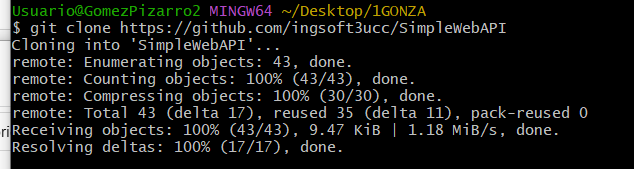
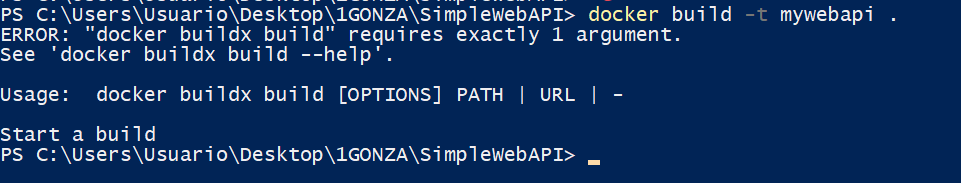
* A partir del código <https://github.com/ingsoft3ucc/SimpleWebAPI> crearemos una imagen.
* Clonar repo

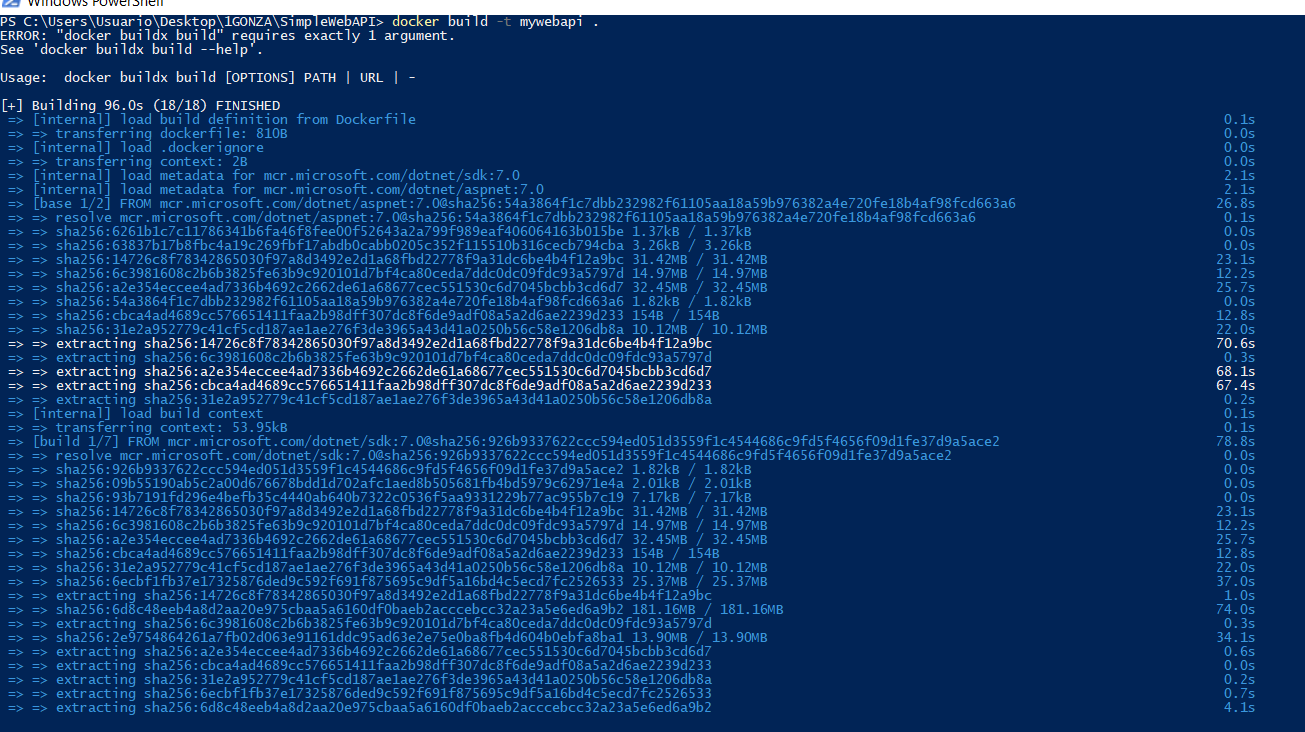


* Crear imagen etiquetándola con un nombre. El punto final le indica a Docker que use el dir actual

docker build –t mywebapi .

ERROR!!!!!!!



