[**1. SECTION – 1 : Pull Image using Docker Hub** 1](#_Toc172235728)

[**2. SECTION – 2 : Create Image for Console Application** 3](#_Toc172235729)

[**3. SECTION – 2.1 : Create Image for .Net Core Application** 6](#_Toc172235730)

[**4. SECTION – 2.2 : Container for SQL Server Image** 10](#_Toc172235731)

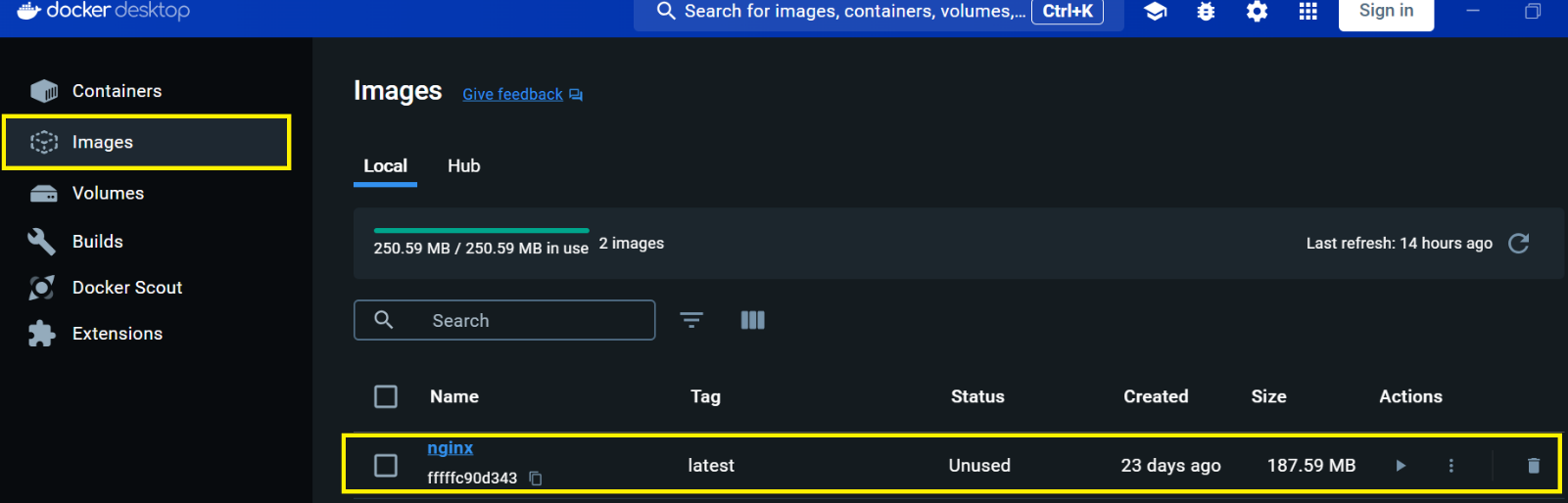
# **1. SECTION – 1 : Pull Image using Docker Hub**

1. Open Docker Desktop.
2. Example => For nginx image, Pull the nginx image from <https://hub.docker.com/>
3. Open Command prompt and execute the below command.

A screenshot of a computer program

Description automatically generated

1. Lates downloaded pulled images we are now able to see in Docker Desktop.



1. Run the below command for container which contains inside images (Publish Code) .

i.e, container contains the running instance of image.



