# Práctica Sesión 12 : Node + Express + Docker en Codespaces

## 1) Crear repo y abrir Codespace

1. En GitHub: **New repository**
    - Nombre: codespaces-node-docker
    - Create
2. Dentro del repo: botón **Code**
3. Pestaña **Codespaces**
4. **Create codespace on main**

## 2) Crear los archivos en la raíz

En el explorador de VS Code (Codespaces), crea estos archivos:

- index.js
- package.json
- Dockerfile

## 3) Pegar este contenido

**index.js**

// Importa la librería Express (framework para crear servidores web en Node)

const express = require("express");

// Crea una aplicación/servidor Express

```
const app = express();

// Define el puerto donde escuchará el servidor dentro del contenedor
const PORT = 3000;

// Define una ruta GET para la raíz "/"
app.get("/", (req, res) => {
  // Cuando alguien visite "/", responde con este texto
  res.send("Hola desde Docker en Codespaces");
});

// Arranca el servidor
// "0.0.0.0" = escucha en todas las interfaces de red del contenedor (necesario para Codespaces/puertos)
app.listen(PORT, "0.0.0.0", () => {
  // Mensaje en consola para confirmar que el servidor está arriba
  console.log(`Servidor escuchando en puerto ${PORT}`);
});
```

**package.json (Debéis borrar los comentarios em JSON)**

```
{
  "name": "codespaces-node-docker",       // Nombre del proyecto (solo identificador)
  "version": "1.0.0",                     // Versión del proyecto
  "main": "index.js",                     // Archivo principal de entrada
```

```
  "scripts": {
    "start": "node index.js"              // Comando: npm start -> ejecuta "node index.js"
  },
  "dependencies": {
    "express": "^4.19.2"                  // Dependencia necesaria para usar Express
  }
}
```

**Dockerfile**

```
# Usa una imagen oficial de Node.js (versión 20) ligera basada en Alpine Linux
FROM node:20-alpine

# Define el directorio de trabajo dentro del contenedor
WORKDIR /app

# Copia SOLO el package.json al contenedor (para instalar dependencias)
COPY package.json .

# Instala dependencias (Express) dentro del contenedor
RUN npm install
```

```
# Copia el archivo del servidor al contenedor
COPY index.js .

# Documenta que el contenedor usará el puerto 3000 (informativo)
EXPOSE 3000

# Comando por defecto al arrancar el contenedor: ejecuta "npm start"
CMD ["npm", "start"]
```

### 4) Construir la imagen y ejecutar el contenedor

Abre la terminal (Terminal → New Terminal) y ejecuta:

```
docker build -t hola-node .
docker run -p 3000:3000 hola-node
```

Deja esa terminal corriendo (verás: “Servidor escuchando en puerto 3000”).

Pulsa abrir en el Navegador.

### 5) Comprobar que responde dentro del Codespace

Abre **otra terminal** (o una pestaña nueva de terminal) y ejecuta:

```
curl http://localhost:3000
```

Debe salir:
Hola desde Docker en Codespaces

## 6) Añadir el puerto manualmente en Codespaces (Por si no os aparece la opción en el apartado anterior)

1. Abre la pestaña **PORTS**
2. Pulsa **Add Port**
3. Escribe 3000 y Enter
4. En la fila del puerto 3000:
    - Cambia **Visibility** a **Public** (o “Org” si te lo pide)
5. Pulsa el icono **Open in Browser** o copia la URL que aparece.

## 7) Parar el contenedor

En la terminal donde corre el contenedor:

- Ctrl + C

Y si quieres limpiar:

docker ps

docker stop ID

## 8) Guardar y subir al repositorio

git add .

git commit -m "Node + Docker en Codespaces"

git push

# Comandos de Git

### Ver estado del repositorio

git status

- Te dice qué archivos están modificados, sin trackear, listos para commit, etc.

### 1) Configurar tu identidad (solo la primera vez, si no está configurado)

git config --global user.name "Tu Nombre"

git config --global user.email "tuemail@correo.com"

- Git firma los commits con tu nombre y email.

Ver configuración:

git config --global --list

### 2) Ver en qué rama estás

git branch

- Normalmente será main.

### 3) Añadir archivos al “staging” (prepararlos para commit)

Añadir todo:

git add .

Añadir uno en concreto:

git add index.js

### 4) Confirmar cambios

git commit -m "Mi primer Docker con Node en Codespaces"

- Guarda un “punto de control” con tus cambios.

### 5) Subir cambios a GitHub

git push

- Sube los commits al repo remoto (GitHub).

### 6) Ver historial de commits

git log --oneline

### 7) Traer cambios del remoto

git pull

- Baja y aplica cambios de GitHub a tu carpeta.

### 8) Ver diferencias (antes de commit)

git diff

- Cambios no añadidos al staging.

### 9) Quitar un archivo del staging (si lo añadiste por error)

git restore --staged archivo.txt