**3- Dockerfiles Multi Etapas**

Se recomienda crear compilaciones de varias etapas para todas las aplicaciones (incluso las heredadas). En resumen, las compilaciones de múltiples etapas:

* Son independientes y auto descriptibles
* Resultan en una imagen de Docker muy pequeña
* Puede ser construido fácilmente por todas las partes interesadas del proyecto (incluso los no desarrolladores)
* Son muy fáciles de entender y mantener.
* No requiere un entorno de desarrollo (aparte del código fuente en sí)
* Se puede empaquetar con pipelines muy simples

Las compilaciones de múltiples etapas también son esenciales en organizaciones que emplean múltiples lenguajes de programación. La facilidad de crear una imagen de Docker por cualquier persona sin la necesidad de JDK / Node / Python / etc. no puede ser sobrestimado.

* 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"]

* Construir nuevamente la imagen

docker build -t miproyectowebapi .

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

La principal diferencia entre los dos Dockerfiles es cómo manejan la construcción en etapas y la creación de imágenes intermedias:

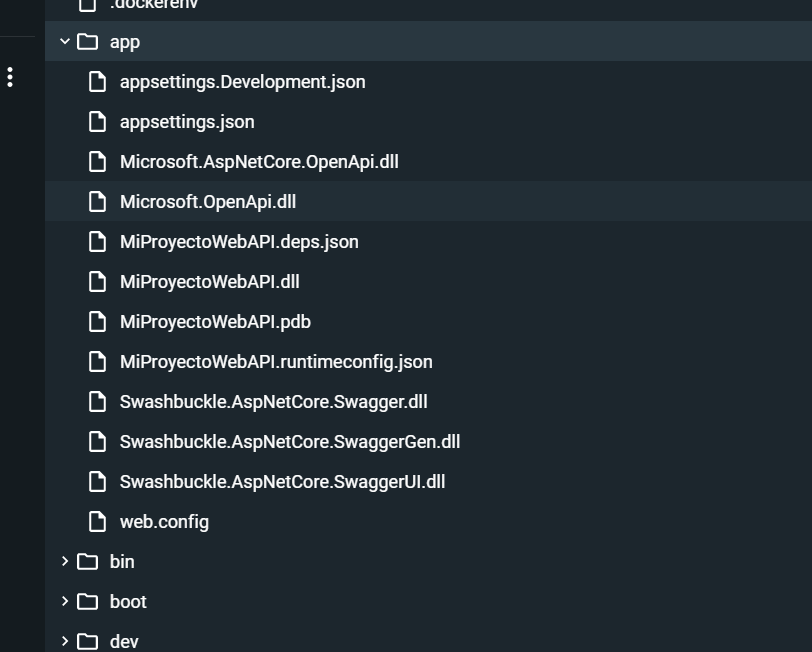
En el primer Dockerfile, todas las instrucciones se ejecutan en un solo bloque de construcción. Primero se realiza la construcción y publicación en la misma etapa, y el punto de entrada se establece en la ruta específica del archivo generado.

En el segundo Dockerfile, se utilizan múltiples etapas de construcción definidas con etiquetas (como AS build, AS publish, AS final). Esto permite una construcción más organizada y modular. La etapa de construcción se divide en dos partes: una para compilar y otra para publicar. Luego, en la etapa final, se copian los archivos publicados desde la etapa "publish" a la imagen final.

La ventaja del segundo enfoque es que puede generar imágenes más pequeñas y eficientes al desechar capas innecesarias en las etapas intermedias. También hace que el proceso de construcción sea más legible y mantenible, ya que cada etapa se enfoca en una tarea específica.

En resumen, ambos Dockerfiles logran el mismo resultado final, pero el segundo enfoque es más modular y se considera una práctica recomendada para la construcción de imágenes Docker más eficientes y mantenibles.

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



Src no existe