

En captura inferior a través de supuesto DNS.

- Importamos la base de datos para asegurarnos de que la aplicación funcione correctamente.
- Se ha comprobado el correcto funcionamiento a base de datos tras el despliegue.

Todos estos pasos garantizan que la aplicación funciona como un servicio real, accesible en red y con todos sus componentes correctamente integrados.