

Trabajo Práctico 6 - Construcción de Imágenes de Docker

1- Describir las instrucciones:

FROM: Esta instrucción se utiliza para especificar la imagen base que se utilizará para construir el contenedor. Establece el punto de partida para la construcción del contenedor.

RUN: Se utiliza para ejecutar comandos dentro del contenedor durante la construcción.

ADD: Se utiliza para copiar archivos y directorios desde el sistema de archivos local a la imagen del contenedor.

COPY: Se utiliza para copiar archivos y directorios desde el sistema de archivos local a la imagen del contenedor, pero se recomienda usar COPY cuando no se necesita la funcionalidad adicional de ADD.

EXPOSE: Indica qué puertos deben exponerse desde el contenedor al host o a otros contenedores.

CMD: Especifica el comando que se ejecutará cuando el contenedor se inicie. Solo debe haber una instrucción CMD en un archivo Dockerfile, y si hay múltiples instrucciones CMD, se utilizará la última.

ENTRYPOINT: Se utiliza para especificar el comando que se ejecutará cuando el contenedor se inicie. Sin embargo, a diferencia de CMD, los argumentos pasados en la línea de comandos al iniciar el contenedor se anexarán al comando ENTRYPOINT en lugar de reemplazarlo.

2- Generar imagen de docker

Utilizar el resultado del paso 1 del TP 5

Agregar un archivo llamado Dockerfile (en el directorio raíz donde se encuentran todos los archivos y directorios)

```
FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base
WORKDIR /app
EXPOSE 80
FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
WORKDIR /src
COPY ["MiProyectoWebAPI.csproj", "."]
RUN dotnet restore "./MiProyectoWebAPI.csproj"
COPY . .
WORKDIR "/src/"
RUN dotnet build "MiProyectoWebAPI.csproj" -c Release -o /app/build
RUN dotnet publish "MiProyectoWebAPI.csproj" -c Release -o /app/publish
/p:UseAppHost=false
ENTRYPOINT ["dotnet", "bin/Debug/net7.0/MiProyectoWebAPI.dll"]
```

3- Generar la imagen de docker con el comando build

docker build -t miproyectowebapi.

Ejecutar el contenedor

docker run -p 8080:80 -it --rm miproyectowebapi

Capturar y mostrar la salida.

```
[agostinamorellato@Agostinas-MacBook-Pro MiProyectoWebAPI % docker build -t miproyectowebapi .
[+] Building 17.2s (13/13) FINISHED                    docker:desktop-linux
=> [internal] load build definition from dockerfile    0.0s
=> => transferring dockerfile: 543B                    0.0s
=> [internal] load .dockerignore                      0.0s
=> => transferring context: 2B                          0.0s
=> [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0 1.4s
=> [internal] load build context                      0.1s
=> => transferring context: 4.80MB                      0.1s
=> [build 1/8] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:926b9337622c 0.0s
=> CACHED [build 2/8] WORKDIR /src                    0.0s
=> [build 3/8] COPY [MiProyectoWebAPI.csproj, .]      0.0s
=> [build 4/8] RUN dotnet restore "./MiProyectoWebAPI.csproj" 12.2s
=> [build 5/8] COPY . .                               0.0s
=> [build 6/8] WORKDIR /src/.                          0.0s
=> [build 7/8] RUN dotnet build "MiProyectoWebAPI.csproj" -c Release -o 2.1s
=> [build 8/8] RUN dotnet publish "MiProyectoWebAPI.csproj" -c Release - 1.1s
=> exporting to image                                0.1s
=> => exporting layers                                0.1s
=> => writing image sha256:a68278053aa0bda117b4943a33c8dd438147265563868 0.0s
=> => naming to docker.io/library/miproyectowebapi    0.0s
```

What's Next?

