1)

Un Dockerfile es un archivo de configuración utilizado para construir imágenes de contenedores en Docker. Estas imágenes son la base para ejecutar aplicaciones en contenedores Docker. A continuación, te proporciono una descripción de las instrucciones más comunes utilizadas en un Dockerfile:

FROM: La instrucción FROM se utiliza para especificar la imagen base a partir de la cual se construirá la nueva imagen. Todas las demás capas y configuraciones se agregarán a esta imagen base. Por ejemplo, puedes especificar una imagen base como "ubuntu" o "nginx" que servirá como punto de partida.

RUN: La instrucción RUN se utiliza para ejecutar comandos en la imagen durante el proceso de construcción. Estos comandos pueden ser instalaciones de paquetes, configuraciones de software o cualquier tarea necesaria para preparar la imagen. Los comandos se ejecutan en una nueva capa que se agrega a la imagen.

ADD y COPY: Estas dos instrucciones se utilizan para copiar archivos y directorios desde el sistema de archivos del host al sistema de archivos de la imagen. La diferencia principal entre ellas es que ADD permite la inclusión de URLs y archivos comprimidos, mientras que COPY se utiliza principalmente para copiar archivos locales.

EXPOSE: La instrucción EXPOSE se utiliza para indicar qué puertos debe escuchar el contenedor cuando se ejecute. Esto no implica que los puertos estén disponibles desde el host, pero permite que otros contenedores se comuniquen con este contenedor a través de los puertos especificados.

CMD: La instrucción CMD se utiliza para especificar el comando que se ejecutará cuando el contenedor se inicie. Puedes proporcionar un comando y sus argumentos como una cadena o como una matriz. Si se proporciona un comando en el momento de ejecución, este reemplazará el comando especificado en CMD.

ENTRYPOINT: Similar a CMD, ENTRYPOINT se utiliza para especificar el comando que se ejecutará cuando el contenedor se inicie. La diferencia principal es que los argumentos proporcionados en el momento de ejecución se agregarán a los especificados en ENTRYPOINT en lugar de reemplazarlos. Esto permite configurar un comando base que siempre se ejecutará, con la flexibilidad de agregar argumentos adicionales.

2)

```
ignacioachaval@Ignacios-MacBook-Pro MiProyectoWebAPI % touch Dockerfile
ignacioachaval@Ignacios-MacBook-Pro MiProyectoWebAPI %
```

Dockerfile > ...

```
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"]
```

```
ignacioachaval@Ignacios-MacBook-Pro MiProyectoWebAPI % docker build -t miproyectowebapi .
[+] Building 123.6s (13/13) FINISHED
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 512B                                                                                0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0                                                   1.2s
 => [build 1/8] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:d0755fbf1ed34aecd79c127fd1dd01b80587acdb8ff50e5a68d1adf8b076922b  0.0s
 => [internal] load build context                                                                                   0.4s
 => => transferring context: 5.04MB                                                                                 0.4s
 => CACHED [build 2/8] WORKDIR /src                                                                                 0.0s
 => [build 3/8] COPY [MiProyectoWebAPI.csproj, .]                                                                   0.1s
 => [build 4/8] RUN dotnet restore "./MiProyectoWebAPI.csproj"                                                     27.1s
 => [build 5/8] COPY . .                                                                                            0.1s
 => [build 6/8] WORKDIR /src/.                                                                                      0.0s
 => [build 7/8] RUN dotnet build "MiProyectoWebAPI.csproj" -c Release -o /app/build                                78.1s
 => [build 8/8] RUN dotnet publish "MiProyectoWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false        13.9s
 => exporting to image                                                                                              2.2s
 => => exporting layers                                                                                             2.2s
 => => writing image sha256:fdbd112efbc3cf103439059354d7ef39f4e7416f1c567c3608013cab517e9814                        0.0s
 => => naming to docker.io/library/miproyectowebapi                                                                 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
ignacioachaval@Ignacios-MacBook-Pro MiProyectoWebAPI %
```

```
ignacioachaval@Ignacios-MacBook-Pro MiProyectoWebAPI % docker run -p 8082:80 -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
```