Where nginx is the Web Server of Linux, same as IIS Web Server of Windows

**--name nginx =>** --name flag is for specify container name and nginx is the Name of the container.

**-p =>** Flag for port

**8080 =>** Port on local machine

**80 =>** Port on Web Server (By default port on web server is 80)

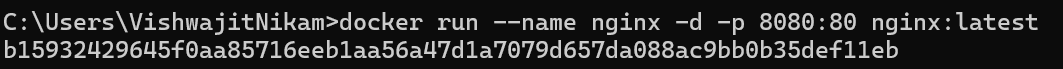
1. Now the container is running on container section of Docker Desktop

A screenshot of a computer

Description automatically generated

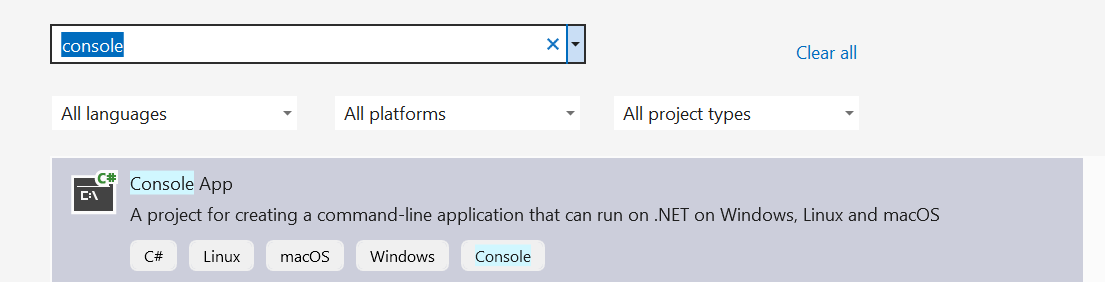
1. We can run the container using detach mode (-d flag used). It creates the container but will not attach to terminal of the running container, which doesn’t show any shell (Logs) in the console. But is show the shell in docker desktop behind the scene.

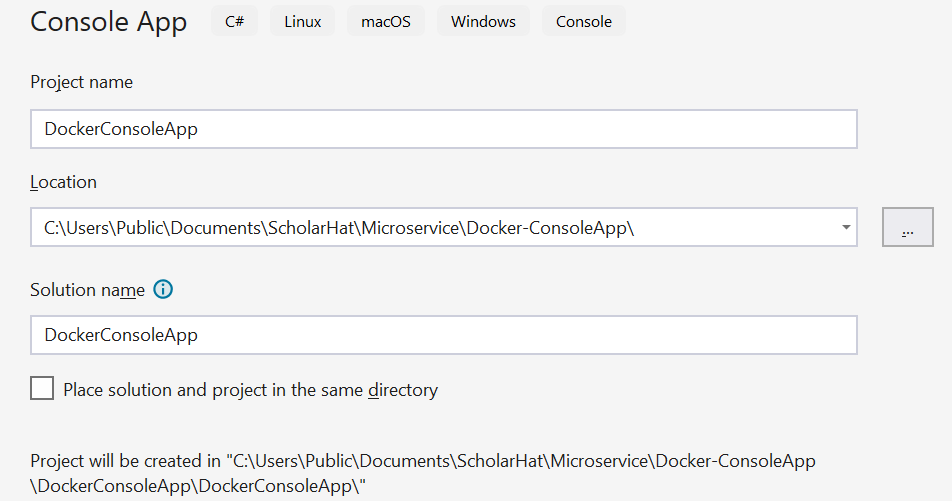
So, delete the existing container which running the inginx image from docker desktop.



# **2. SECTION – 2 : Create Image for Console Application**

1. Create Project 🡪 Select the console application template

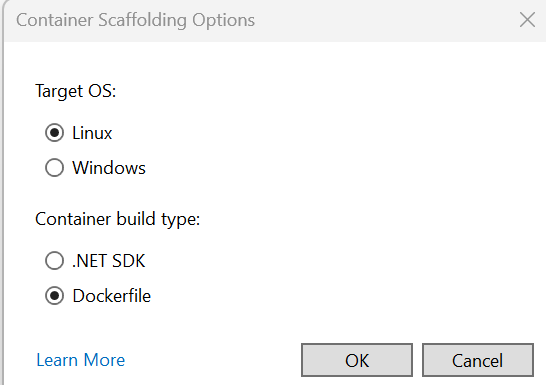




A screenshot of a computer

Description automatically generated

1. Right click on ConsoleApp 🡪 Add 🡪 Docker Support 🡪 choose the below options



1. In Docker file mention the below commands.

FROM mcr.microsoft.com/dotnet/sdk:8.0 AS Build

WORKDIR /app

COPY . .

