

1. **FROM:** La instrucción **FROM** se utiliza para especificar la imagen base a partir de la cual se creará una nueva imagen en Docker. Es la primera instrucción en un archivo Dockerfile y establece la base sobre la cual se construirá tu contenedor.
2. **RUN:** La instrucción **RUN** permite ejecutar comandos en el sistema operativo dentro del contenedor Docker durante la construcción de la imagen. Puedes utilizar **RUN** para instalar software, configurar el entorno o realizar otras tareas necesarias para preparar la imagen.
3. **ADD:** La instrucción **ADD** se utiliza para copiar archivos y directorios desde el sistema de archivos del host a la imagen de Docker durante la construcción de la imagen. Puedes usar **ADD** para agregar archivos, como aplicaciones, scripts o recursos, a la imagen.
4. **COPY:** La instrucción **COPY** es similar a **ADD**, pero se utiliza específicamente para copiar archivos y directorios desde el sistema de archivos del host a la imagen de Docker. A diferencia de **ADD**, **COPY** no realiza ninguna extracción automática de archivos comprimidos y se recomienda para copiar archivos locales.
5. **EXPOSE:** La instrucción **EXPOSE** se utiliza para especificar los puertos en los que un contenedor escuchará las solicitudes entrantes. No abre ni publica puertos, solo documenta la intención de exponer ciertos puertos, lo que es útil para documentar el funcionamiento del contenedor.
6. **CMD:** La instrucción **CMD** se utiliza para especificar el comando predeterminado que se ejecutará cuando el contenedor se inicie. Puede ser un ejecutable o un script que se ejecutará automáticamente al iniciar el contenedor. Solo puede haber una instrucción **CMD** en un Dockerfile, y si se proporcionan múltiples comandos, el último prevalece.
7. **ENTRYPOINT:** La instrucción **ENTRYPOINT** es similar a **CMD**, pero se utiliza para configurar un comando o script que actúa como un punto de entrada fijo al iniciar el contenedor. A diferencia de **CMD**, los argumentos pasados al contenedor sobrescriben el comando **CMD**, pero se agregan a la instrucción **ENTRYPOINT**.

```
$ docker run -p 8080:80 -it --rm webapplication2
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/
```

```

C:\Users\Usuario\source\repos\WebApplication2>docker exec -it f3386a5875579f4475b0c101da32052c60b91f49a37498a51f7e293198
2e816f /bin/bash
root@f3386a587557:/src# ls
Controllers  Program.cs  WeatherForecast.cs  WebApplication2.sln  appsettings.json  obj
Dockerfile  Properties  WebApplication2.csproj  appsettings.Development.json  bin
root@f3386a587557:/src# cd "app/build"
bash: cd: app/build: No such file or directory
root@f3386a587557:/src# cd /app/build
root@f3386a587557:/app/build# ls
Microsoft.OpenApi.dll  Swashbuckle.AspNetCore.SwaggerUI.dll  WebApplication2.pdb
Newtonsoft.Json.dll  WebApplication2  WebApplication2.runtimeconfig.json
Swashbuckle.AspNetCore.Swagger.dll  WebApplication2.deps.json  appsettings.Development.json
Swashbuckle.AspNetCore.SwaggerGen.dll  WebApplication2.dll  appsettings.json
root@f3386a587557:/app/build# cd ..
root@f3386a587557:/app# cd ..
root@f3386a587557:/# cd /app/publish
root@f3386a587557:/app/publish# ls
Microsoft.OpenApi.dll  Swashbuckle.AspNetCore.SwaggerUI.dll  WebApplication2.runtimeconfig.json
Newtonsoft.Json.dll  WebApplication2.deps.json  appsettings.Development.json
Swashbuckle.AspNetCore.Swagger.dll  WebApplication2.dll  appsettings.json
Swashbuckle.AspNetCore.SwaggerGen.dll  WebApplication2.pdb  web.config
root@f3386a587557:/app/publish#

```

3)

Etapa 1: Configuración de la imagen base

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

WORKDIR /app

EXPOSE 80

Etapa 2: Construcción del proyecto

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

Etapa 3: Publicación del proyecto

FROM build AS publish

RUN dotnet publish "MiProyectoWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false

Etapa 4: Configuración final de la imagen

FROM base AS final

WORKDIR /app

COPY --from=publish /app/publish .

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

4)

Dockerfile:

```
# Etapa de Build
FROM node:13.12.0-alpine as build

WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .

# Etapa de Producción
FROM node:13.12.0-alpine

WORKDIR /app
COPY --from=build /app ./

EXPOSE 3000

CMD ["node", "index.js"]
```