1)

FROM

Un Dockerfile debe comenzar con una instrucción FROM.

Lo que hace es inicializar un nuevo escenario de build y setear la base image para las siguientes instrucciones.

FROM puede aparecer varias veces en un único dockerfile para crear múltiples imágenes o usar un escenario de build como dependencia de otro.

La flag --platform puede setearse si FROM referencia una imagen multiplataforma.

RUN

Esta instrucción ejecuta cualquier comando en una nueva capa en el tope (parte superior) de la imagen actual y commiteará los resultados para que sean usados en el siguiente paso.

Tiene dos formas:

-shell: RUN <command> (el comando correrá en el shell).

-exec: RUN ["executable, "param1", "param2"].

ADD

Copia nuevos archivos, directorios o archivos remoto desde una fuente <src> y los añade a el filesystem de la imagen en la ruta destino <dest>. <src> debe estar dentro del context del build.

Tiene dos formas:

-ADD [--chown=<user>:<group>] <src>... <dest>.

-ADD [--chown=<user>:<group>] ["<src>",... "<dest>"].

COPY

Copia nuevos archivos o directorios desde el path fuente <src> y los añade al filesystem del container en el path destino <dest>. Misma finalidad que ADD pero soporta únicamente copiar desde un directorio local del host y no ofrece la funcionalidad de desempaquetar el contenido desde un src comprimido.

Tiene dos formas:

-COPY [--chown=<user>:<group>] <src>... <dest>.

-COPY [--chown=<user>:<group>] ["<src>",... "<dest>"].

EXPOSE [/...]

Informa que el container escucha en un puerto de la red específico en tiempo de ejecución. No es que publica en el puerto, funciona como un tipo de documentación entre la persona que construye la imagen y la que ejecuta el container.

CMD

Tiene tres formas:

-shell: CMD command param1 param2.

-exec: CMD ["executable","param1","param2"].

-default parameters to entrypoint: CMD ["param1","param2"].

Solo puede haber una en un Dockerfile, su propósito es proveer un contenedor de ejecución por defecto, puede ser incluido u omitido del ejecutable, en el último caso se debe especificar un ENTRYPOINT.

ENTRYPOINT

Esta instrucción te permite configurar el contenedor que va a correr como un ejecutable.

Tiene dos formas:

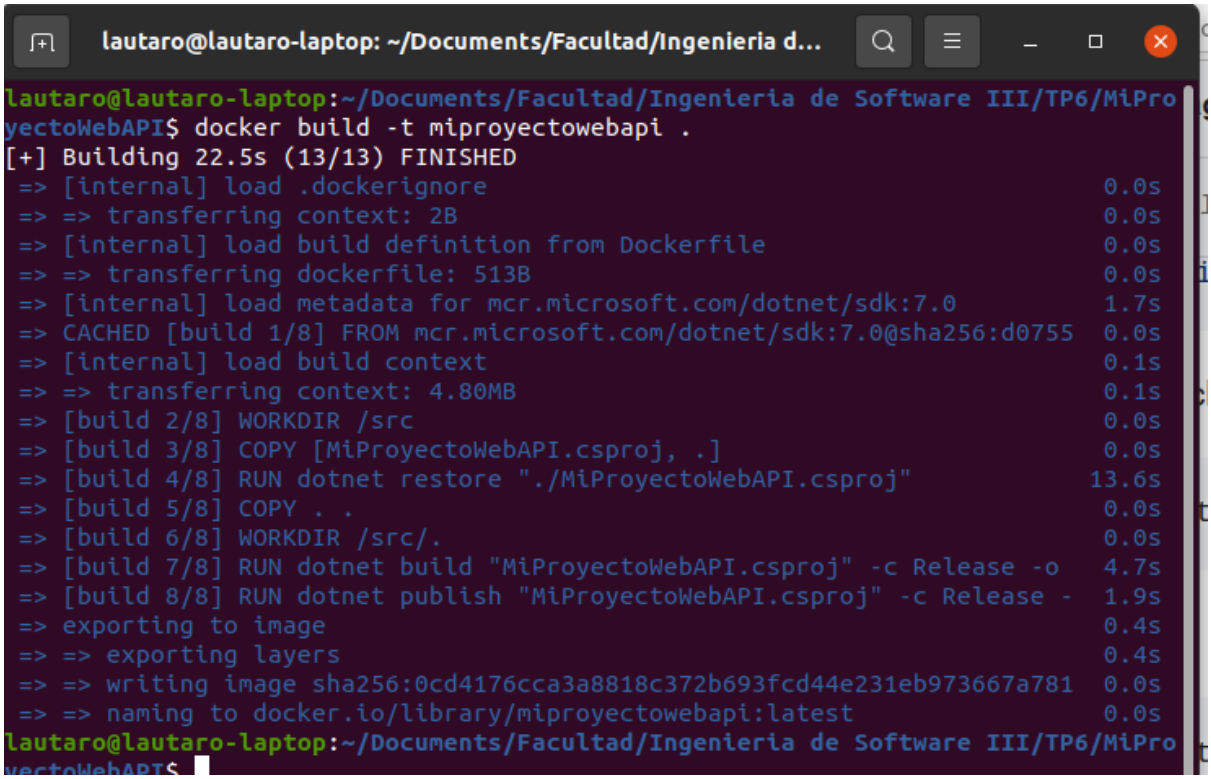
-shell: ENTRYPOINT command param1 param2.