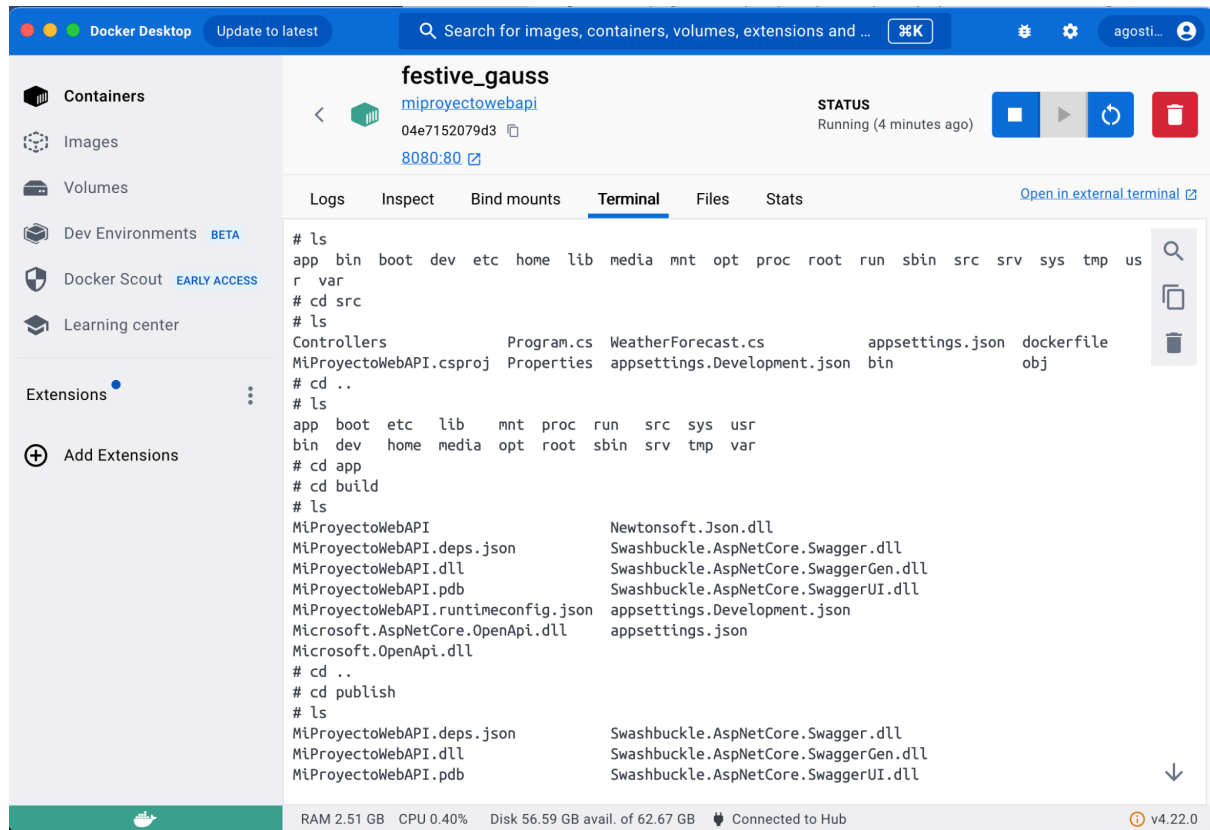
View summary of image vulnerabilities and recommendations → [docker scout quick view](#)

```
[agostinamorellato@Agostinas-MacBook-Pro MiProyectoWebAPI % docker run -p 8080:80 -it --rm miproyectowebapi
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
```

The screenshot shows the Docker Desktop application window. On the left is a sidebar with navigation options: Containers, Images, Volumes, Dev Environments (BETA), Docker Scout (EARLY ACCESS), Learning center, Extensions, and Add Extensions. The main area is titled 'Containers' and displays a table of running and exited containers. At the top, it shows overall system usage: 0.03% / 500% CPU and 19.57MB / 7.49GB memory. A search bar and a toggle for 'Only show running containers' are present. The container table lists four items: 'db' (exited), 'myapi' (exited), 'silly_chebys' (exited), and 'festive_gau' (running). The 'festive_gau' container is highlighted, showing it is running on the 'miproyectowebapi' image, using 0.03% CPU, and has port 8080 mapped to port 8080. The interface is clean and modern, with a blue header bar and a light gray background.

