



**PROGRAMA EDUCATIVO EN:**  
**TECNOLOGIAS DE LA INFORMACIÓN**

**NOMBRE DE LA MATERIA:**  
**Desarrollo Web Profesional**

**NOMBRE DEL ESTUDIANTE:**  
**CLAUDIA ESPINDOLA LOPEZ**

**NOMBRE DEL PROFESOR:**  
**José Miguel Carrera Pacheco**

**NOMBRE DE LA UNIVERSIDAD:**  
**UNIVERSIDAD TECNOLOGICA DE  
TEHUACAN**

# 1. Introducción

Este documento describe el flujo de trabajo y la infraestructura de desarrollo del proyecto Plataforma Web, incluyendo el uso de Git, Docker, Docker Compose y CI/CD con GitHub Actions.

El propósito es garantizar un entorno de desarrollo consistente, permitir la construcción automática de la aplicación y facilitar la colaboración entre los miembros del equipo.

## 2. Control de Versiones – Git

### 2.1 Inicialización del repositorio

El repositorio se inició con un commit inicial (Initial commit) que establece la estructura básica del proyecto, incluyendo README y carpetas iniciales.

Comandos utilizados:

```
git init
git add .
git commit -m "Initial commit"
git branch -M main
git remote add origin <https://github.com/zimber1/Plataforma_web.git>
git push -u origin main
```

### 2.2 Convenciones de commits

Se adoptaron las siguientes convenciones para mantener claridad en el historial de cambios:

Prefijo	Uso
docs:	Cambios en documentación (README, roles, arquitectura)
feat:	Nuevas funcionalidades
ci:	Cambios en CI/CD
fix:	Correcciones de errores

## **Ejemplo real de commit:**

docs: documentar proyecto, arquitectura, roles, Docker y CI

### **2.3 Flujo de trabajo recomendado**

```
git checkout -b feature/nombre_feature  # Crear rama de feature  
git add .  
git commit -m "feat: descripción del cambio"  
git push origin feature/nombre_feature  
# Luego se realiza merge a main mediante Pull Request
```

## **3. Contenerización – Docker**

### **3.1 Objetivo**

Docker se utiliza para crear un entorno aislado donde la aplicación puede ejecutarse de manera consistente, sin depender de la configuración de la máquina local.

### **3.2 Dockerfile**

```
FROM node:20  
  
WORKDIR /app  
  
COPY ..  
  
RUN npm install || echo "Sin dependencias aún"  
  
EXPOSE 3000  
  
CMD ["node", "-e", "console.log('Contenedor Plataforma Web ejecutándose')"]
```

### **3.3 Descripción de cada instrucción**

- FROM node:20 – Imagen base con Node.js versión 20.
- WORKDIR /app – Establece /app como directorio de trabajo en el contenedor.
- COPY . – Copia todos los archivos del proyecto al contenedor.
- RUN npm install || echo "Sin dependencias aún" – Instala dependencias; si no hay, muestra un mensaje de aviso.
- EXPOSE 3000 – Expone el puerto 3000 del contenedor.
- CMD ["node", "-e", "console.log('Contenedor Plataforma Web ejecutándose')"] – Ejecuta un mensaje de prueba al iniciar el contenedor.

### **3.4 Comandos básicos**

```
docker build -t plataforma-web:latest .
```

```
docker run -p 3000:3000 plataforma-web:latest
```

---

## **4. Orquestación de servicios – Docker Compose**

### **4.1 Objetivo**

Docker Compose permite levantar y gestionar servicios de manera sencilla, especialmente útil en entornos de desarrollo colaborativo.

### **4.2 docker-compose.yml**

```
version: "3.9"

services:
  plataforma_web:
    build: .
    ports:
      - "3001:3000"
```

### **4.3 Explicación**

- version: "3.9" – Define la versión de la sintaxis de Docker Compose.
- services: – Sección que define los servicios a levantar.
- plataforma\_web: – Nombre del servicio principal.
- build: . – Construye la imagen usando el Dockerfile del directorio actual.
- ports: "3001:3000" – Mapea el puerto 3000 del contenedor al puerto 3001 de la máquina host.

### **4.4 Comandos básicos**

```
docker-compose up -d # Levantar servicio
```

```
docker-compose ps  # Ver contenedores corriendo  
docker-compose down # Detener servicios
```

## 5. Integración Continua – GitHub Actions

### 5.1 Objetivo

Automatizar la construcción de la imagen Docker en cada push a la rama main, asegurando que el proyecto se pueda levantar correctamente en cualquier entorno.

### 5.2 Workflow (.github/workflows/ci.yml)

```
name: CI Pipeline

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: Descargar código
        uses: actions/checkout@v4

      - name: Construir imagen Docker
        run: docker build -t plataforma_web .
```

### 5.3 Explicación

- on: push – Ejecuta el pipeline al hacer push a main.
- runs-on: ubuntu-latest – Runner de Ubuntu para ejecutar el workflow.
- actions/checkout@v4 – Descarga el código del repositorio en el runner.
- docker build -t plataforma\_web . – Construye la imagen Docker usando el Dockerfile del proyecto.

## 5.4 Beneficios

- Garantiza que la imagen Docker siempre se pueda construir.
- Permite detectar errores de configuración antes de hacer merge a `main`.
- Base para agregar pruebas automáticas o despliegue automatizado en el futuro.

## 7. Conclusión

Este manual proporciona las instrucciones necesarias para:

- Gestionar versiones del proyecto con Git.
- Ejecutar la aplicación en un entorno Dockerizado.
- Levantar el proyecto de manera sencilla usando Docker Compose.
- Validar la construcción automática de la imagen con GitHub Actions.

El manual refleja exactamente la configuración actual del repositorio, y sirve como guía formal para desarrolladores o evaluadores del proyecto.