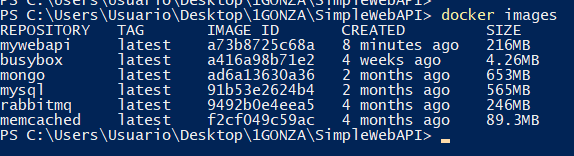
al final corrió, no se por que  
  


* Revisar Dockerfile y explicar cada línea

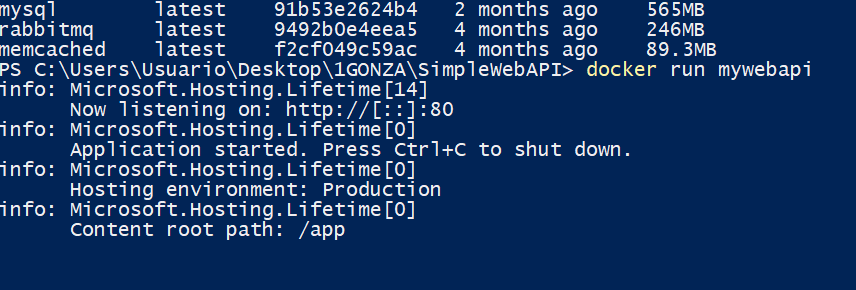
1. **FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base**: Esta línea establece la imagen base para esta etapa con la versión 7.0 de ASP.NET. Se crea una etapa con el nombre **base**.
2. **WORKDIR /app**: Define el directorio de trabajo dentro del contenedor como **/app**.
3. **EXPOSE 80**, **EXPOSE 443**, **EXPOSE 5254**: Estas líneas exponen los puertos 80, 443 y 5254 en el contenedor, lo que permite que las conexiones entrantes se enruten a estos puertos. Sin embargo, es importante tener en cuenta que esto solo declara la exposición, no realiza el mapeo de puertos entre el host y el contenedor.
4. **FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build**: Se define una nueva etapa con la imagen base que contiene el SDK de .NET Core 7.0. Esta etapa se llama **build**.
5. **WORKDIR /src**: Establece el directorio de trabajo dentro del contenedor para la etapa de construcción.
6. **COPY ["SimpleWebAPI/SimpleWebAPI.csproj", "SimpleWebAPI/"]**: Copia el archivo de proyecto **.csproj** de la carpeta **SimpleWebAPI** en el directorio de trabajo de la etapa de construcción.
7. **RUN dotnet restore "SimpleWebAPI/SimpleWebAPI.csproj"**: Ejecuta el comando **dotnet restore** para restaurar las dependencias de la aplicación.
8. **COPY . .**: Copia todo el contenido del directorio actual (del host) en el directorio de trabajo de la etapa de construcción.
9. **WORKDIR "/src/SimpleWebAPI"**: Cambia el directorio de trabajo dentro del contenedor a la carpeta de la aplicación.
10. **RUN dotnet build "SimpleWebAPI.csproj" -c Release -o /app/build**: Compila la aplicación utilizando el comando **dotnet build** y coloca los archivos compilados en la carpeta **/app/build**.
11. **FROM build AS publish**: Define una nueva etapa llamada **publish** basada en la etapa de construcción anterior.
12. **RUN dotnet publish "SimpleWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false**: Publica la aplicación utilizando el comando **dotnet publish** con el perfil de **Release** y deshabilita la generación del host de aplicación.
13. **FROM base AS final**: Define una etapa final basada en la etapa **base**.
14. **WORKDIR /app**: Cambia el directorio de trabajo dentro del contenedor nuevamente a **/app**.
15. **COPY --from=publish /app/publish .**: Copia los archivos publicados desde la etapa **publish** al directorio de trabajo actual.
16. **ENTRYPOINT ["dotnet", "SimpleWebAPI.dll"]**: Establece el comando de inicio cuando se ejecuta el contenedor. En este caso, ejecuta la aplicación ASP.NET usando el archivo **SimpleWebAPI.dll**.
17. **#CMD ["/bin/bash"]**: Esta línea está comentada. En muchos sistemas operativos basados en Linux, el comando **/bin/bash** se usa para ejecutar un shell interactivo en el contenedor, pero en este caso no se está utilizando.

En resumen, este Dockerfile crea una imagen Docker que construye, publica y ejecuta una aplicación ASP.NET en un contenedor. Las diferentes etapas se utilizan para organizar las tareas de construcción y separar las dependencias de tiempo de ejecución de la aplicación final.

* Ver imágenes disponibles



* Ejecutar un contenedor con nuestra imagen



* Subir imagen a nuestra cuenta de dockerhub