ENTRYPOINT ["dotnet", "run"]

Where 1st line is base image from Microsoft Container Registry

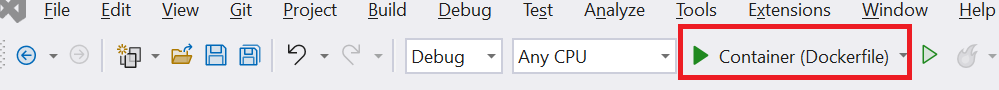
2nd line is Working directory where application code will copy

3rd line of first dot (.) => Current directory folder structure

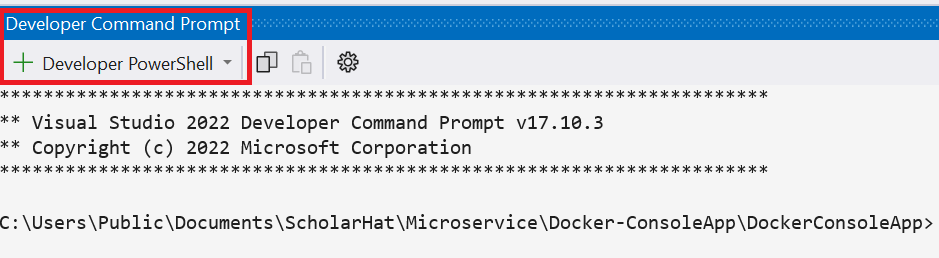
And second dot (.) is the destination folder where the all the files will copy. Here files will copy in app folder working directory.

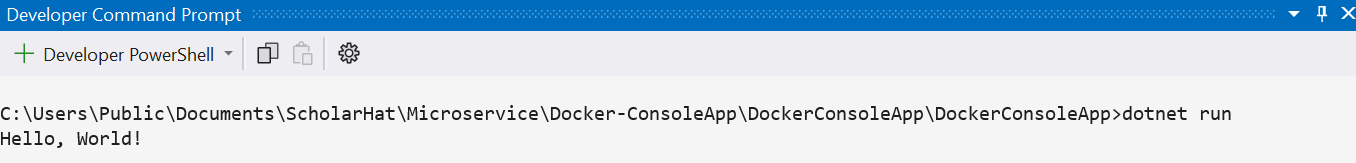
4th line => Entry point to execute the application.

We can run the Docker file using VS



Run the Docker file using Developer Command Prompt ( Right Click on Project 🡪 Open in Terminal and Select the Developer Command Prompt option from Developer PowerShell





Visual Studio by default create the image. We can check image and container in Docker Desktop as well.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

