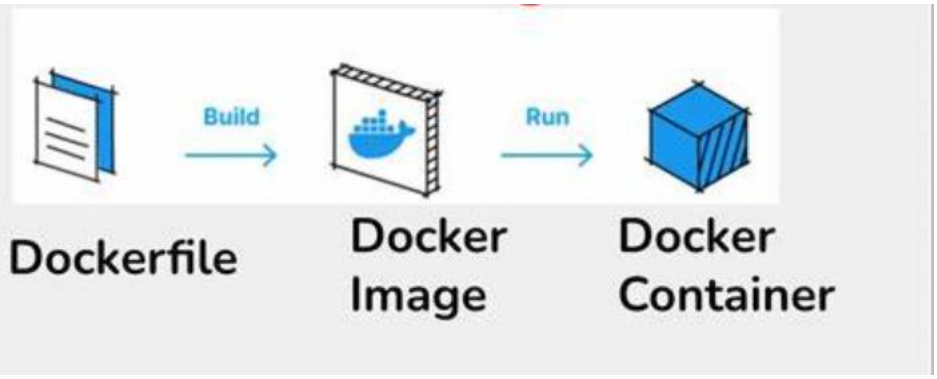
Docker Documentation

Docker is mainly needed for deploying the application in a platform-independent system.

Steps for Containerizing a .NET 8 Console App



Step 1:

Publish your existing application and see following files exist or not:

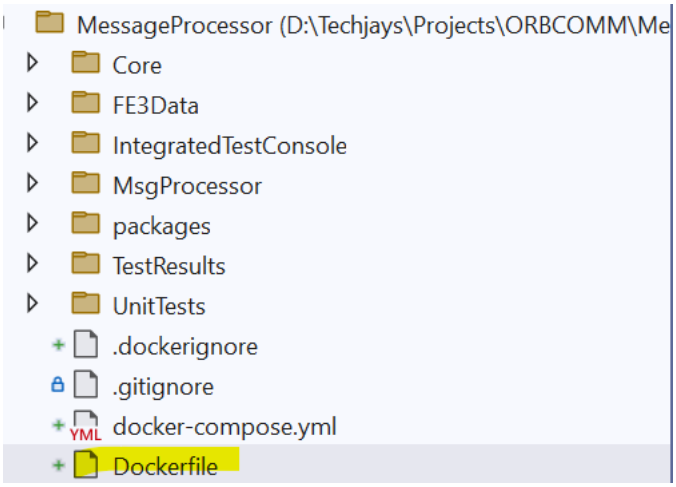
Suppose your app name is Docker

```
C:\Users\default\App\bin\Release\net8.0\publish

      LastWriteTime         Length Name
      -------------         ------ ----
 9/22/2023  9:17 AM            431 DotNet.Docker.deps.json
 9/22/2023  9:17 AM           6144 DotNet.Docker.dll
 9/22/2023  9:17 AM         157696 DotNet.Docker.exe
 9/22/2023  9:17 AM          11688 DotNet.Docker.pdb
 9/22/2023  9:17 AM            353 DotNet.Docker.runtimeconfig.js
```

Step 2:

Create a docker file in the root directory under the solution:



And the content of dockerfile is like below:

```
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build-env       # Take Image of .NET 8
WORKDIR /App

# Copy everything from your local projects all the folders and sub folders to the /App
# You created this directory in docker registory via  WORKDIR /App
#  Forget about your local project path. Your project is in now /App Directory.

COPY . ./

# Restore and publish for MsgProcessor project
WORKDIR /App/MsgProcessor   # change the directory to  MsgProcessor bacuse it is the main app

 # ./MsgProcessor.csproj    means  /App/MsgProcessor/ MsgProcessor.csproj  and u change the path

RUN dotnet restore ./MsgProcessor.csproj   # this is the main project file ./MsgProcessor.csproj
RUN dotnet publish -c Release -o out    # publish file in the out directory /App/MsgProcessor/out

# Build runtime image
FROM mcr.microsoft.com/dotnet/aspnet:8.0
WORKDIR /App            # again Change the directory
COPY --from=build-env /App/MsgProcessor/out .    # copy build env copy to the /out directory

ENTRYPOINT ["dotnet", "MsgProcessor.dll"]
```

Step 3:

To build the container, from your terminal, run the following command:
Go to the application directory and open the terminal:

```
docker build -t counter-image -f Dockerfile .
```

Where `counter-image` is the image name.

To see a list of images installed

```
docker images
```

Step 4:

Create a Container from the Image

```
docker create --name core-counter counter-image
```

To see a list of *all* containers:

```
docker ps -a
```

Start the container

```
docker start core-counter
```

Stop the Container

```
docker stop core-counter
```

To Build the docker images from the Remote Windows or linux server

Follow the following steps:

Login To docker hub- https://hub.docker.com/ (if not sign up sign up first)

Build the Docker image (if not already built)

```
docker build -t my_app_image .
```

Tag the Docker image

```
docker tag my_app_image myusername/my_app_image:latest
```

Log in to Docker Hub

```
docker login
```

Push the Docker image to Docker Hub

```
docker push myusername/my_app_image:latest
```

On the Target Server

Log in to Docker Hub

```
docker login
```

Pull the Docker image from Docker Hub

```
docker pull myusername/my_app_image:latest
```

Run a container from the pulled image

```
docker run --name my_app_container -d myusername/my_app_image:latest
```

 # To see the docker logs from the running application

```
docker logs nameofyourcontainer
```