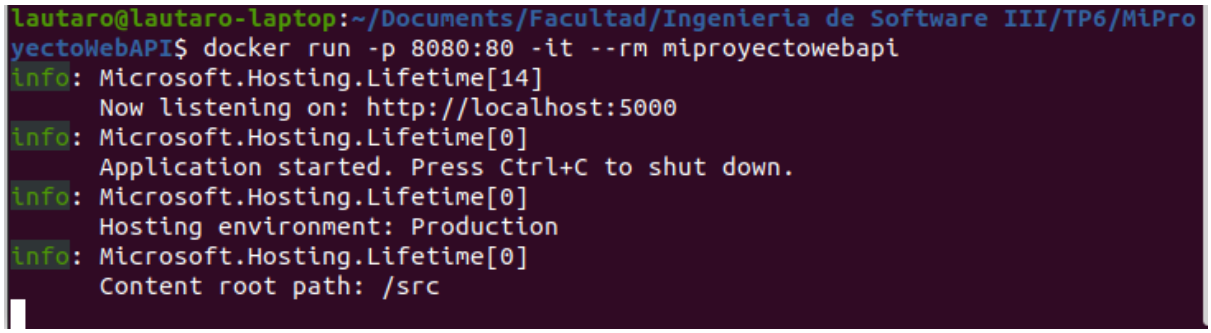
-exec: RUN ["executable, "param1", "param2"].

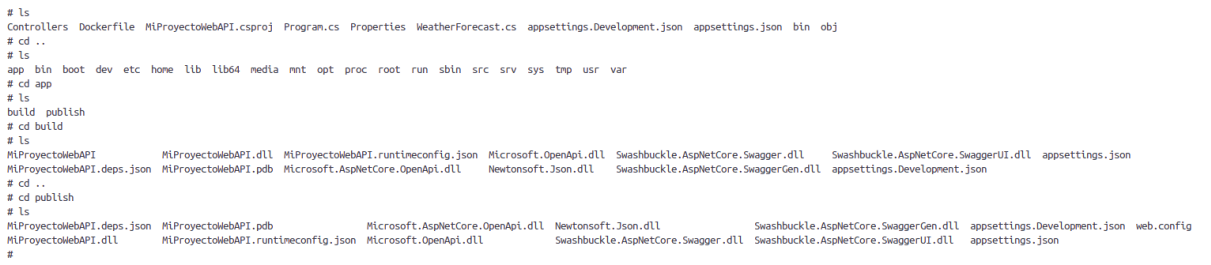
Los argumentos pasados en el comando run (en la línea de comandos) serán agregados después de todos los elementos en un exec desde el ENTRYPOINT. Esto permite pasar los elementos al entry point.

