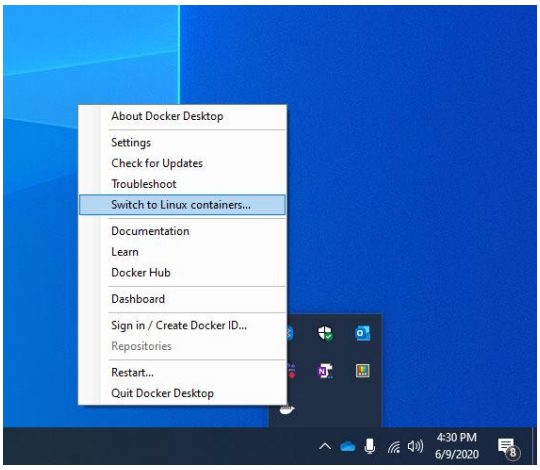
ASP.NET Core MVC & SQL/Server with Docker Compose

Let's start to dockerize your ***ASP.NET Core MVC*** project along with ***Database*** through *Docker Compose file* with me. Note few things and verify them.

- You must have installed ***Docker Desktop*** on your PC.
- You must use WSL 2 based Version.
- You must switch to Linux Containers like this.



- If it shows *Switch to Windows Containers,* it means Docker Desktop is running already in Linux Containers, in this case skip this step.
- You must have ASP.NET Core MVC 6.0 Project.
- You must use Code First Approach of Entity Framework in your project.
- Make sure to download ***Azure Studio*** for manually handle Database.

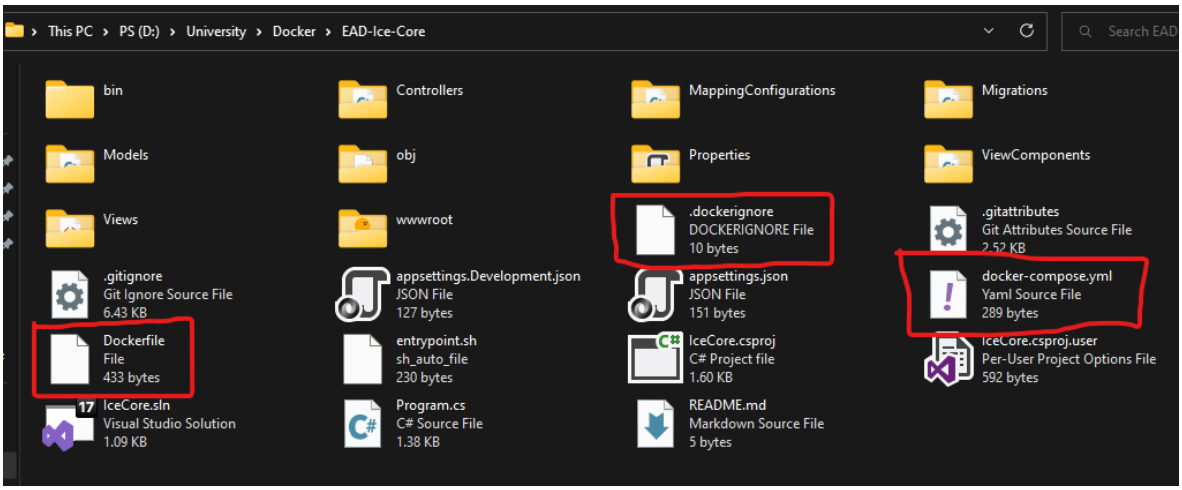
STEP 1

⇨ Go to C:\Users*[User Name]*\.docker\config.json

⇨ Replace `credsStore` to `credStore`

STEP 2

⇨ Open your project Main Directory and create these files, for example in my case.