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	db	redis:alpine	Exited (255)	0%		12 days ago	▶ ⋮ 🗑
<input type="checkbox"/>	myapi	mywebapi	Exited	0%	5254:5254	12 days ago	▶ ⋮ 🗑
<input type="checkbox"/>	silly_chebys	mywebapi	Exited	0%		12 days ago	▶ ⋮ 🗑
<input type="checkbox"/>	festive_gau	miproyectowebapi	Running	0.03%	8080:80	52 seconds ago	▶ ⋮ 🗑

4- Entrar a la terminal del contenedor y ver directorios src, app/build y app/publish



5- Modificar el dockerfile para el proyecto anterior de la siguiente forma

FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base

WORKDIR /app

EXPOSE 80

FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build

WORKDIR /src

COPY ["MiProyectoWebAPI.csproj", "."]

RUN dotnet restore "./MiProyectoWebAPI.csproj"

COPY . .

WORKDIR "/src/."

RUN dotnet build "MiProyectoWebAPI.csproj" -c Release -o /app/build

FROM build AS publish

RUN dotnet publish "MiProyectoWebAPI.csproj" -c Release -o /app/publish

/p:UseAppHost=false

FROM base AS final

WORKDIR /app

COPY --from=publish /app/publish .

ENTRYPOINT ["dotnet", "MiProyectoWebAPI.dll"]

6- Construir nuevamente la imagen

`docker build -t miproyectowebapi .`

```
agostinamorellato@Agostinas-MacBook-Pro MiProyectoWebAPI % docker build -t miproyectowebapi .
[+] Building 4.5s (18/18) FINISHED                                docker:desktop-linux
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load build definition from dockerfile              0.0s
=> => transferring dockerfile: 618B                               0.0s
=> [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0 0.9s
=> [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:7.0 1.5s
=> [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:926b9337622c 0.0s
=> [base 1/2] FROM mcr.microsoft.com/dotnet/aspnet:7.0@sha256:54a3864f1c 0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 4.53kB                                0.0s
=> CACHED [build 2/7] WORKDIR /src                              0.0s
=> CACHED [build 3/7] COPY [MiProyectoWebAPI.csproj, .]         0.0s
=> CACHED [build 4/7] RUN dotnet restore "./MiProyectoWebAPI.csproj" 0.0s
=> [build 5/7] COPY . .                                         0.0s
=> [build 6/7] WORKDIR /src/.                                    0.0s
=> [build 7/7] RUN dotnet build "MiProyectoWebAPI.csproj" -c Release -o 1.8s
=> [publish 1/1] RUN dotnet publish "MiProyectoWebAPI.csproj" -c Release 1.1s
=> CACHED [base 2/2] WORKDIR /app                                0.0s
=> CACHED [final 1/2] WORKDIR /app                              0.0s
=> [final 2/2] COPY --from=publish /app/publish .               0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:7f2851ffe4ce5ccc468543f1fa2ebb17d6ea13de238c2 0.0s
=> => naming to docker.io/library/miproyectowebapi              0.0s

What's Next?
View summary of image vulnerabilities and recommendations → docker scout quick view
agostinamorellato@Agostinas-MacBook-Pro MiProyectoWebAPI %
```

7- Analizar y explicar el nuevo Dockerfile, incluyendo las nuevas instrucciones.

```
EXPLORER
OPEN EDITORS
Welcome
x dockerfile
MIPROYECTOWEBAPI
  .vscode
  bin
  Controllers
  obj
  Properties
  appsettings.Development.json
  appsettings.json
  dockerfile
  MiProyectoWebAPI.csproj
  Program.cs
  WeatherForecast.cs
OUTLINE
TIMELINE

Welcome | dockerfile x
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"]
21

Ln 4, Col 1 Spaces: 4 UTF-8 LF D
```

Este Dockerfile construye una imagen de Docker para la aplicación web ASP.NET Core.

