PRÁCTICA 2: Variables de Entorno en Docker Compose

Objetivo

Aprender a gestionar variables de entorno de forma segura y flexible utilizando diferentes métodos.

Conceptos Teóricos

¿Por qué usar variables de entorno?

- Separar configuración del código
- Facilitar diferentes entornos (desarrollo, producción)
- Mantener información sensible segura
- Hacer las aplicaciones más portables

Métodos para definir variables:

1. Directamente en `docker-compose.yml`
2. Usando archivo `.env`
3. Usando archivos específicos con `env_file`
4. Variables del sistema host

Pasos de la Práctica

Paso 1: Crear estructura del proyecto

```
mkdir practica2-variables-entorno
cd practica2-variables-entorno
```

Paso 2: Crear una aplicación Node.js simple

Crea un archivo `app.js` y copiar el siguiente código fuente esta aplicación esta basada en nodejs y permitirá leer y visualizar en el navegador web multiples variables de entorno.

```
// app.js
const express = require('express');
const app = express();

// Leer variables de entorno
const PORT = process.env.PORT || 3000;
const APP_NAME = process.env.APP_NAME || 'Mi Aplicación';
const ENVIRONMENT = process.env.NODE_ENV || 'development';
const DB_HOST = process.env.DB_HOST || 'localhost';
const DB_USER = process.env.DB_USER || 'usuario';
const DB_PASSWORD = process.env.DB_PASSWORD || 'sin_password';
const DB_NAME = process.env.DB_NAME || 'sin_base_datos';

app.get('/', (req, res) => {
res.json({
mensaje: `Bienvenido a ${APP_NAME}`,
entorno: ENVIRONMENT,
puerto: PORT,
configuracion_db: {
host: DB_HOST,
usuario: DB_USER,
password: DB_PASSWORD.replace(/./g, '*'), // Ocultar password
base_datos: DB_NAME
}

});
});


app.listen(PORT, () => {
console.log(`${APP_NAME} ejecutándose en puerto ${PORT}`);
console.log(`Entorno: ${ENVIRONMENT}`);
});
```

Paso 3: Crear Dockerfile para la aplicación

Crea un archivo `Dockerfile`:

```
FROM node:18-alpine
WORKDIR /app
# Crear package.json
RUN echo '{"name":"app-
variables","version":"1.0.0","main":"app.js","dependencies":
```

```
{"express":"^4.18.2"}}' > package.json
# Instalar dependencias
RUN npm install
# Copiar aplicación
COPY app.js .
EXPOSE 3000
CMD ["node", "app.js"]
```

Paso 4: Crear archivo .env

Crea un archivo `.env` (este archivo contiene valores por defecto):

```
COMPOSE_PROJECT_NAME=practica2
APP_NAME=Sistema de Variables
NODE_ENV=development
APP_PORT=3000
MYSQL_ROOT_PASSWORD=root_secret_123
MYSQL_DATABASE=aplicacion_db
MYSQL_USER=app_user
MYSQL_PASSWORD=user_secret_456
```

Paso 5: Crear archivo de variables de producción

Crea un archivo `production.env`:

```
NODE_ENV=production
APP_NAME=Sistema Producción
```

Paso 6: Crear docker-compose.yml

```
version: '3.8'

services:
  app:
    build: .
    container_name: app-variables
    ports:
      - "${APP_PORT}:3000"
    environment:
      PORT: 3000
```

```
      APP_NAME: ${APP_NAME}
      NODE_ENV: ${NODE_ENV}

      DB_HOST: database
      DB_USER: ${MYSQL_USER}
      DB_PASSWORD: ${MYSQL_PASSWORD}
      DB_NAME: ${MYSQL_DATABASE}

    # Método 4: Cargar variables desde archivo adicional
    env_file:
      - production.env

    depends_on:
      - database

  # Base de datos MySQL
  database:
    image: mysql:8.0
    container_name: mysql-variables
    environment:
      # Usando variables del archivo .env
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    ports:
      - "3306:3306"
```

Paso 7: Ejecutar y probar

Ejecute los comandos siguientes para recrear los contenedores.

```
docker compose up --build -d
docker compose logs app
```

Abrir su navegador con la siguiente URL: `http://localhost:3000`

Paso 8: Probar diferentes configuraciones

Modifique el archivo .env y modifique la variable de entorno APP_PORT de 3000 a 4000. Posterior actualizar el archivo es necesario detener y inicializar de nuevo el contenedor.

```
docker compose down
docker compose up -d
```

Ejercicios:

1. Crea un archivo `development.env` con configuraciones específicas de desarrollo
2. Agrega una variable para el puerto de MySQL y úsala en el mapeo de puertos
3. ¿Cual es la funcionalidad del apartado depends_on?
4. ¿Por que NODE_ENV aparece como develoment en lugar de production?
5. ¿Qué función tiene el archivo package.json?
6. Agregar un nuevo contenedor(servicio) con phpmyadmin para acceder a la base de datos. Asignar el puerto 9366 en el host.