Crear imagen de Nodejs con docker

Gabriel Rodríguez Flores

February 13, 2024

ADA ITS
INSTITUTO TECNOLÓGICO SUPERIOR

- Crear imagen de Nodejs con docker
- Automatizar el despliegue con docker-compose
- Definir variables de entorno

Contents

1 Teoría **3**
1.1 Dockerfile . . . 3
1.1.1 Instrucciones . . . 3
1.2 dockerignore . . . 3
1.3 Docker-compose . . . 3
1.4 Utilidades . . . 3
1.4.1 No puedo eliminar una red . . . 3

2 Ejemplos **4**
2.1 Dockerfile . . . 4
2.1.1 API Nodejs . . . 4
2.1.2 API Nodejs con Typescript . . . 4
2.2 dockerignore . . . 5
2.3 Docker-compose . . . 5
2.3.1 Con variables de entorno . . . 5
2.3.2 Sin definir las variables de entorno . . . 5
2.4 Comandos para ejecutar . . . 6

3 Ejercicios **6**

4 Entregables **6**
4.1 En clase . . . 6
4.2 Tarea . . . 6

1 Teoría

1.1 Dockerfile

- Dockerfile reference - Official
- Node.js Docker Best Practices Optimización y seguridad

1.1.1 Instrucciones

- `FROM` - Imagen base
- `WORKDIR` - Directorio de trabajo
- `COPY` - Copiar archivos
- `RUN` - Ejecutar comandos
- `CMD` - Comando por defecto
- `EXPOSE` - Puertos a exponer
- `USER` - Usuario por defecto
- `ENTRYPOINT` - Entrypoint por defecto
- `ARG` - Argumentos
- `ENV` - Variables de entorno
- `ADD` - Copiar archivos y directorios

1.2 dockerignore

Como `.gitignore` pero para Docker. Ignora archivos y directorios que no se quieren incluir en la imagen.

1.3 Docker-compose

Debe tener la propiedad build para que se construya la imagen.

1.4 Utilidades

1.4.1 No puedo eliminar una red

```
# 1. Inspect the network which we are unable to delete
docker network inspect [<id> or <name>]
# 2. Disconnect the network
```

```
docker network disconnect -f [<networkID> or <networkName>] [<endpointName
    > or <endpointId>]
# 3. Delete unused networks
docker network prune
```

2 Ejemplos

2.1 Dockerfile

2.1.1 API Nodejs

```
FROM node:14.15.4-alpine

WORKDIR /usr/src/app
COPY package*.json ./

RUN npm ci --production --no-optional

COPY ./src ./src

EXPOSE 3000

USER node

CMD [ "node", "src/index" ]
```

2.1.2 API Nodejs con Typescript

```
# Build stage
FROM node:latest as build

WORKDIR /app
COPY package*.json ./
COPY tsconfig*.json ./
COPY src ./src

RUN npm ci
RUN npm run build

# Launch stage
FROM node:20-alpine

WORKDIR /app
COPY package*.json ./
RUN npm ci --only=production
```

```
COPY --from=build /app/dist .

USER node

ENTRYPOINT [ "node" ]
CMD [ "main.js" ]
```

2.2 dockerignore

```
node_modules
coverage
.env
.nyc_output
```

2.3 Docker-compose

2.3.1 Con variables de entorno

```
version: '3.8'
services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=production
      - PORT=3000
    volumes:
      - .:/usr/src/app
    networks:
      - backend
```

2.3.2 Sin definir las variables de entorno

```
version: '3.8'
services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "${PORT}:${PORT}"
```

```
    environment:
      - PORT
```

2.4 Comandos para ejecutar

```
# Levantar el contenedor con variables de entorno del sistema
docker-compose up -d
# Levantar el contenedor con variables de entorno del archivo .env
docker-compose --env-file .env up -d
```

3 Ejercicios

4 Entregables

4.1 En clase

4.2 Tarea