El ENTRYPOINT especifica el ejecutable que usará el contenedor y CMD se corresponde con los parámetros a usar con dicho ejecutable.

2)







3)

FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base: Esta línea establece la imagen base para esta etapa del proceso de construcción. En este caso, está utilizando una imagen oficial de Microsoft con ASP.NET en la versión 7.0. Esta imagen ya tiene el entorno necesario para ejecutar aplicaciones ASP.NET.

WORKDIR /app: Esta línea establece el directorio de trabajo dentro del contenedor como "/app". Todos los comandos posteriores se ejecutarán en este directorio.

EXPOSE 80: Indica que el contenedor va a escuchar en el puerto 80. Esto no implica que el puerto 80 del host se vincule con el contenedor, simplemente es una información para quien esté usando el contenedor.

FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build: Esta línea establece otra etapa de construcción usando una imagen SDK de .NET en la versión 7.0. Esta etapa se utiliza para compilar la aplicación.

WORKDIR /src: Establece el directorio de trabajo en la etapa de construcción como "/src".

COPY ["MiProyectoWebAPI.csproj", "."]: Copia el archivo de proyecto ("MiProyectoWebAPI.csproj") al directorio de trabajo actual en la etapa de construcción.

RUN dotnet restore "./MiProyectoWebAPI.csproj": Ejecuta el comando dotnet restore para restaurar las dependencias del proyecto.

COPY . . : Copia todos los archivos del contexto del Dockerfile (todos los archivos en el mismo directorio que el Dockerfile) al directorio de trabajo en la etapa de construcción.

WORKDIR "/src/.": Cambia el directorio de trabajo a la raíz del proyecto en la etapa de construcción.

RUN dotnet build "MiProyectoWebAPI.csproj" -c Release -o /app/build: Compila el proyecto en modo Release y coloca los archivos de salida en el directorio "/app/build".

FROM build AS publish: Establece otra etapa llamada "publish" basada en la etapa de construcción anterior.

RUN dotnet publish "MiProyectoWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false: Publica la aplicación compilada en el directorio "/app/publish".

FROM base AS final: Establece la última etapa llamada "final" basada en la etapa base.

WORKDIR /app: Cambia el directorio de trabajo a "/app" en la última etapa.

C**OPY --from=publish /app/publish .:** Copia los archivos publicados desde la etapa "publish" al directorio de trabajo en la última etapa.

ENTRYPOINT ["dotnet", "MiProyectoWebAPI.dll"]: Define el comando de entrada cuando se inicia un contenedor basado en esta imagen. En este caso, ejecuta el archivo "MiProyectoWebAPI.dll" utilizando el entorno de ejecución de .NET.