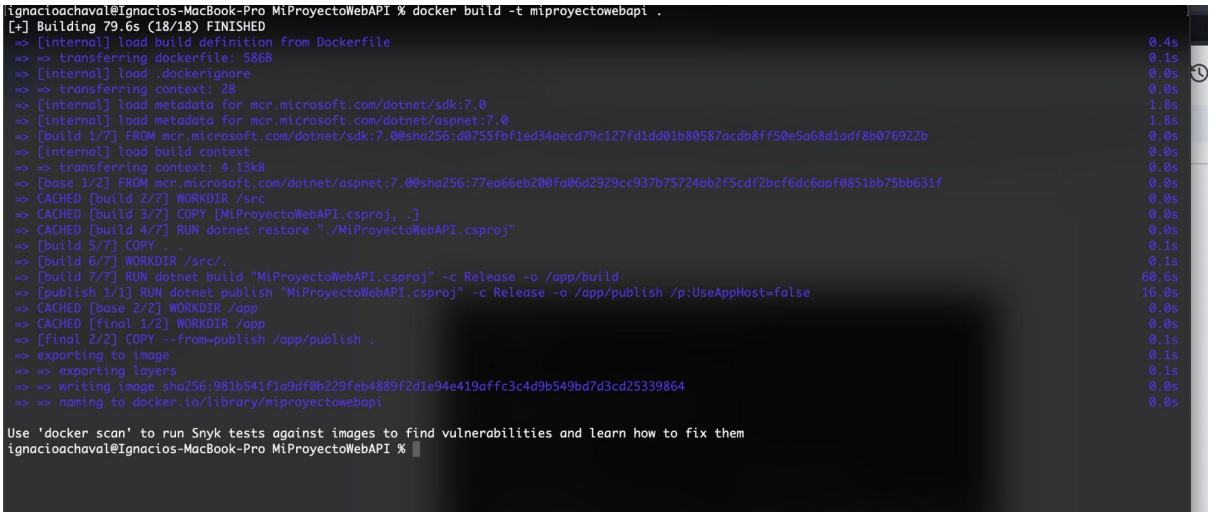
3)

```dockerfile
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"]
```



El Dockerfile proporcionado utiliza una estrategia de compilación en múltiples etapas (multi-stage build). Esto se puede ver por la presencia de las diferentes secciones o etapas claramente definidas en el archivo Dockerfile, cada una con su propia imagen base y propósito:

La primera etapa (llamada "base") utiliza la imagen mcr.microsoft.com/dotnet/aspnet:7.0 como base para ejecutar la aplicación ASP.NET. En esta etapa, se establece el directorio de trabajo, se expone el puerto y se configuran las variables de entorno para la aplicación en tiempo de ejecución.

La segunda etapa (llamada "build") utiliza la imagen mcr.microsoft.com/dotnet/sdk:7.0 como base para compilar la aplicación. En esta etapa, se copian los archivos de código fuente, se restauran las dependencias, se compilan y se colocan los archivos compilados en un directorio específico.

La tercera etapa (llamada "publish") también se basa en la etapa "build" y se utiliza para publicar la aplicación. Aquí, se ejecuta el comando dotnet publish para generar los archivos listos para la implementación.

Finalmente, la cuarta etapa (llamada "final") se basa en la etapa "base" y se utiliza para construir la imagen final. En esta etapa, se copian los archivos publicados desde la etapa "publish" al directorio de trabajo de la etapa "final", y se establece el punto de entrada para la aplicación.

4)

```
ignacioachaval@Ignacios-MacBook-Pro ingsoft3 % mkdir nodejs-docker
ignacioachaval@Ignacios-MacBook-Pro ingsoft3 % cd nodejs-docker
ignacioachaval@Ignacios-MacBook-Pro nodejs-docker % touch Dockerfile
ignacioachaval@Ignacios-MacBook-Pro nodejs-docker % mate Dockerfile
```

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

WORKDIR /app

# Copia los archivos del proyecto y el package.json
COPY package*.json ./
COPY . .

# Instala las dependencias
RUN npm install

# Compila la aplicación (esto puede variar según tu proyecto)
RUN npm run build

# Etapa de producción
FROM node:13.12.0-alpine

WORKDIR /app

# Copia los archivos generados en la etapa de construcción
COPY --from=build /app .

# Expon el puerto 3000
EXPOSE 3000

# Inicia la aplicación
CMD ["npm", "start"]
```

```
ignacioachaval@Ignacios-MacBook-Pro nodejs-docker % docker build -t test-node .

