

Ingeniera de Software 3

TP 6: Construcción de imágenes docker.

4.1: Descripción de instrucciones:

FROM: Se utiliza para especificar la imagen base a partir de la cual se construirá la nueva imagen del contenedor.

RUN: se utiliza para ejecutar comandos en una nueva capa sobre la imagen actual. Estos comandos suelen utilizarse para instalar software.

ADD: En Docker, la instrucción "ADD" se utiliza para copiar archivos y directorios desde el sistema de archivos del host al sistema de archivos del contenedor Docker durante el proceso de construcción de la imagen.

COPY: Se utiliza para copiar archivos y directorios desde el sistema de archivos del host al sistema de archivos del contenedor durante la construcción de la imagen. Sin embargo, COPY no admite URLs y se utiliza principalmente para copiar archivos locales.

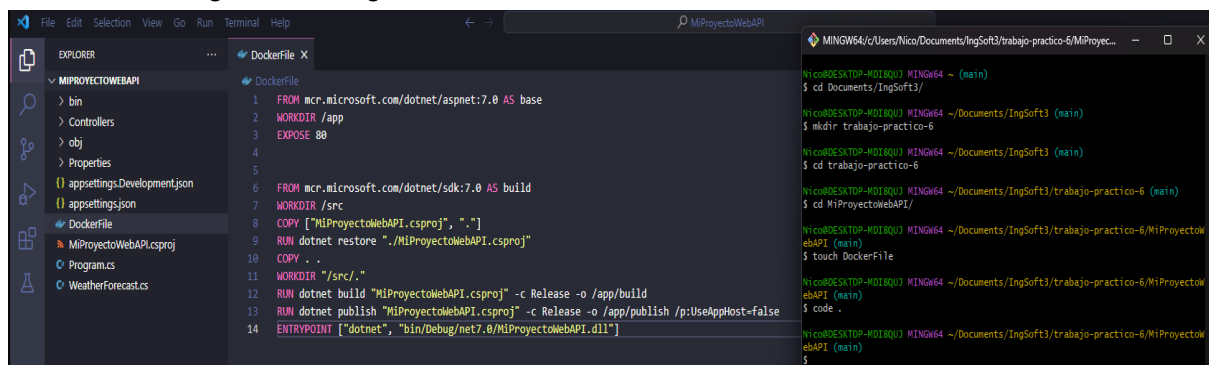
EXPOSE: Se utiliza para especificar los puertos en los que el contenedor escuchará las conexiones entrantes.

CMD: Se utiliza para proporcionar un comando y sus argumentos predeterminados que se ejecutarán cuando se inicie el contenedor.

ENTRYPOINT: Se utiliza para definir un comando que se ejecutará como proceso principal del contenedor.

4.2: Generar imagen docker

Utilizamos los resultados del paso 1 del TP5 y agregamos un archivo "Dockerfile" donde insertamos el siguiente código.



```
File Edit Selection View Go Run Terminal Help
EXPLORER
  MIPROJECTOWEBAPI
    bin
    Controllers
    obj
    Properties
    appsettings.Development.json
    appsettings.json
    Dockerfile
    MiProyectoWebAPI.csproj
    Program.cs
    WeatherForecast.cs
  Dockerfile X
1 FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base
2 WORKDIR /app
3 EXPOSE 80
4
5
6 FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
7 WORKDIR /src
8 COPY ["MiProyectoWebAPI.csproj", "."]
9 RUN dotnet restore "/MiProyectoWebAPI.csproj"
10 COPY . .
11 WORKDIR "/src/"
12 RUN dotnet build "MiProyectoWebAPI.csproj" -c Release -o /app/build
13 RUN dotnet publish "MiProyectoWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false
14 ENTRYPOINT ["dotnet", "bin/Debug/net7.0/MiProyectoWebAPI.dll"]

MINGW64~/Users/Nico/Documents/IngSoft3/trabajo-practico-6/MiProyectoWebAPI
$ cd Documents/IngSoft3/
$ mkdir trabajo-practico-6
$ cd trabajo-practico-6
$ cd MiProyectoWebAPI/
$ touch DockerFile
$ code .
$
```

