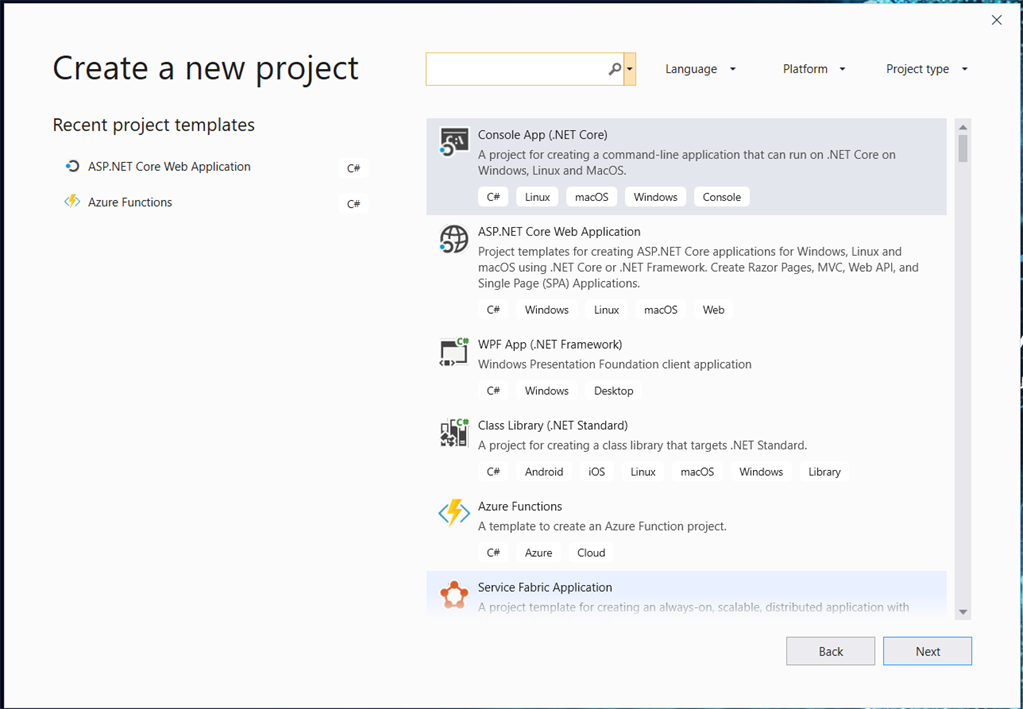
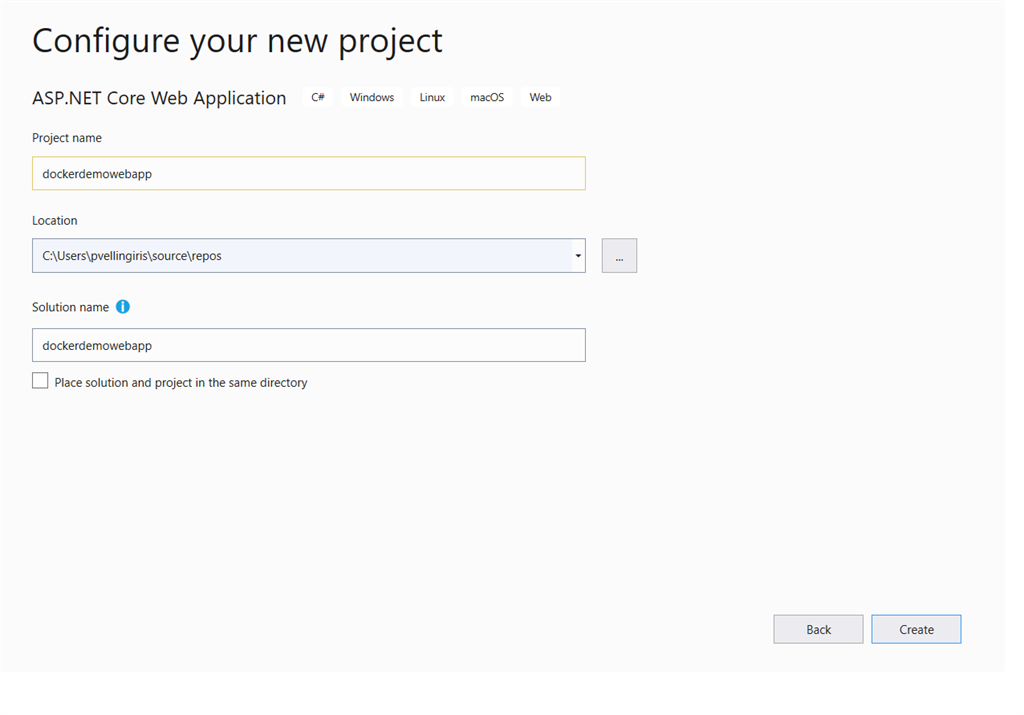
**Step 1**

Open Visual Studio 2019 and create a new project.