Base Image (base):

- **FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base:** Define la primera etapa de la imagen llamada "base", basada en una imagen oficial de ASP.NET Core 7.0, que proporciona el entorno necesario para ejecutar aplicaciones ASP.NET Core.
- **WORKDIR /app:** Establece el directorio base de trabajo para la aplicación en la imagen dentro del contenedor en "/app".

Build Image (build):

- **FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build:** Define la segunda etapa de la imagen llamada "build". Esta etapa se basa en una imagen oficial de SDK de .NET Core 7.0, que proporciona las herramientas necesarias para compilar y publicar la aplicación.
- **WORKDIR /src:** Establece el directorio de trabajo en "/src" en esta etapa.
- **COPY ["MiProyectoWebAPI.csproj", "."]:** Copia el archivo del proyecto al directorio actual en el contenedor.
- **RUN dotnet restore "/MiProyectoWebAPI.csproj":** Ejecuta el comando dotnet restore para restaurar las dependencias del proyecto especificado.
- **COPY . . :** Copia todo el contenido del directorio de construcción local al directorio actual en el contenedor.
- **WORKDIR "/src/.":** Establece el directorio de trabajo en "/src/." en esta etapa.

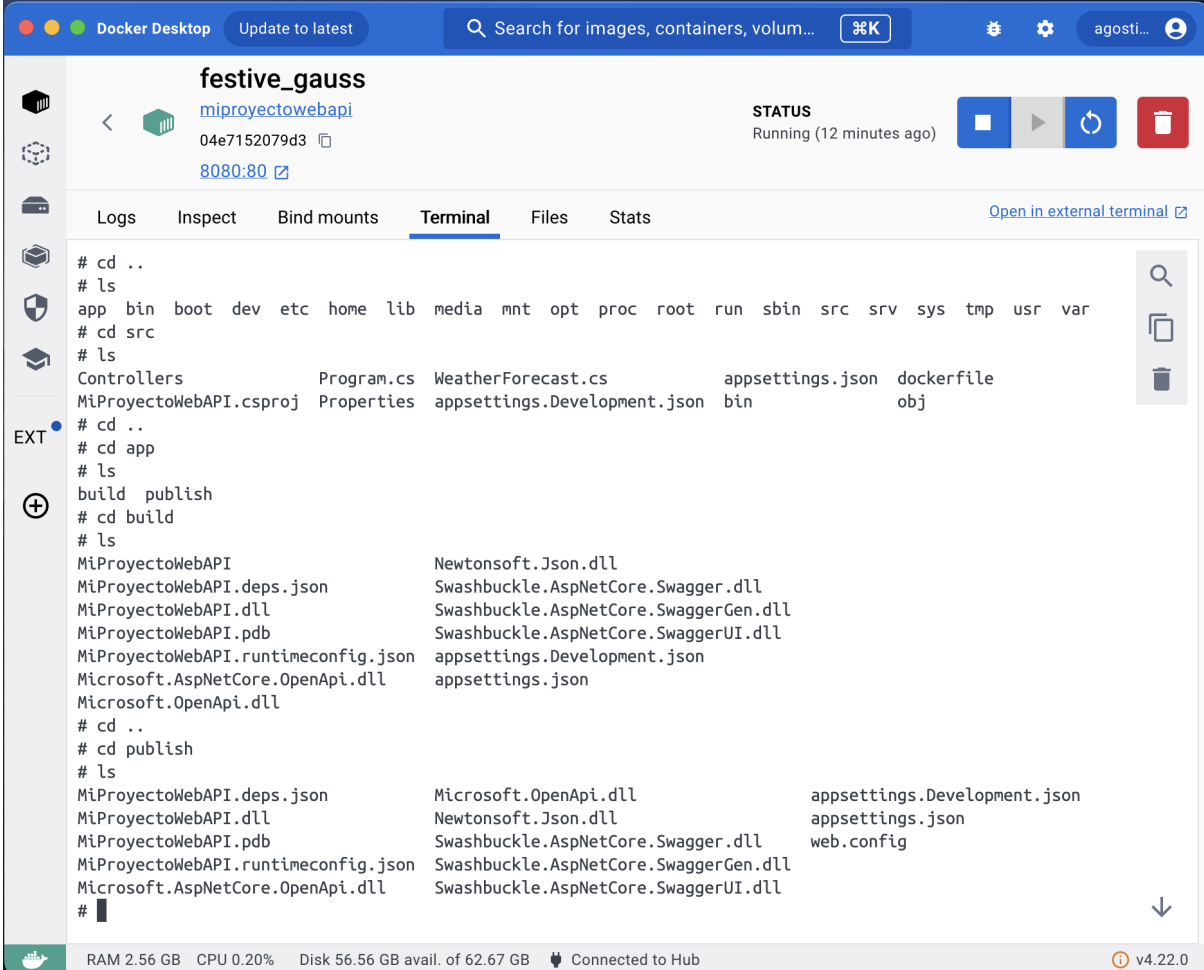
Publish Image (publish):