Generamos la imagen

```
Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6/MiProyectoWebAPI (main)
$ docker build -t miprojectowebapi .
#0 building with "default" instance using docker driver

#1 [internal] load .dockerignore
#1 transferring context: 2B done
#1 DONE 0.0s

#2 [internal] load build definition from Dockerfile
#2 transferring dockerfile: 519B done
#2 DONE 0.0s

#3 [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0
#3 DONE 0.8s

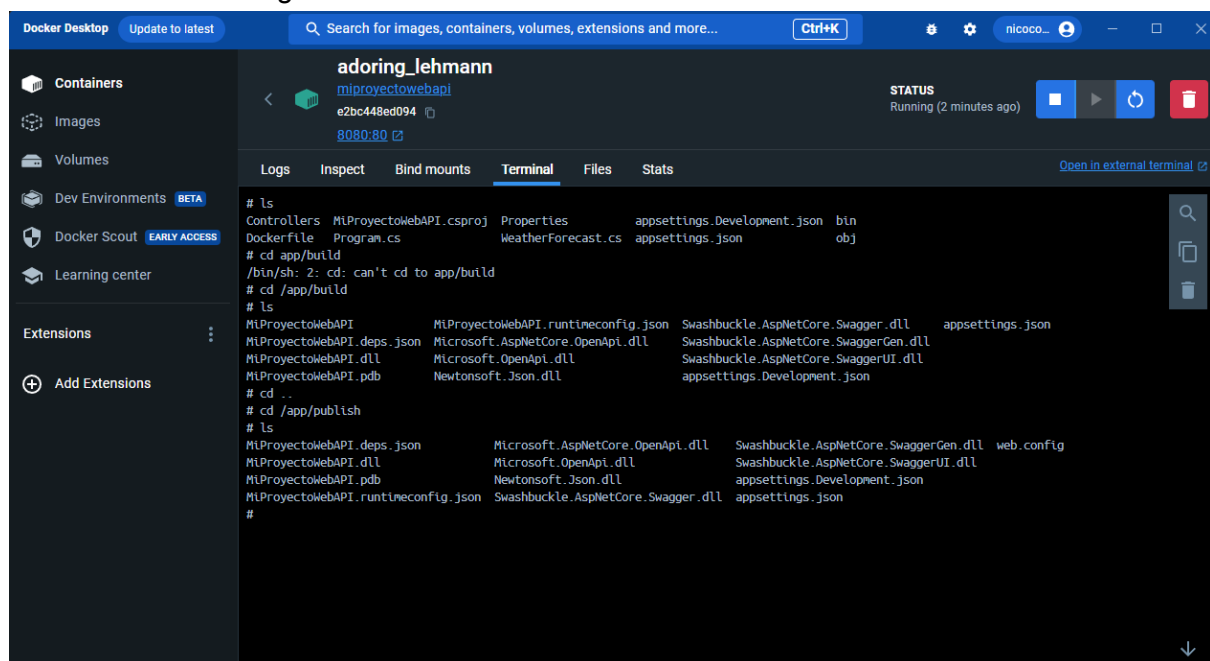
#4 [build 1/8] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:d0755fbf1ed34aec79c127fd1dd01b80587acdb8ff50e5a68d1adf8b076922b
#4 DONE 0.0s
```

Ejecutamos el contenedor, uso el comando winpty ya que es una utilidad que permite interactuar correctamente con contenedores Docker en el emulador de terminal GitBash.

```
Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6/MiProyectoWebAPI (main)
$ docker run -p 8080:80 -it --rm miprojectowebapi
the input device is not a TTY. If you are using mintty, try prefixing the command with 'winpty'

Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6/MiProyectoWebAPI (main)
$ winpty docker run -p 8080:80 -it --rm miprojectowebapi
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /src
```