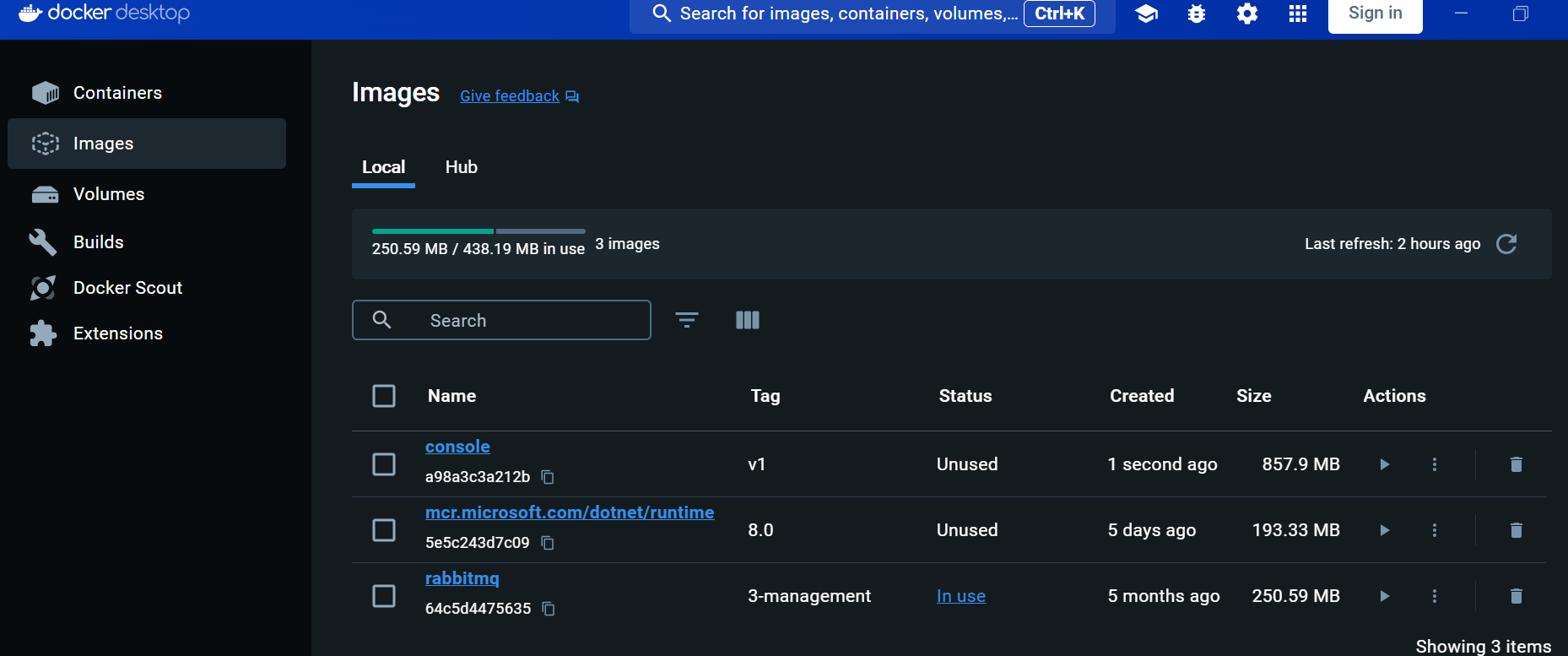
Delete the image and container from Docker Desktop. We can create (build) image and container our self.

**docker -t <ImageName>: version\_number <current working directory path i.e .>**

We can create the image as below command.



The image (console) is created as below screenshot



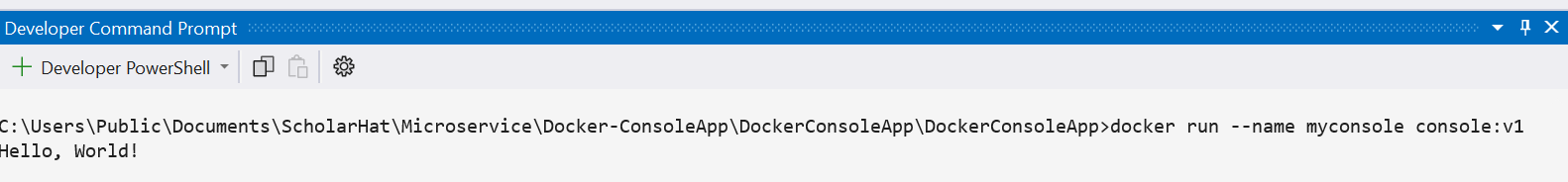
To see the list of image use below command