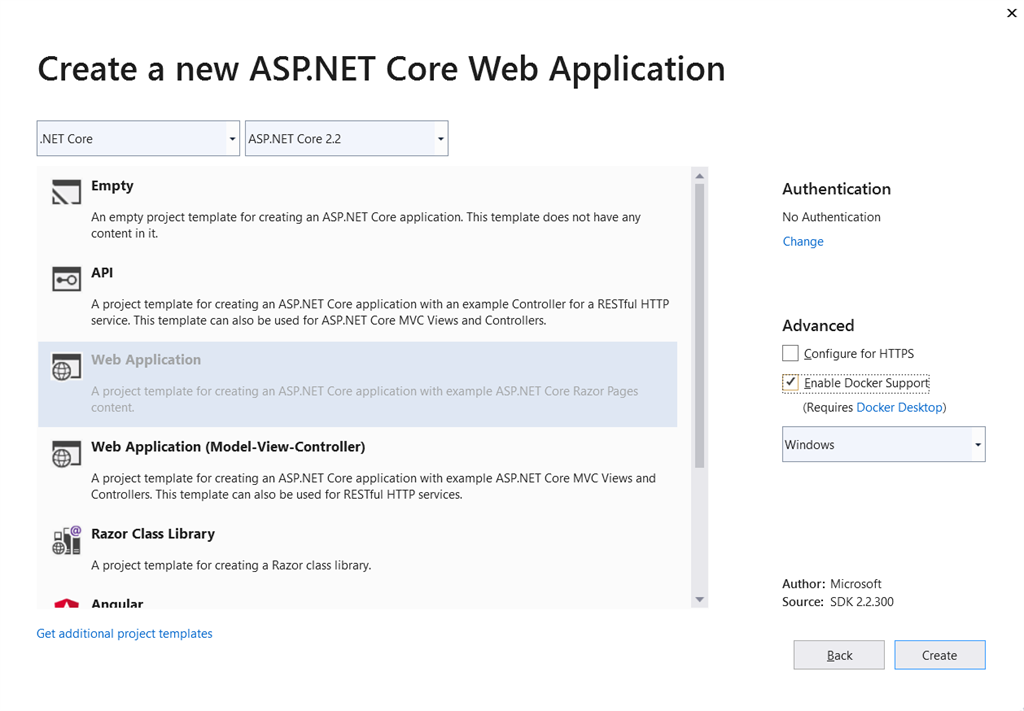
**Step 2**

Select ASP.Net Core Web Application --> Click Next --> Provide project name.



**Step 3**

Click Create and then select .Net Core as a framework and select .Net Core 2.2 . Next, Select Web Application in the left side panel. In the right-side panel, select Enable docker support and make sure the windows option is selected. This docker image is windows-specific.



**Note**

If you are missed to select docker for windows support while creating an application, don't worry, you have an alternate option. Right-click on the project in solution explorer and select Add --> there you can see docker support.

**Step 4**

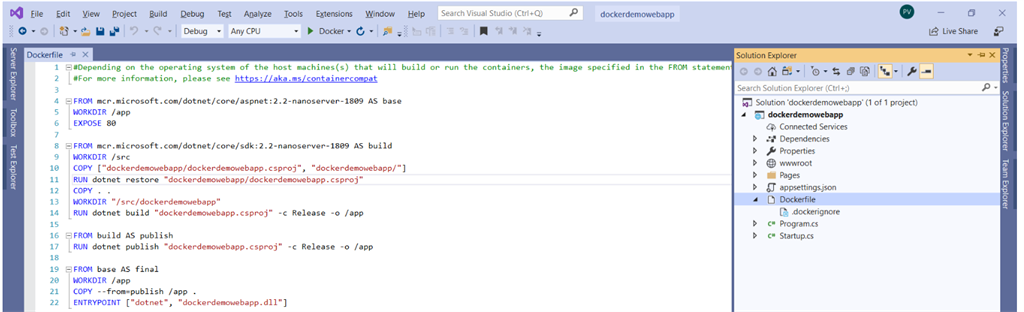
Once the sample project is created, you can visit DockerFile in the solution explorer of the created project. There you can find a few commands, which states that what are the steps to be followed to create an image of this application. You can also see .dockerignore file in project solution explorer. It is acting like a git ignore file, ignoring a few files while creating a docker image.