Y ahora vemos los siguientes directorios



4.3: Dockerfiles multi etapas

Modificamos el Dockerfile

```
Dockerfile
1 FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base
2 WORKDIR /app
3 EXPOSE 80
4
5
6 FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
7 WORKDIR /src
8 COPY ["MiProyectoWebAPI.csproj", "."]
9 RUN dotnet restore "./MiProyectoWebAPI.csproj"
10 COPY . .
11 WORKDIR "/src/."
12 RUN dotnet build "MiProyectoWebAPI.csproj" -c Release -o /app/build
13
14 FROM build AS publish
15 RUN dotnet publish "MiProyectoWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false
16
17 FROM base AS final
18 WORKDIR /app
19 COPY --from=publish /app/publish .
20 ENTRYPOINT ["dotnet", "MiProyectoWebAPI.dll"]
```

Primero definimos la etapa inicial llamada base, el directorio en el que vamos a trabajar en este caso /app y expone el puerto 80 en el contenedor.

Definimos una etapa llamada build donde definimos el directorio que vamos a trabajar /src, copiamos el archivo "MiProyectoWebApi, ejecutamos dotnet restore para restaurar las dependencias, copiamos el código de la aplicación al directorio de trabajo y lo corremos en /app/build

Definimos una etapa llamada publish y hacemos un publish de la aplicación en /app/publish

Al último definimos una etapa llamada final que va a trabajar en el directorio /app, copia los archivos y por último hace un entrypoint como "dotnet MiProyectoWebApi"

```
Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6/MiProyectoWebAPI (main)
$ docker build -t miprojectowebapi .
#0 building with "default" instance using docker driver

#1 [internal] load .dockerignore
#1 transferring context: 2B done
#1 DONE 0.0s

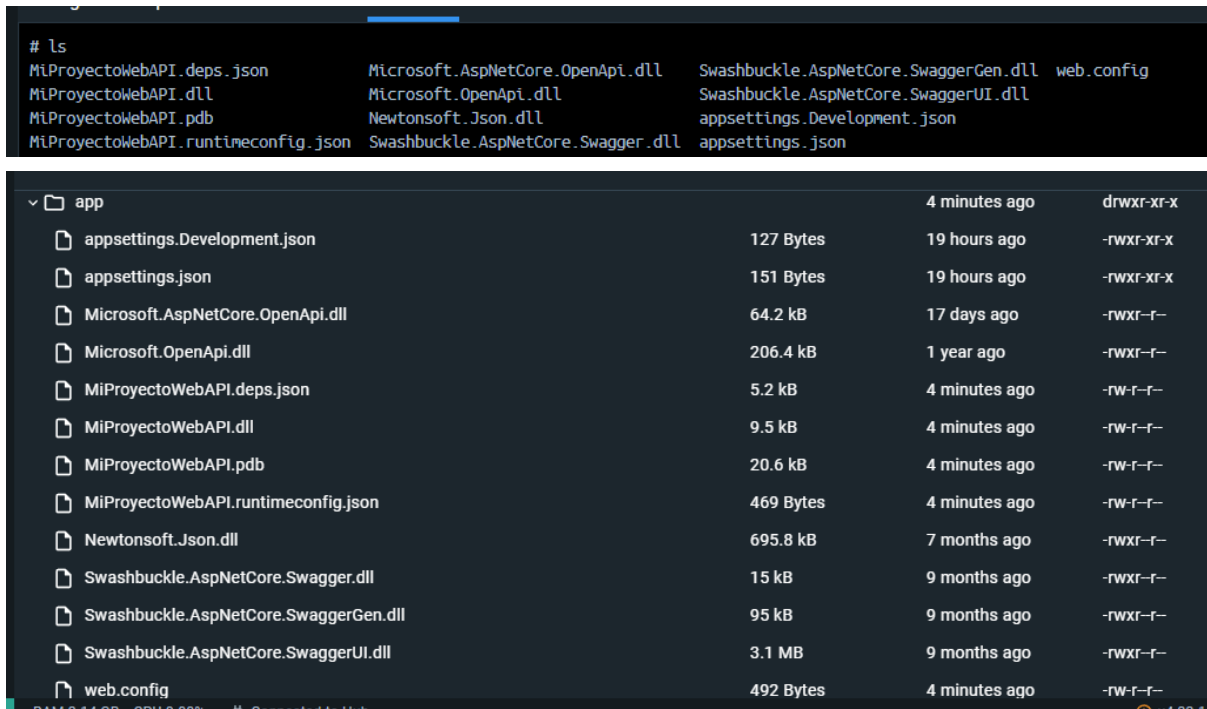
#2 [internal] load build definition from Dockerfile
#2 transferring dockerfile: 599B done
#2 DONE 0.0s

#3 [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0
#3 DONE 0.8s

#4 [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:7.0
#4 DONE 0.8s

#5 [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:d0755fbf1ed34aecd79c127fd1dd01b80587acdb8ff
50e5a68d1adf8b076922b
#5 DONE 0.0s
```