**docker image ls**

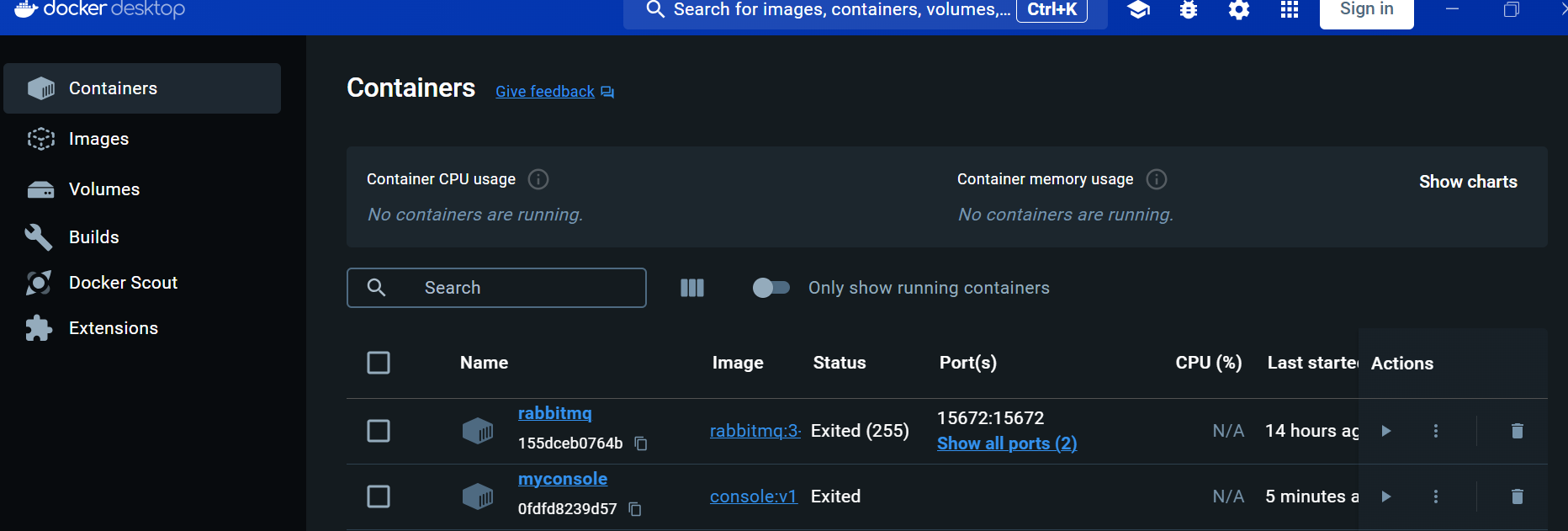
Now run the images through container by using below command

**docker run - - name <container-name> <image-name>: <version>**

**Note:** Console application don’t have port, so not mentioned the port.



Running container can see in Docker Desktop as below. (Container is a running instance of an image)