4)

```dockerfile
FROM node:13.12.0-alpine as dependencies
COPY ./package.json /tmp/
WORKDIR /tmp/
RUN npm install

FROM dependencies
COPY --from=dependencies /tmp/node_modules ./node_modules
COPY . .
EXPOSE 3000
CMD npm run start
```

```
lautaro@lautaro-laptop:~/Documents/Facultad/Ingenieria de Software III/TP6/tp07-react-ing3$ docker build -t mi-app .
[+] Building 238.7s (11/11) FINISHED
 => [internal] load build definition from Dockerfile                                                                                                0.0s
 => => transferring dockerfile: 257B                                                                                                                0.0s
 => [internal] load .dockerignore                                                                                                                   0.0s
 => => transferring context: 2B                                                                                                                     0.0s
 => [internal] load metadata for docker.io/library/node:13.12.0-alpine                                                                              1.1s
 => [dependencies 1/4] FROM docker.io/library/node:13.12.0-alpine@sha256:cc85e728fab3827ada20a181ba280cae1f8b625f256e2c86b9094d9bfe834766          36.5s
 => => resolve docker.io/library/node:13.12.0-alpine@sha256:cc85e728fab3827ada20a181ba280cae1f8b625f256e2c86b9094d9bfe834766                        0.0s
 => => sha256:cc85e728fab3827ada20a181ba280cae1f8b625f256e2c86b9094d9bfe834766 1.19kB / 1.19kB                                                      0.0s
 => => sha256:ed06820d0fb6f4711e0a6f50c9f147fb259639986319e1bb3b0a52393c5615f 1.16kB / 1.16kB                                                       0.0s
 => => sha256:483343d6c5f5d658963b7c16e4299247f765bef4025012457ee105481ea1afc1 6.77kB / 6.77kB                                                      0.0s
 => => sha256:aad63a9339440e7c3e1fff2b988991b9bfb81280042fa7f39a5e327023056819 2.80MB / 2.80MB                                                      7.0s
 => => sha256:a00bd932208e2de29cd2b7e0bab954325913d146401effb482fff3d8775aaaeb 35.29MB / 35.29MB                                                   34.3s
 => => sha256:c57f2c59b93778192e60e0e213949630f9ad3032473 6bbb0a71e12adf8a16d46 2.24MB / 2.24MB                                                     4.0s
 => => sha256:f3446470f297e51c1e27f90f7ae9a94f1ee7b6c8e529aec83aa1a04ad70531c0 284B / 284B                                                          4.2s
 => => extracting sha256:aad63a9339440e7c3e1fff2b988991b9bfb81280042fa7f39a5e327023056819                                                           0.1s
 => => extracting sha256:a00bd932208e2de29cd2b7e0bab954325913d146401effb482fff3d8775aaaeb                                                           1.9s
 => => extracting sha256:c57f2c59b93778192e60e0e213949630f9ad3032473 6bbb0a71e12adf8a16d46                                                          0.1s
 => => extracting sha256:f3446470f297e51c1e27f90f7ae9a94f1ee7b6c8e529aec83aa1a04ad70531c0                                                           0.0s
 => [internal] load build context                                                                                                                   1.2s
 => => transferring context: 3.32MB                                                                                                                 1.2s
 => [dependencies 2/4] COPY ./package.json /tmp/                                                                                                    0.2s
 => [dependencies 3/4] WORKDIR /tmp/                                                                                                                0.0s
 => [dependencies 4/4] RUN npm install                                                                                                            165.3s
 => [stage-1 1/2] COPY --from=dependencies /tmp/node_modules ./node_modules                                                                         7.8s
 => [stage-1 2/2] COPY . .                                                                                                                         10.6s
 => exporting to image                                                                                                                             17.1s
 => => exporting layers                                                                                                                            17.0s
 => => writing image sha256:3188a61a0980398ad6e765f2dc8157c578a938f2caf4763a5004457ca9696726                                                        0.0s
 => => naming to docker.io/library/mi-app:latest                                                                                                    0.1s
lautaro@lautaro-laptop:~/Documents/Facultad/Ingenieria de Software III/TP6/tp07-react-ing3$
```

Cambie de nombre la imagen porque me equivoque:

```
docker tag mi-app test-node
```

```
lautaro@lautaro-laptop:~/Documents/Facultad/Ingenieria de Software III/TP6/tp07-react-ing3$ docker run -p 3000:3000 test-node

> mi-app@0.1.0 start /tmp
> react-scripts start

(node:25) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
(node:25) [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 mi-app in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://172.17.0.3:3000

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
```

5)

```
lautaro@lautaro-laptop:~/Documents/Facultad/Ingenieria de Software III/TP6/tp07-react-ing3$ docker tag test-node lautarose/test-node:latest
lautaro@lautaro-laptop:~/Documents/Facultad/Ingenieria de Software III/TP6/tp07-react-ing3$ docker push lautarose/test-node:latest
The push refers to repository [docker.io/lautarose/test-node]
cf7269f6e00f: Pushed
af30e48537b2: Pushed
7653ea10f01a: Pushed
5f70bf18a086: Mounted from lautarose/simple-web-api-gh
645d19332ff6: Pushed
65d358b7de11: Mounted from library/node
f97384e8ccbc: Mounted from library/node
d56e5e720148: Mounted from library/node
beee9f30bc1f: Mounted from library/node
latest: digest: sha256:0f455f8e6fa4a16dba97a45b788ac220439e9b78a2d5cea847587f2cfbd33e8b size: 2208
```