Trabajo Práctico N°6

Ejercicio 1

Conceptos de Dockerfile:

1. **FROM (DESDE):**
    - Descripción: Inicializa una nueva etapa de construcción y establece la Imagen Base para instrucciones posteriores. Cada Dockerfile válido debe empezar con esta instrucción.
    - Uso: `FROM <nombre_imagen>`
2. **RUN (EJECUTAR):**
    - Descripción: Ejecuta comandos en una nueva capa encima de la imagen actual y guarda los resultados en una nueva imagen. Útil para instalar software y configurar el entorno.
    - Uso:

```
RUN <comando>    (forma de shell)
RUN ["ejecutable", "param1", "param2"]   (forma ejecutable)
```

3. **ADD (AGREGAR):**
    - Descripción: Copia nuevos archivos, directorios o URLs remotas a la imagen del contenedor en una ruta específica.
    - Uso:

```
ADD [--chown=<usuario>:<grupo>] [--chmod=<permisos>] <fuente> <destino>
ADD [--chown=<usuario>:<grupo>] [--chmod=<permisos>] ["<fuente>",... "<destino>"]
```

4. **COPY (COPIAR):**
    - Descripción: Copia nuevos archivos o directorios desde la fuente al sistema de archivos del contenedor en una ruta específica.

- Uso:

```
COPY [--chown=<usuario>:<grupo>] [--chmod=<permisos>] <fuente> <destino>
COPY [--chown=<usuario>:<grupo>] [--chmod=<permisos>] ["<fuente>",... "<destino
>"]
```

5. **EXPOSE (EXPONER):**
   - Descripción: Informa a Docker que el contenedor escucha en el puerto especificado durante la ejecución. No publica el puerto, solo documenta la intención.
   - Uso: `EXPOSE <puerto> [<protocolo>]`
6. **CMD (COMANDO):**
   - Descripción: Define la configuración predeterminada para la ejecución del contenedor. Puede especificar un ejecutable o usarlo en conjunto con ENTRYPOINT.
   - Uso:

```
CMD ["ejecutable", "param1", "param2"]   (forma ejecutable, preferida)
CMD ["param1", "param2"]   (como parámetros por defecto para ENTRYPOINT)
CMD comando param1 param2   (forma de shell)
```

7. **ENTRYPOINT (PUNTO DE ENTRADA):**
   - Descripción: Configura el contenedor para que se ejecute como un ejecutable. Puede ser en forma ejecutable o en forma de shell.
   - Uso:

```
ENTRYPOINT ["ejecutable", "param1", "param2"]   (forma ejecutable)
ENTRYPOINT comando param1 param2   (forma de shell)
```

Ejercicio 2

```
Usuario@DESKTOP-8G1HL0R MINGW64 ~/source/repos/WebApplication2
$ 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#
```

Ejercicio 3

1. **Primera Etapa (Base):**
   - Utiliza la imagen `mcr.microsoft.com/dotnet/aspnet:6.0` como base. Esta imagen contiene el entorno de ejecución ASP.NET.
   - Establece el directorio de trabajo en `/app`.
   - Expone el puerto 80.
2. **Segunda Etapa (Build):**
   - Utiliza la imagen `mcr.microsoft.com/dotnet/sdk:6.0` como base. Esta imagen contiene el SDK de .NET para compilar aplicaciones.
   - Establece el directorio de trabajo en `/src`.
   - Copia el archivo de proyecto `WebApplication2.csproj` al directorio actual.

- Ejecuta `dotnet restore` para restaurar las dependencias del proyecto.
- Copia todo el contenido actual al directorio de trabajo en el contenedor (`/src`).
- Cambia el directorio de trabajo a `/src`.
- Ejecuta `dotnet build` para compilar el proyecto en modo Release y lo coloca en el directorio `/app/build`.
- Ejecuta `dotnet publish` para publicar la aplicación en modo Release y lo coloca en el directorio `/app/publish`. Se deshabilita el uso de un host de aplicación adicional.