[+] Building 24.3s (4/12)
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 632B                                                                                0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [internal] load metadata for docker.io/library/node:13.12.0-alpine                                              3.7s
 => [auth] library/node:pull token for registry-1.docker.io                                                         0.0s
 => [internal] load build context                                                                                  20.4s
 => => transferring context: 60.09MB                                                                               20.4s
 => [build 1/6] FROM docker.io/library/node:13.12.0-alpine@sha256:cc85e728fab3827ada20a181ba280cae1f8b625f256e2c86b9094d9bfe834766  20.4s
 => => resolve docker.io/library/node:13.12.0-alpine@sha256:cc85e728fab3827ada20a181ba280cae1f8b625f256e2c86b9094d9bfe834766  0.0s
 => => sha256:483343d6c5f5d658963b7c16e4299247f765bef4025012457ee105481ea1afc1 6.77kB / 6.77kB                      0.0s
 => => sha256:aad63a9339440e7c3e1fff2b988991b9bfb81280042fa7f39a5e327023056819 2.80MB / 2.80MB                      0.8s
 => => sha256:a00bd932208e2de29cd2b7e0bab954325913d146401effb482fff3d8775aaaeb 35.29MB / 35.29MB                    7.7s
 => => sha256:cc85e728fab3827ada20a181ba280cae1f8b625f256e2c86b9094d9bfe834766 1.19kB / 1.19kB                      0.0s
 => => sha256:ed06820d0fb6f4711e0a6f50c9f147fb2596399866319e1bb3b0a52393c5615f 1.16kB / 1.16kB                      0.0s
 => => sha256:c57f2c59b93778192e60e0e213949630f9ad30324736bbb0a71e12adf8a16d46 2.24MB / 2.24MB                      2.4s
 => => extracting sha256:aad63a9339440e7c3e1fff2b988991b9bfb81280042fa7f39a5e327023056819                           1.8s
 => => sha256:f3446470f297e51c1e27f90f7ae9a94f1ee7b6c8e529aec83aa1a04ad70531c0 284B / 284B                          1.4s
 => => extracting sha256:a00bd932208e2de29cd2b7e0bab954325913d146401effb482fff3d8775aaaeb                          11.7s
```

```
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
ignacioachaval@Ignacios-MacBook-Pro my-app % docker run -p 8083:80 -it --rm miproyectowebapi
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:80
```


```
[{"date":"2023-10-21","temperatureC":33,"temperatureF":91,"summary":"Scorching"},{"date":"2023-10-22","temperatureC":1,"temperatureF":33,"summary":"Hot"},{"date":"2023-10-23","temperatureC":25,"temperatureF":76,"summary":"Sweltering"},{"date":"2023-10-24","temperatureC":-8,"temperatureF":18,"summary":"Mild"},{"date":"2023-10-25","temperatureC":-8,"temperatureF":18,"summary":"Scorching"}]
```


5)

```
      Application is shutting down...
ignacioachaval@Ignacios-MacBook-Pro my-app % docker login
Authenticating with existing credentials...
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
ignacioachaval@Ignacios-MacBook-Pro my-app % docker tag test-node ignacioachaval94/test-node:latest
ignacioachaval@Ignacios-MacBook-Pro my-app % docker push ignacioachaval94/test-node:latest
The push refers to repository [docker.io/ignacioachaval94/test-node]
5f098022b171: Pushed
75bbdfba4b79: Pushed
65d358b7de11: Mounted from library/node
f97384e8ccbc: Mounted from library/node
d56e5e720148: Mounted from library/node
beee9f30bc1f: Mounted from library/node
latest: digest: sha256:a7ef8ff899814f34b6692b1dcf67c205bc5166e2256c14995a2394a480d0ef29 size: 1577
ignacioachaval@Ignacios-MacBook-Pro my-app %
```