Corremos y analizamos los directorios nuevamente



4.4: Imagen para aplicacion web en Nodejs

Generamos un proyecto de Nodejs y creamos un archivo Dockerfile

```

Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6 (main)
$ npx create-react-app my-app

Creating a new React app in C:\Users\Nico\Documents\IngSoft3\trabajo-practico-6\my-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1459 packages in 1m
242 packages are looking for funding
  run 'npm fund' for details

Installing template dependencies using npm...
added 69 packages, and changed 1 package in 6s
246 packages are looking for funding
  run 'npm fund' for details
Removing template package using npm...

removed 1 package, and audited 1528 packages in 4s
246 packages are looking for funding
  run 'npm fund' for details
8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run 'npm audit' for details.

Success! Created my-app at C:\Users\Nico\Documents\IngSoft3\trabajo-practico-6\my-app
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd my-app
  npm start

Happy hacking!

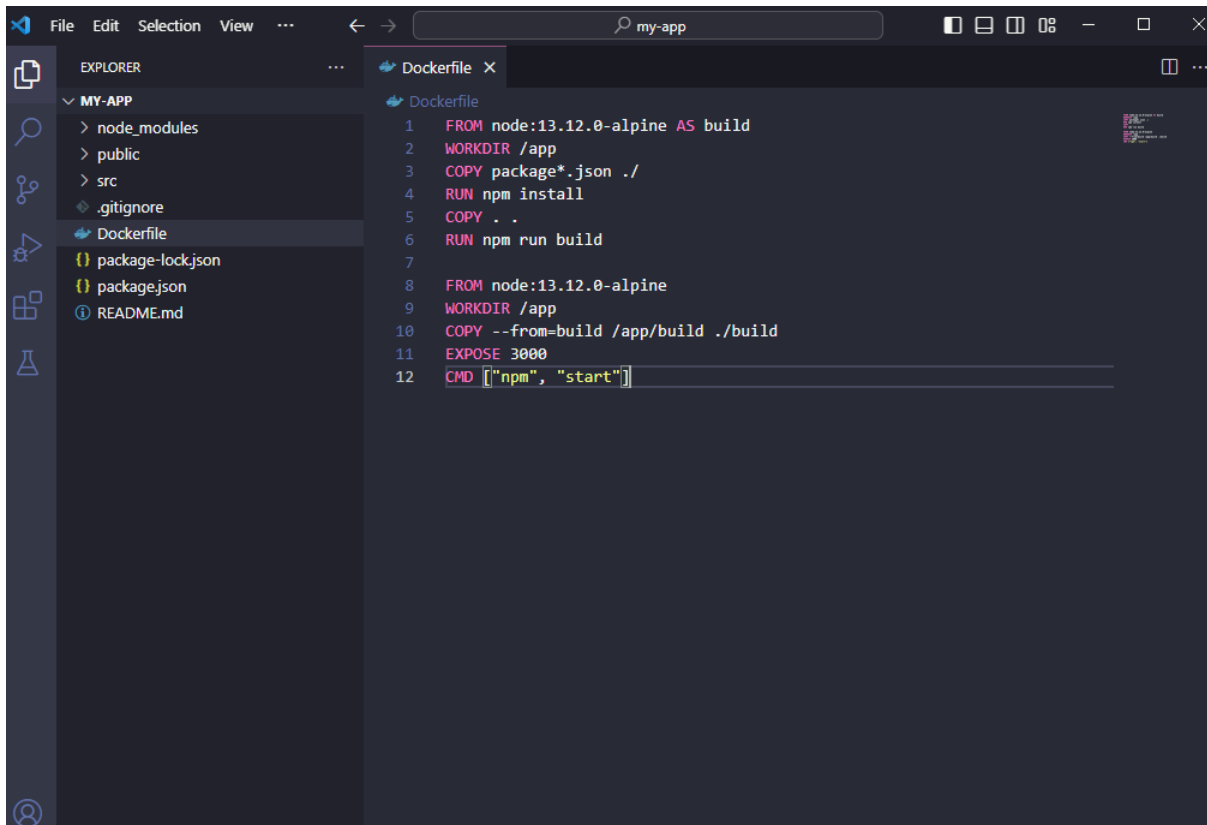
Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6 (main)
$ cd m
MiProyectoWebAPI/ my-app/

Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6 (main)
$ cd my-app/

Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6/my-app (main)
$ touch Dockerfile

