Informe Académico

Automatización de API con Docker, GitHub Actions y Telegram

Autor: Ángel Panadero Rodríguez

Fecha: 7 de diciembre de 2025

Introducción

En este proyecto he construido un flujo de trabajo completo que va desde el desarrollo de una API en Node.js hasta su despliegue automatizado usando Docker, GitHub Actions y un sistema de notificaciones en Telegram q te avisa cuando haces un commit a la rama main.

Base de datos

He creado una base de datos MySQL llamada `afundamentos`. La base de datos tiene dos tablas:

- `grupos`: contiene la información de distintos grupos como Administradores, Profesores o Alumnos.
- `usuarios`: almacena nombres, edades, contraseñas y el grupo al que pertenece cada usuario.

Incluí registros de ejemplo para poder probar la API fácilmente desde el principio.

Desarrollo de la API

La API está creada con Express y usa el paquete `mysql2/promise` para conectarse a la base de datos. Utilizo `async/await` para que el código sea más limpio y fácil de entender.

Los endpoints principales son:

- `/` – Mensaje de prueba indicando que la API está funcionando.
- `/grupos` – Devuelve la lista de grupos.
- `/usuarios` – Devuelve todos los usuarios junto con su grupo.
- `/usuarios/:id` – Devuelve un usuario concreto por su ID.

Un fragmento del código principal es:

```
app.get('/usuarios', async (req, res) => {
  const [rows] = await pool.query(`
    SELECT u.id, u.nombre, u.edad, g.nombre AS grupo
    FROM usuarios u
    LEFT JOIN grupos g ON u.id_grupo = g.id
  `);
  res.json(rows);
});
```

Toda la configuración de la conexión se realiza usando variables de entorno para que funcione igual dentro y fuera de Docker.

Contenedor de Docker

Para ejecutar la API dentro de un contenedor creé un `Dockerfile`. Este archivo instala las dependencias, copia el código y ejecuta el servidor:

```
FROM node:20
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "server.js"]
```

Luego configuré `docker-compose.yml` para levantar dos servicios:

- **db**: un contenedor MySQL.
- **api**: la API ejecutándose en Node.js.

Con un solo comando se levanta todo el entorno:

```
docker compose up -d
```

Durante la configuración solucioné problemas como puertos ocupados, reinicios del contenedor y variables de entorno incorrectas.

Publicación en Docker Hub

Creé un repositorio en Docker Hub:

`angelrxdriguez/ejemplo-api`

La imagen generada por la API queda guardada allí para poder usarla en cualquier lugar o incluirla en despliegues futuros.

Automatización con GitHub Actions

Para automatizar el proceso, preparé un workflow que se ejecuta automáticamente cuando hago un `push` en la rama principal. El workflow hace lo siguiente:

- Descarga el repositorio.
- Construye la imagen Docker de la API.
- Publica la imagen en Docker Hub con dos etiquetas:

    - **latest**
    - El hash del commit para identificar la versión exacta.

- Envía una notificación a mi Telegram personal.

Los secretos necesarios (`DOCKER_USERNAME`, `DOCKER_PASSWORD`, `TELEGRAM_BOT_TOKEN`, `TELEGRAM_CHAT_ID`) se guardan de forma segura en GitHub.

Fragmento real del workflow

```
- name: Construir y subir imagen Docker
  uses: docker/build-push-action@v5
  with:
    context: .
    file: ./Dockerfile
    push: true
    tags: |
      angelrxdriguez/ejemplo-api:latest
      angelrxdriguez/ejemplo-api:${{ github.sha }}
```

Bot y notificaciones en Telegram

Para recibir notificaciones automáticas creé un bot con `@BotFather`. Después envié un mensaje al bot y consulté mi `chat_id` usando:

`https://api.telegram.org/botTOKEN/getUpdates`

El resultado mostró que mi chat privado tenía el identificador:

8353125857

Gracias a este dato, cada vez que la imagen se publica correctamente, GitHub Actions me envía un aviso.

Código usado para enviar la notificación

```
- name: Notificar por Telegram
  if: success()
  run: |
    MESSAGE="Nueva imagen subida a Docker Hub%0ARepo: ${{ github.repository
    curl -s -X POST "https://api.telegram.org/bot${{ secrets.TELEGRAM_BOT_TOK
      -d chat_id="${{ secrets.TELEGRAM_CHAT_ID }}" \
      -d text="$MESSAGE"
```

Esta notificación aparece en mi chat personal en segundos, confirmando que todo el proceso ha funcionado bien.

Problemas y soluciones

Durante el desarrollo surgieron varios errores habituales:

- **Permisos denegados al subir la imagen a Docker Hub**: El problema era un nombre incorrecto de los secrets. Simplemente modifique.
- **Telegram devolvía "chat not found"**: Esto ocurria pq en la URL me cargaba el "bot.ªntes del token. Hay que poner el enlace con la palabra "bot.ªntes del token.
- **Errores de YAML en GitHub Actions**: Algunos saltos de línea se interpretaban como claves nuevas. Lo solucioné usando una sola línea con saltos codificados (%0A).

Conclusión

Este proyecto me ha permitido construir un flujo de trabajo completo y profesional. Ahora tengo:

- Una API que funciona dentro de contenedores Docker.
- Una imagen que se sube automáticamente a Docker Hub.
- Un sistema de avisos en Telegram para confirmar que todo se ha ejecutado correctamente.