3. **Tercera Etapa (Publish):**
   - Utiliza la imagen definida en la etapa anterior (`build`) como base.
   - Ejecuta `dotnet publish` para publicar la aplicación en modo Release y la coloca en el directorio `/app/publish`. Se deshabilita el uso de un host de aplicación adicional.
4. **Cuarta Etapa (Final):**
   - Utiliza la imagen definida en la primera etapa (`base`) como base.
   - Establece el directorio de trabajo en `/app`.
   - Copia los archivos publicados desde la etapa de publicación (`publish`) al directorio actual (`/app`).
   - Establece el punto de entrada de la aplicación utilizando el comando `dotnet` para ejecutar `WebApplication2.dll`.

La principal diferencia entre los dos Dockerfiles es la introducción de una tercera etapa llamada `publish` en el segundo Dockerfile. Esta etapa se utiliza específicamente para realizar la publicación de la aplicación. Luego, en la cuarta etapa (`final`), se copian los archivos publicados desde la etapa de `publish` al directorio de trabajo actual. Esto ayuda a reducir el tamaño de la imagen final, ya que solo incluye los archivos necesarios para ejecutar la aplicación y no los archivos de desarrollo y construcción.

```
C:\Users\Usuario\source\repos\WebApplication2>docker exec -it 431d35d8e84cfe947a0cd83dd423f6129be27dd4ba36504c1c16c0fa0e4d1c3e /bin/bash
root@431d35d8e84c:/app# cd ..
root@431d35d8e84c:/# ls
app  bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@431d35d8e84c:/# cd app
root@431d35d8e84c:/app# cd build
bash: cd: build: No such file or directory
root@431d35d8e84c:/app# 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
```

Ejercicio 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"]
```

```
Usuario@DESKTOP-8G1HL0R MINGW64 ~/OneDrive/Escritorio/2023/Ing SW 3/tp6/trabajo-practico-06/nodejs-docker
$ docker build -t test-node .
[+] Building 3.2s (12/12) FINISHED                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                          0.1s
 => => transferring dockerfile: 296B                                                                          0.0s
 => [internal] load .dockerignore                                                                             0.1s
 => => transferring context: 2B                                                                               0.0s
 => [internal] load metadata for docker.io/library/node:13.12.0-alpine                                        2.5s
 => [auth] library/node:pull token for registry-1.docker.io                                                   0.0s
 => [internal] load build context                                                                             0.0s
 => => transferring context: 445B                                                                             0.0s
 => [stage-1 1/3] FROM docker.io/library/node:13.12.0-alpine@sha256:cc85e728fab3827ada20a181ba280cae1f8b625f256e2  0.0s
 => CACHED [stage-1 2/3] WORKDIR /app                                                                         0.0s
 => CACHED [build 3/5] COPY package*.json ./                                                                  0.0s
 => CACHED [build 4/5] RUN npm install                                                                        0.0s
 => [build 5/5] COPY . .                                                                                      0.1s
 => [stage-1 3/3] COPY --from=build /app ./                                                                   0.1s
 => exporting to image                                                                                        0.1s
 => => exporting layers                                                                                       0.1s
 => => writing image sha256:126622045ceec869f8cdda4b2c0f83a319d831f40e00c74db5b6f4edf93e4ea0                  0.0s
 => => naming to docker.io/library/test-node                                                                  0.0s

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview

Usuario@DESKTOP-8G1HL0R MINGW64 ~/OneDrive/Escritorio/2023/Ing SW 3/tp6/trabajo-practico-06/nodejs-docker
$

Usuario@DESKTOP-8G1HL0R MINGW64 ~/OneDrive/Escritorio/2023/Ing SW 3/tp6/trabajo-practico-06/nodejs-docker
$ docker run -p 3000:3000 test-node
Servidor corriendo en http://localhost:3000/
```

Ejercicio 5

```
C:\Users\Usuario>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/acc
ess-tokens/

C:\Users\Usuario>docker tag test-node aaraya0/test-node:latest

C:\Users\Usuario>docker push aaraya0/test-node:latest
The push refers to repository [docker.io/aaraya0/test-node]
4b072afbf52d: Pushed
d1fac9a5d80f: Pushed
65d358b7de11: Mounted from library/node
f97384e8ccbc: Mounted from library/node
d56e5e720148: Mounted from library/node
beee9f30bc1f: Mounted from library/node
latest: digest: sha256:71fb18475404c86155ec2c3d0ad27996d2d4d7c4fb3afeb7517394ad79f1cb46 size: 1572
```