```

Escribimos un Dockerfile multistage con una imagen build y otra produccion.



Buildéamos y corremos

```
Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6/my-app (main)
$ docker build -t test-node .
#0 building with "default" instance using docker driver

#1 [internal] load .dockerignore
#1 transferring context: 2B done
#1 DONE 0.0s

#2 [internal] load build definition from Dockerfile
#2 transferring dockerfile: 270B done
#2 DONE 0.0s

#3 [internal] load metadata for docker.io/library/node:13.12.0-alpine
#3 ...

#4 [auth] library/node:pull token for registry-1.docker.io
#4 DONE 0.0s

#3 [internal] load metadata for docker.io/library/node:13.12.0-alpine
#3 DONE 2.8s
```

```
Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6/my-app (main)
$ docker run -p 3000:3000 test-node
```



Edit `src/App.js` and save to reload.

[Learn React](#)

4.5: Publicamos la imagen en Docker

```
Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6/my-app (main)
$ docker login
Authenticating with existing credentials...
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/

Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6/my-app (main)
$ cd ..

Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6 (main)
$ cd my-app/

Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6/my-app (main)
$ docker tag test-node <nicocolman>/test-node:latest
bash: nicocolman: No such file or directory

Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6/my-app (main)
$ docker tag test-node nicocolman3/test-node:latest

Nico@DESKTOP-MDI8QUJ MINGW64 ~/Documents/IngSoft3/trabajo-practico-6/my-app (main)
$ docker push nicocolman3/test-node:latest
The push refers to repository [docker.io/nicocolman3/test-node]
fd59e6faf051: Preparing
b0e875a9b010: Preparing
65d358b7de11: Preparing
f97384e8ccbc: Preparing
d56e5e720148: Preparing
beee9f30bc1f: Preparing
beee9f30bc1f: Waiting
d56e5e720148: Mounted from library/node
f97384e8ccbc: Mounted from library/node
65d358b7de11: Mounted from library/node
b0e875a9b010: Pushed
beee9f30bc1f: Mounted from library/node
fd59e6faf051: Pushed
```