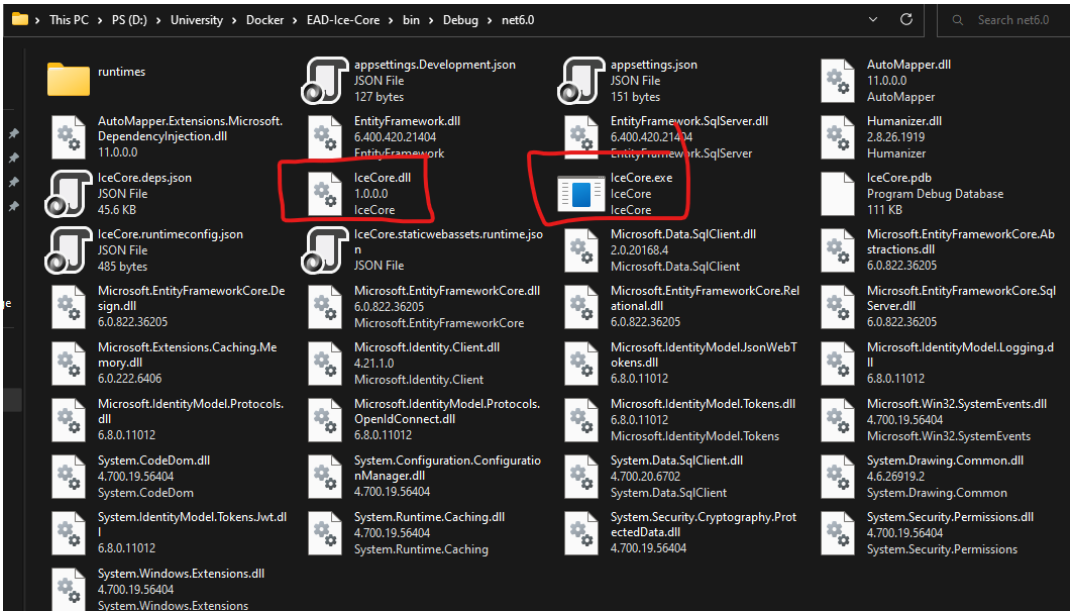
- ⇨ Common mistakes here.
  - Dockerfile has small ***'f'*** in name and has no extention
  - docker-compose.yml, extention = ***.yml***, **'d' small, 'c' small**

STEP 3

- ⇨ Paste the following code in your **Dockerfile**

```
# syntax=docker/dockerfile:1
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build-env
WORKDIR /app
# Copy csproj and restore as distinct layers
COPY *.csproj ./
RUN dotnet restore
# Copy everything else and build
COPY ./ ./
RUN dotnet publish -c Release -o out
# Build runtime image
FROM mcr.microsoft.com/dotnet/aspnet:6.0
WORKDIR /app
COPY --from=build-env /app/out .
ENTRYPOINT ["dotnet", "@YourProject.dll"]
```

- ⇨ You can get **@YourProject.dll** by simply going to bin/Debug/net6.0/ folder.
- ⇨ Search for the **.dll** file, just before **.exe** file it will be your is your **@YourProject.dll**
- ⇨ In my case



ENTRYPOINT ["dotnet", "**IceCore.dll**"]

- ⇨ Paste the following code in your **docker-compose.yml**
- ⇨ Common Mistakes:

- Dockerfile & docker-compose.yml must be in same directory, pay intention on indentation of code, if is just like python indentation.

```
version: "3.9"
services:
  frontend:
    build: .
    ports:
      - "8000:80"
  backend:
    image: "mcr.microsoft.com/mssql/server"
    environment:
      SA_PASSWORD: "Docker123!"
      ACCEPT_EULA: "Y"
    ports:
      - "1440:1433"
```

⇨ Try to use same port numbers, that I mentioned here.

⇨ Paste the following code in your .**dockerignore**

```
bin/
obj/
```

STEP 4

⇨ Go to your ***[my]DBcontext.cs*** file present in Models folder of your project

⇨ Change the connection string to given string

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) {
   if (!optionsBuilder.IsConfigured) {
        optionsBuilder.UseSqlServer(@"Server = db; Database = master; User = sa; Password = Docker123!;");
   }
}
```

⇨ Open terminal and add migrations

- dotnet ef migrations add DbContainer
- dotnet ef database update
- or you can also mention in constructor for migrations

