

Lab Docker: Frontend + Backend

Lo mismo que deployaste en EC2 en la sesión 2 — pero ahora con contenedores Docker y un solo comando.



ANTES DE EMPEZAR

- ☐ Docker Desktop instalado y corriendo
- ☐ Terminal abierta

PARTE 1 — DESCARGAR EL PROYECTO

1 Descargar la carpeta del lab

Todo el proyecto ya está listo. Solo descargá la carpeta y entrá:

```
terminal

# Clonar el repo del curso
$ git clone https://github.com/micadevops/camino-devops-en-aws.git

# Entrar a la carpeta del lab
$ cd camino-devops-en-aws/sesion3/sesion3-lab-home-docker
```

2 Ver qué hay adentro

```
sesion3-lab-home-docker/
├── backend/
│   ├── backend.py ← API Flask
│   ├── requirements.txt ← dependencias
│   └── Dockerfile
├── frontend/
│   ├── frontend.html ← trophy card
│   └── Dockerfile
└── docker-compose.yml
```

3 Dockerfile del backend

```
FROM python:3.12-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY backend.py .
EXPOSE 5000
CMD ["python", "backend.py"]
```

Parte de una imagen de Python, instala Flask, copia el código y arranca la app.

4 Dockerfile del frontend

```
FROM nginx:alpine
COPY frontend.html /usr/share/nginx/html/index.html
EXPOSE 80
```

El frontend es solo HTML — no necesita Docker propio para funcionar, pero lo metemos en nginx para simular un servidor web real como teníamos con Apache en EC2.

5 docker-compose.yml

```
services:

  backend:
    build: ./backend
    ports:
      - "5000:5000"

  frontend:
    build: ./frontend
    ports:
      - "80:80"
    depends_on:
      - backend
```

Un solo archivo que define los dos servicios, sus puertos y el orden de arranque.

PARTE 3 — LEVANTAR TODO

6 Un solo comando

```
terminal

$ docker compose up

[+] Building ...
```

[+] Running 2/2

- ✓ Container backend Started
- ✓ Container frontend Started

7

Probar en el browser

```
browser

# Frontend (trophy card)
http://localhost

# Backend directo
http://localhost:5000/info?nombre=TuNombre
```



Ingresá tu nombre en el formulario → Generar mi página → aparece la trophy card. Lo mismo que en la sesión 2 pero sin EC2, sin SSH, sin instalar nada.

PARTE 4 — COMANDOS ÚTILES

8

Ver los contenedores corriendo

```
terminal nueva

$ docker ps

CONTAINER ID  IMAGE  PORTS  NAMES
a1b2c3...  backend  0.0.0.0:5000->5000/tcp  ...backend-1
d4e5f6...  frontend  0.0.0.0:80->80/tcp  ...frontend-1
```

9

Ver los logs

```
terminal

# Logs de todos los servicios
$ docker compose logs

# Logs solo del backend, en tiempo real
$ docker compose logs -f backend
```

10

Entrar a un contenedor

```
terminal

$ docker exec -it sesion3-lab-home-docker-backend-1 bash

root@a1b2c3:/app#
root@a1b2c3:/app# ls
backend.py requirements.txt
root@a1b2c3:/app# exit
```

11

Parar todo

```
terminal

# Ctrl+C en la terminal donde corre, o:
$ docker compose down

✓ Container frontend Removed
✓ Container backend Removed
```

BONUS — EXPERIMENTÁ

B1

Correr en background

```
terminal

$ docker compose up -d
(libera la terminal, todo corre de fondo)
```

B2

Reconstruir después de cambiar código

```
terminal

$ docker compose up --build
(rebuilds las imágenes con tus cambios)
```

B3

Ver cuánto ocupa Docker

```
terminal

$ docker system df

# Limpiar todo lo que no se usa
$ docker system prune
```

Listo! Hiciste lo mismo que en la sesión 2 (dos servidores conectados) pero en vez de configurar EC2 + SSH + instalar paquetes, fue un solo `docker compose up`.