

1- Primero creamos los archivos Dockerfile

Usamos una imagen de Node.js

FROM node:18-alpine

Establecemos el directorio de trabajo dentro del contenedor

WORKDIR /app

Copiamos los archivos de la aplicación al contenedor

COPY package*.json ./

COPY app.js .

Instalamos las dependencias de la aplicación

RUN npm install

Exponemos el puerto en el que la aplicación va a escuchar

EXPOSE 80

Comando para ejecutar la aplicación

CMD ["node", "app.js"]

2- Configuramos los archivos app.js para cada contenedor

web-1:

```
web-1> app.js > ...
1  const http = require('http');
2
3  const server = http.createServer((req, res) => {
4    res.writeHead(200, { 'Content-Type': 'text/plain' });
5    res.end(' servidor web-1');
6  });
7
8  const PORT = process.env.PORT || 80;
9
10 server.listen(PORT, () => {
11   console.log('El Servidor web-1 escuchando en el puerto ${PORT}');
12 });
13
```

web-2:

```
web-2> app.js > ...
1  const http = require('http');
2
3  const server = http.createServer((req, res) => {
4    res.writeHead(200, { 'Content-Type': 'text/plain' });
5    res.end(' servidor web-2');
6  });
7
8  const PORT = process.env.PORT || 80;
9
10 server.listen(PORT, () => {
11   console.log('El Servidor web-2 escuchando en el puerto ${PORT}');
12 });
13
```

3- Configuramos el nginx.conf

```
balanceador > cat nginx.conf
1  worker_processes 1;
2
3  events {
4      worker_connections 1024;
5  }
6
7  http {
8      upstream backend {
9          server web1:80;
10         server web2:80;
11     }
12
13     server {
14         listen 80;
15
16         location / {
17             proxy_pass http://backend;
18             proxy_set_header Host $host;
19             proxy_set_header X-Real-IP $remote_addr;
20         }
21     }
22 }
23
```

Este archivo de configuración de Nginx establece un servidor web con un solo proceso de trabajador y una capacidad máxima de 1024 conexiones simultáneas. Además, configura un equilibrador de carga que distribuye las solicitudes entrantes entre dos servidores backend (web1 y web2) en el puerto 80.

4- Configurar el docker compose

```
balanceador > cat docker-compose.yml
1  version: '3'
2  services:
3      nginx:
4          image: nginx:latest
5          ports:
6              - "8088:80"
7          volumes:
8              - ./nginx.conf:/etc/nginx/nginx.conf
9          depends_on:
10             - web1
11             - web2
12          networks:
13             - backend
14
15      web1:
16          image: web1_image
17          container_name: web1 # Asigna un nombre personalizado al contenedor
18          ports:
19              - "8001:80"
20          networks:
21             - backend
22
23      web2:
24          image: web2_image
25          container_name: web2 # Asigna un nombre personalizado al contenedor
26          ports:
27              - "8002:80"
28          networks:
29             - backend
30
31  networks:
32      backend:
33
```

Este archivo Docker Compose define tres servicios: `nginx`, `web1`, y `web2`. Nginx actúa como un equilibrador de carga para `web1` y `web2`, y todos están conectados a la red `backend`.