- **FROM build AS publish:** Define la tercera etapa de la imagen llamada "publish". Se basa en la etapa "build".
- **RUN dotnet publish "MiProyectoWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false:** Ejecuta el comando dotnet publish para publicar la aplicación en modo Release en el directorio "/app/publish" y deshabilita el uso de AppHost.

Final Image (final):

- **FROM base AS final:** Define la cuarta etapa de la imagen llamada "final". Se basa en la etapa "base".
- **WORKDIR /app:** Establece el directorio de trabajo en "/app".
- **COPY --from=publish /app/publish .:** Copia el resultado de la etapa "publish" (los archivos publicados) desde la imagen de "publish" al directorio actual en esta etapa.
- **ENTRYPOINT ["dotnet", "MiProyectoWebAPI.dll"]:** Establece el comando de entrada para el contenedor, que inicia la aplicación ASP.NET Core cuando se ejecuta el contenedor.

8- Entrar a la terminal del contenedor y ver directorios src, app/build y app/publish



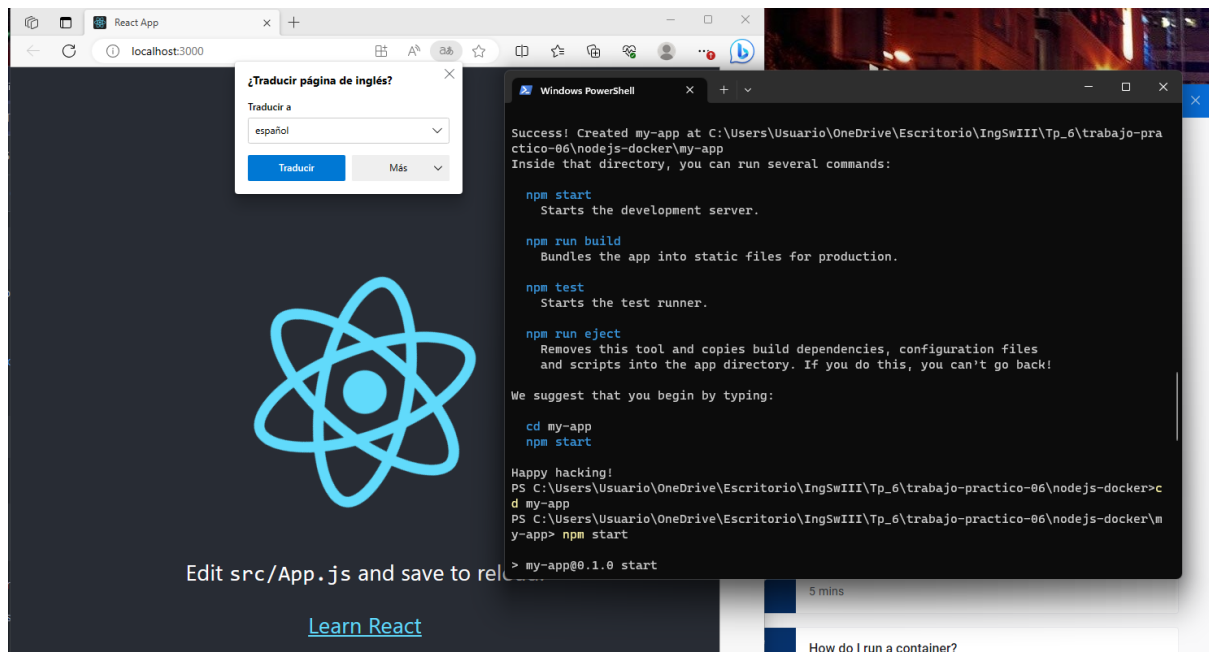
The screenshot shows the Docker Desktop interface. At the top, there's a search bar and a status bar. The main area displays the container 'festive_gauss' with its ID '04e7152079d3' and status 'Running (12 minutes ago)'. Below this, there are tabs for 'Logs', 'Inspect', 'Bind mounts', 'Terminal', 'Files', and 'Stats'. The 'Terminal' tab is active, showing a command prompt where the user has navigated through the directory structure: root, src, app, build, and publish. The terminal output shows the contents of each directory as listed by the 'ls' command. The status bar at the bottom indicates system resources: RAM 2.56 GB, CPU 0.20%, Disk 56.56 GB avail. of 62.67 GB, and a connection to Hub.

```
# cd ..
# ls
app bin boot dev etc home lib media mnt opt proc root run sbin src srv sys tmp usr var
# cd src
# ls
Controllers Program.cs WeatherForecast.cs appsettings.json dockerfile