For Example:

1. # Get base sdk from microsoft
2. #Copy the CSPROJ and any dependencies via nuget
4. //FROM mcr.microsoft.com/dotnet/core/aspnet:2.2-nanoserver-1809 AS base
5. //WORKDIR /app
6. //EXPOSE 80 ---> For Port mapping with container.
8. #Copy our project files and build our release
10. //FROM mcr.microsoft.com/dotnet/core/sdk:2.2-nanoserver-1809 AS build
11. //WORKDIR /src
12. //COPY ["dockerdemowebapp/dockerdemowebapp.csproj", "dockerdemowebapp/"]
13. //RUN dotnet restore "dockerdemowebapp/dockerdemowebapp.csproj"
14. //COPY . .
15. //WORKDIR "/src/dockerdemowebapp"
16. //RUN dotnet build "dockerdemowebapp.csproj" -c Release -o /app
18. //FROM build AS publish
19. //RUN dotnet publish "dockerdemowebapp.csproj" -c Release -o /app
21. //FROM base AS final
22. //WORKDIR /app
23. //COPY --from=publish /app .
24. //ENTRYPOINT ["dotnet", "dockerdemowebapp.dll"

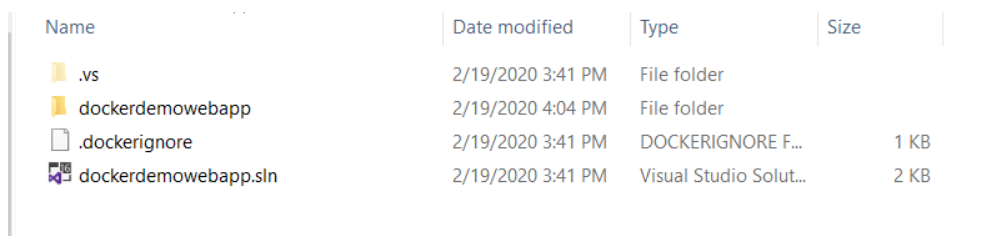
**Step 5**

Finally, our application contains the following files.



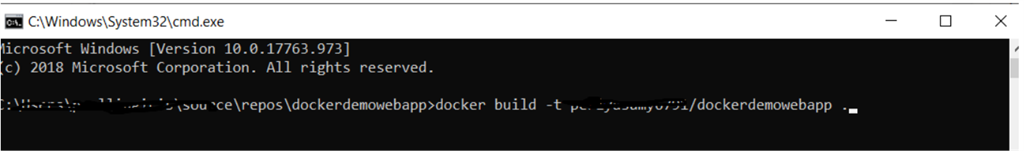
**Step 6**

Let's start with creating a docker image, so to do that open command prompt on project file location. Which means the below folder location:



**Step 7**

In opened command prompt enter the following command to create an image. docker build -t dockerhubid/projectname: latest .



Command: docker build -t dockerhubid/projectname: latest.

So in our case command is like, docker build -t dockerhubid/dockerdemowebapp .

-t --> means tagging this image with the following name.

dockerhubid --> https://hub.docker.com/ // Here you can create docker hubid.

dockerhubid/dockerdemowebapp --> You can give any name in that place but when you try to host this docker container into Azure or any other cloud service means you should move this docker container image into docker hub and then only you can deploy.

By projectname, it means the full name that we have given.

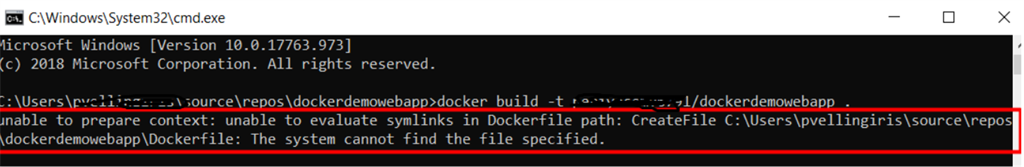
Finally we have . (dot) symbol which is important with space. We have to follow the docker principle.

**Note**

In the command part, you can see the latest keyword. It is not mandatory, if not provided, it will take the latest version.

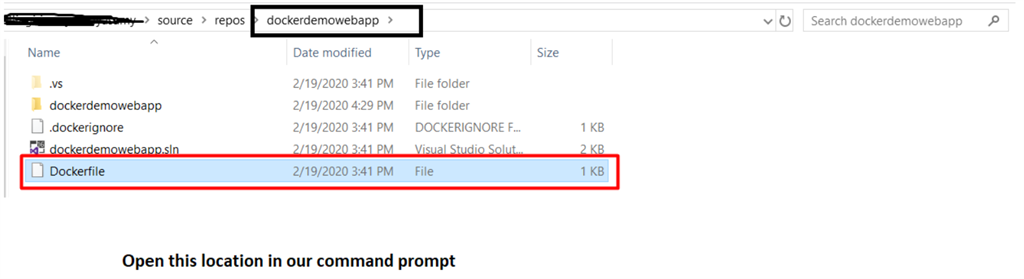
**Step 8**

When you hit enter, it will try to read the docker file and execute the commands which we have written. Let see!!