```
public [my]DbContext() {
   Database.Migrate();
}
```

⇨ now you have ***master*** Database contains all tables of your project, lets start to containerize it.

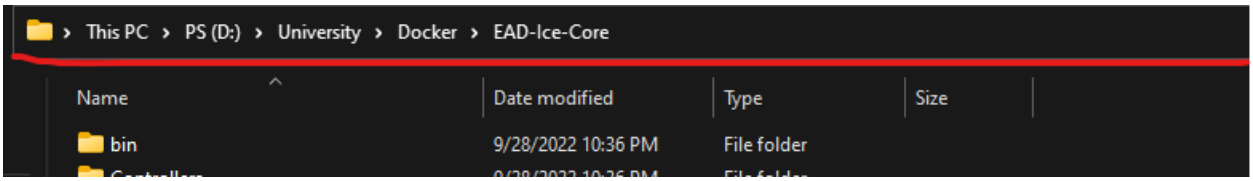
⇨ Again change & replace the following code.

```
 protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) {
if (!optionsBuilder.IsConfigured) {
   optionsBuilder.UseSqlServer(@"Server = localhost, 1440; Database = master; User = sa; Password =
Docker123!;");
}
```

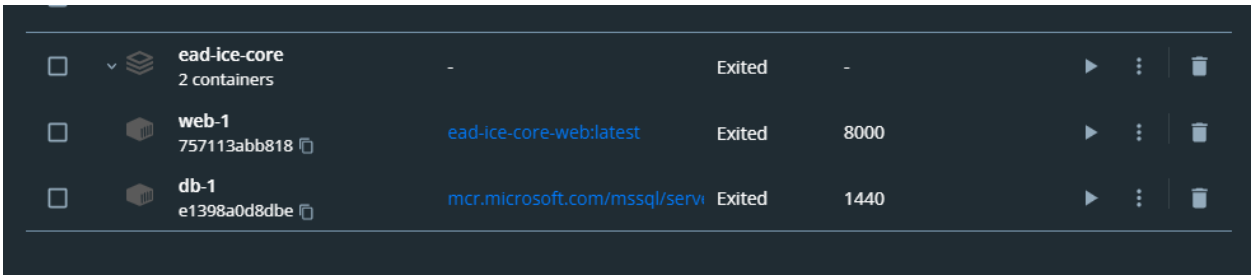
```
}
```

STEP 5

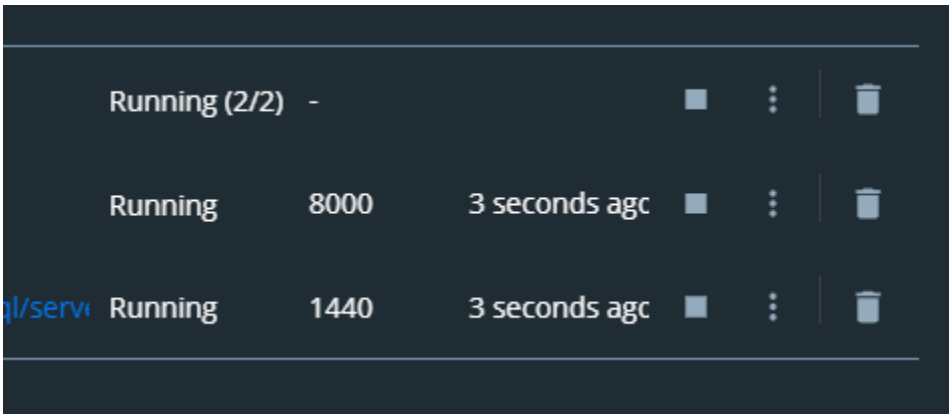
- Go to your project directory, in address bar type **cmd,** *press enter*.



- Type ***docker-compose build*** in **cmd** prompt, enter.
- After this, type ***docker-compose up*** in **cmd** prompt, enter.
- Now you have both containers in your **Docker Desktop** like this.



- Run the containers and run web or frontend having port 8000



- For further help visit: https://docs.docker.com/samples/aspnet-mssql-compose/

Credit of this file goes to Syed Inshal Hussain