MiProyectoWebAPI.csproj Properties appsettings.Development.json bin obj
# cd ..
# cd app
# ls
build publish
# cd build
# ls
MiProyectoWebAPI Newtonsoft.Json.dll
MiProyectoWebAPI.deps.json Swashbuckle.AspNetCore.Swagger.dll
MiProyectoWebAPI.dll Swashbuckle.AspNetCore.SwaggerGen.dll
MiProyectoWebAPI.pdb Swashbuckle.AspNetCore.SwaggerUI.dll
MiProyectoWebAPI.runtimeconfig.json appsettings.Development.json
Microsoft.AspNetCore.OpenApi.dll appsettings.json
Microsoft.OpenApi.dll
# cd ..
# cd publish
# ls
MiProyectoWebAPI.deps.json Microsoft.OpenApi.dll appsettings.Development.json
MiProyectoWebAPI.dll Newtonsoft.Json.dll appsettings.json
MiProyectoWebAPI.pdb Swashbuckle.AspNetCore.Swagger.dll web.config
MiProyectoWebAPI.runtimeconfig.json Swashbuckle.AspNetCore.SwaggerGen.dll
Microsoft.AspNetCore.OpenApi.dll Swashbuckle.AspNetCore.SwaggerUI.dll
#
```

4- Imagen para aplicación web en Nodejs

Crear una la carpeta **trabajo-practico-06/nodejs-docker**

Generar un proyecto siguiendo los pasos descritos en el trabajo práctico 5 para Nodejs



Escribir un Dockerfile para ejecutar la aplicación web localizada en ese directorio

- Idealmente que sea multistage, con una imagen de build y otra de producción.
- Usar como imagen base `node:13.12.0-alpine`
- Ejecutar `npm install` dentro durante el build.
- Exponer el puerto 3000

```
File Edit Selection View ... microservices-demo
Dockerfile.dockerfile
C: > Users > Usuario > OneDrive > Escritorio > IngSwIII > Tp_6 > trabajo-practico-06 > nodejs-docker >
1 # Etapa de construcción (Build Stage)
2 FROM node:13.12.0-alpine AS build
3
4 WORKDIR /app
5
6 COPY package*.json ./
7
8 RUN npm install
9 COPY . .
10
11 # Etapa de producción
12 FROM node:13.12.0-alpine AS production
13 WORKDIR /app
14 COPY --from=build /app ./
15
16 # Exponer el puerto 3000 en el contenedor
17 EXPOSE 3000
18
19 # Iniciar la aplicación en modo de producción
20 CMD ["npm", "start"]
21
```

Hacer un build de la imagen, nombrar la imagen test-node.