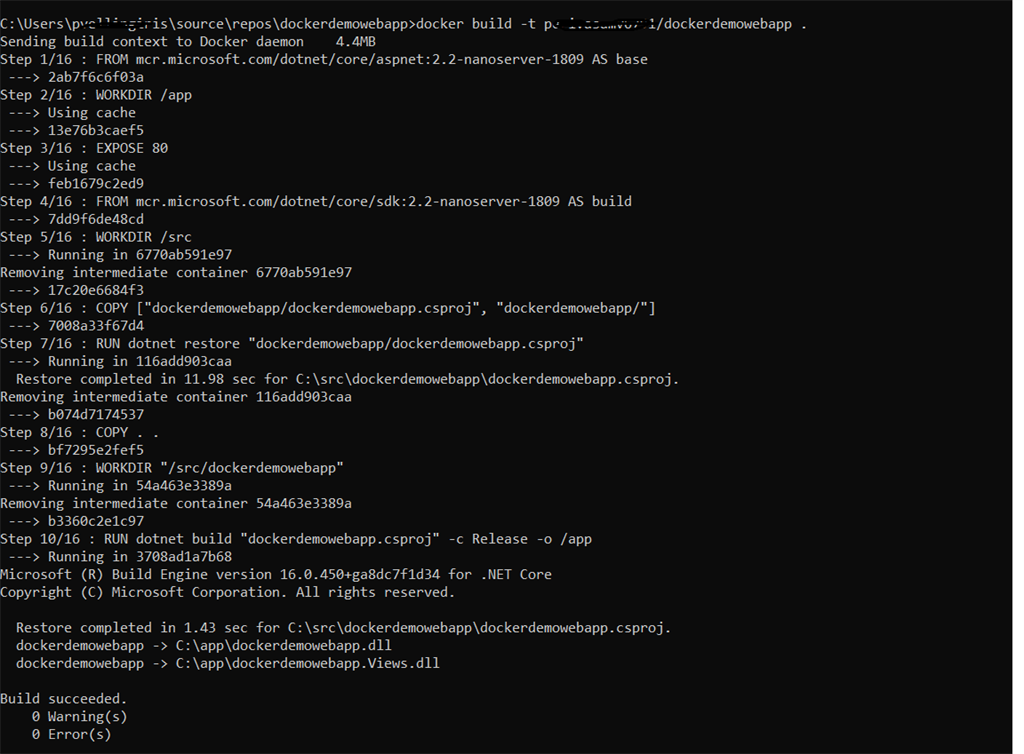
**Step 9**

OOPS, something went wrong, it is because the docker file is not in the correct place. Open the file explorer and navigate into the project location. Move the dockerfile one step backward, which means move the docker file into solution file location and open the command prompt on that location. Like in the below image.



**Step 10**

Now enter the build command and wait for executing the docker file commands (which means dependencies and all other related). Once the command is successful, you can see the success window.



**Step 11**

Once the docker image has been created successfully, you can enter the below command to see the images.

command: docker images

There you can see your recently created images.

**Step 12**

So, we have created the docker image. Now we need to run this image locally. To do that we need a container to run this image. This process can be achieved by running the below command.

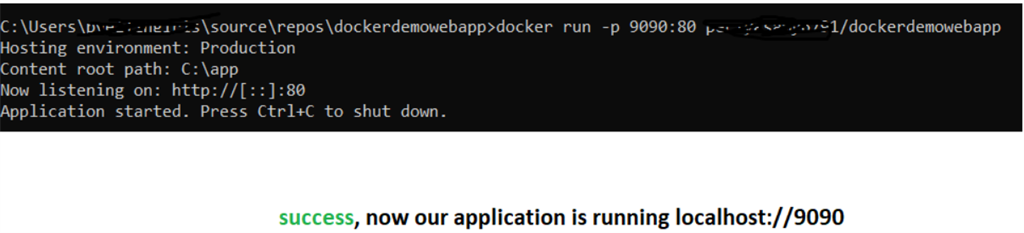
command: docker run -p 9090:80 dockerhubid/projectname

-p --> It means port mapping.

9090 --> It is our random port number to run our application (We can give our own).

80 --> This one is we exposed 80 port while creating an image. So that port number we have to map with container that we created now.

So let see by running the command in our case.



**Step 13**

Success**,**now our application is running as a docker image in this URL.

*http://localhost:9090/*