`docker build -t test-node`

```

PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_6\trabajo-practico-06\nodejs-docker\my-app> docker build
-t test-node .
[+] Building 236.9s (12/12) FINISHED                                docker:default
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                     0.0s
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 409B                                 0.0s
=> [internal] load metadata for docker.io/library/node:13.12.0-alpine 3.4s
=> [auth] library/node:pull token for registry-1.docker.io        0.0s
=> [internal] load build context                                  147.6s
=> => transferring context: 298.43MB                               147.5s
=> [build 1/5] FROM docker.io/library/node:13.12.0-alpine@sha256:cc85e728fab3827ada20a181ba280cae1f 11.3s
=> => resolve docker.io/library/node:13.12.0-alpine@sha256:cc85e728fab3827ada20a181ba280cae1f8b625f2 0.0s
=> => sha256:483343d6c5f5d658963b7c16e4299247f765bef4025012457ee105481ea1afc1 6.77kB / 6.77kB 0.0s
=> => sha256:aad63a9339440e7c3e1fff2b988991b9bfb81280042fa7f39a5e327023056819 2.80MB / 2.80MB 1.3s
=> => sha256:a00bd932208e2de29cd2b7e0bab954325913d146401effb482fff3d8775aaab 35.29MB / 35.29MB 3.6s
=> => sha256:c57f2c59b93778192e60e0e213949630f9ad30324736bbb0a71e12adf8a16d46 2.24MB / 2.24MB 0.8s
=> => sha256:cc85e728fab3827ada20a181ba280cae1f8b625f256e2c86b9094d9bfe834766 1.19kB / 1.19kB 0.0s
=> => sha256:ed06820d0fb6f4711e0a6f50c9f147fb2596399866319e1bb3b0a52393c5615f 1.16kB / 1.16kB 0.0s
=> => sha256:f3446470f297e51c1e27f90f7ae9a94f1ee7b6c8e529aec83aa1a04ad70531c0 284B / 284B 1.3s
=> => extracting sha256:aad63a9339440e7c3e1fff2b988991b9bfb81280042fa7f39a5e327023056819 0.4s
=> => extracting sha256:a00bd932208e2de29cd2b7e0bab954325913d146401effb482fff3d8775aaab 7.0s
=> => extracting sha256:c57f2c59b93778192e60e0e213949630f9ad30324736bbb0a71e12adf8a16d46 0.1s
=> => extracting sha256:f3446470f297e51c1e27f90f7ae9a94f1ee7b6c8e529aec83aa1a04ad70531c0 0.0s
=> [build 2/5] WORKDIR /app                                       0.3s
=> [build 3/5] COPY package*.json ./                               0.3s
=> [build 4/5] RUN npm install                                    67.4s
=> [build 5/5] COPY . .                                           5.9s
=> [stage-1 3/3] COPY --from=build /app ./                         3.8s
=> exporting to image                                              5.6s
=> => exporting layers                                           5.6s
=> => writing image sha256:65364863d72454b975b1eec396ff4b2ae62f4854f1551dd799258f918fac0a47 0.0s
=> => naming to docker.io/library/test-node                       0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview

```

Ejecutar la imagen test-node publicando el puerto 3000.

docker run -p 3000:3000 test-node

```

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_6\trabajo-practico-06\nodejs-docker\my-app> docker run -
p 3000:3000 test-node

> my-app@0.1.0 start /app
> react-scripts start

(node:30) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' op
tion is deprecated. Please use the 'setupMiddlewares' option.
(node:30) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware'
option is deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...

One of your dependencies, babel-preset-react-app, is importing the
"@babel/plugin-proposal-private-property-in-object" package without
declaring it in its dependencies. This is currently working because
"@babel/plugin-proposal-private-property-in-object" is already in your
node_modules folder for unrelated reasons, but it may break at any time.

babel-preset-react-app is part of the create-react-app project, which
is not maintained anymore. It is thus unlikely that this bug will
ever be fixed. Add "@babel/plugin-proposal-private-property-in-object" to
your devDependencies to work around this error. This will make this message
go away.

Compiled successfully!

You can now view my-app in the browser.

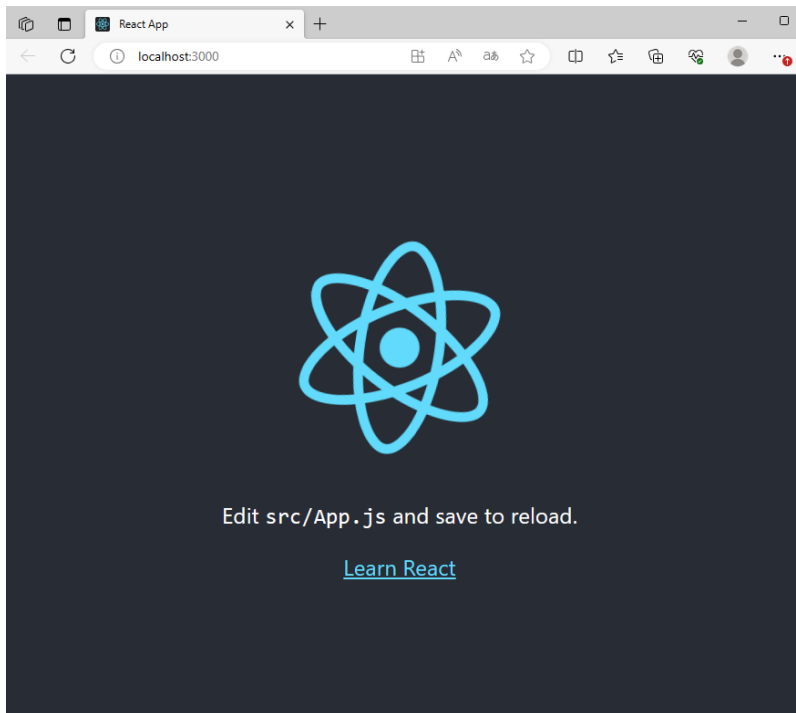
Local:            http://localhost:3000
On Your Network:  http://172.17.0.2:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
Compiling...
Compiled successfully!
webpack compiled successfully

```

Verificar en <http://localhost:3000> que la aplicación está funcionando.



5- Publicar la imagen en Docker Hub.

Crear una cuenta en Docker Hub si no se dispone de una.

Registrarse localmente a la cuenta de Docker Hub:

docker login

Crear un tag de la imagen generada en el ejercicio 3. Reemplazar <mi_usuario> por el creado en el punto anterior.

docker tag test-node agostinamorellato/test-node:latest

Subir la imagen a Docker Hub con el comando

docker push agostinamorellato/test-node:latest

Como resultado de este ejercicio mostrar la salida de consola, o una captura de pantalla de la imagen disponible en Docker Hub.

```
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_6\trabajo-practico-06\nodejs-docker\my-app> docker login
Authenticating with existing credentials...
Login Succeeded
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_6\trabajo-practico-06\nodejs-docker\my-app> d
>
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_6\trabajo-practico-06\nodejs-docker\my-app> docker tag t
est-node agostinamorellato/test-node:latest
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_6\trabajo-practico-06\nodejs-docker\my-app> docker push
agostinamorellato/test-node:latest
The push refers to repository [docker.io/agostinamorellato/test-node]
4815e92f75bf: Pushed
53c0c5b58271: Pushed
65d358b7de11: Mounted from library/node
f97384e8ccbc: Mounted from library/node
d56e5e720148: Mounted from library/node
beee9f30bc1f: Mounted from library/node
latest: digest: sha256:0bb935958f75a759aba99708af7e86daf3e77f42814f1eba4e46084e64dfdefd size: 1576
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_6\trabajo-practico-06\nodejs-docker\my-app> |
```