**Note :** We can create multiple container for the same image.

# **3. SECTION – 2.1 : Create Image for .Net Core Application**

1. Create New Project 🡪 Select ASP.NET Core Web App (Model-View-Controller) template.

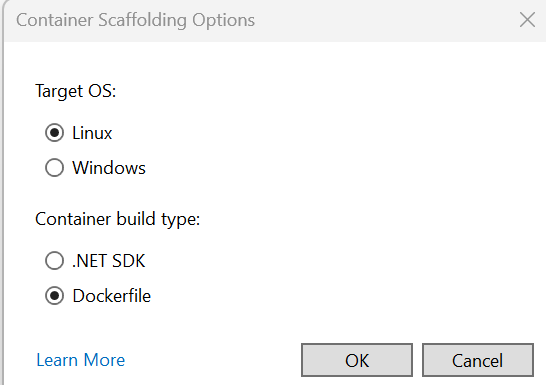
A screenshot of a computer

Description automatically generated

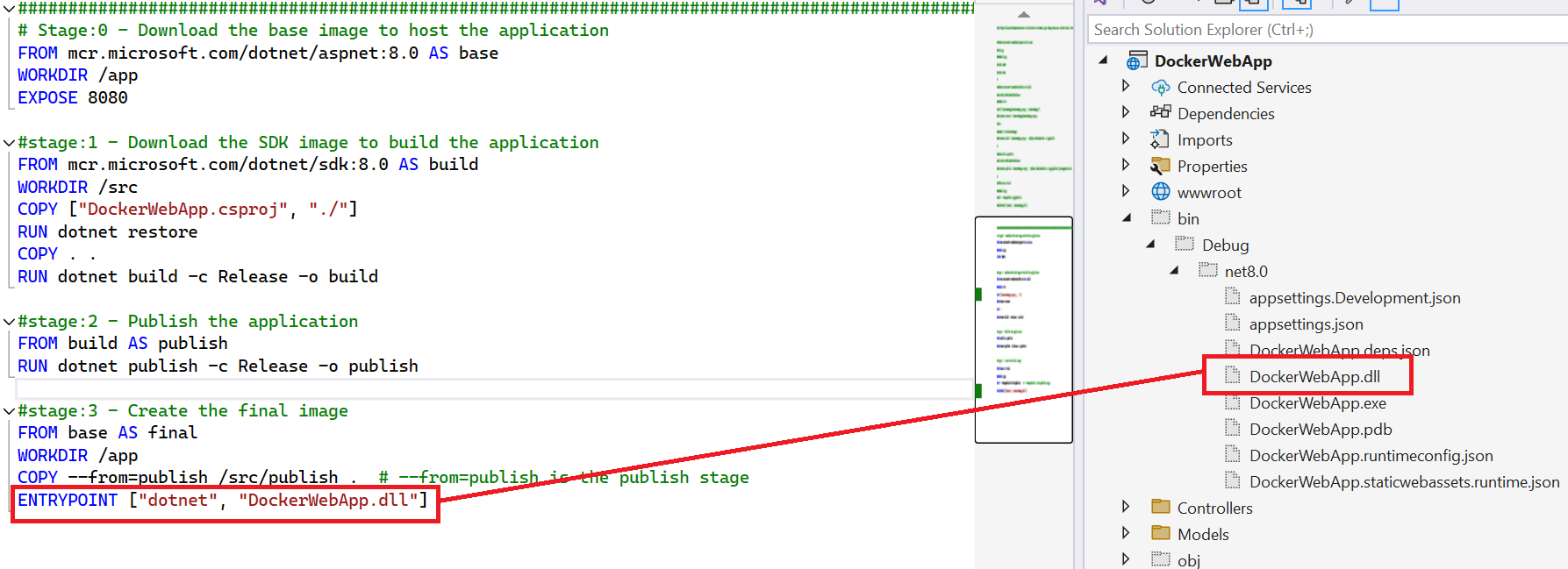
A screenshot of a computer

Description automatically generated

1. Right click on Project 🡪 Add 🡪 Docker Support 🡪 select the below option is will create the Docker file with default commands



Replace the default commands with the below command.



1. Right click on Project 🡪 Open in Terminal 🡪 Select the Developer Command Prompt option from Developer PowerShell

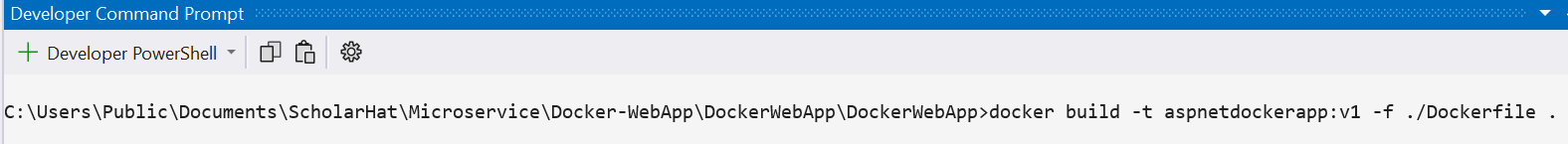
If the docker file in the same location, then use below command.

**docker -t <ImageName>: version\_number <current working directory path i.e .>**

**or**

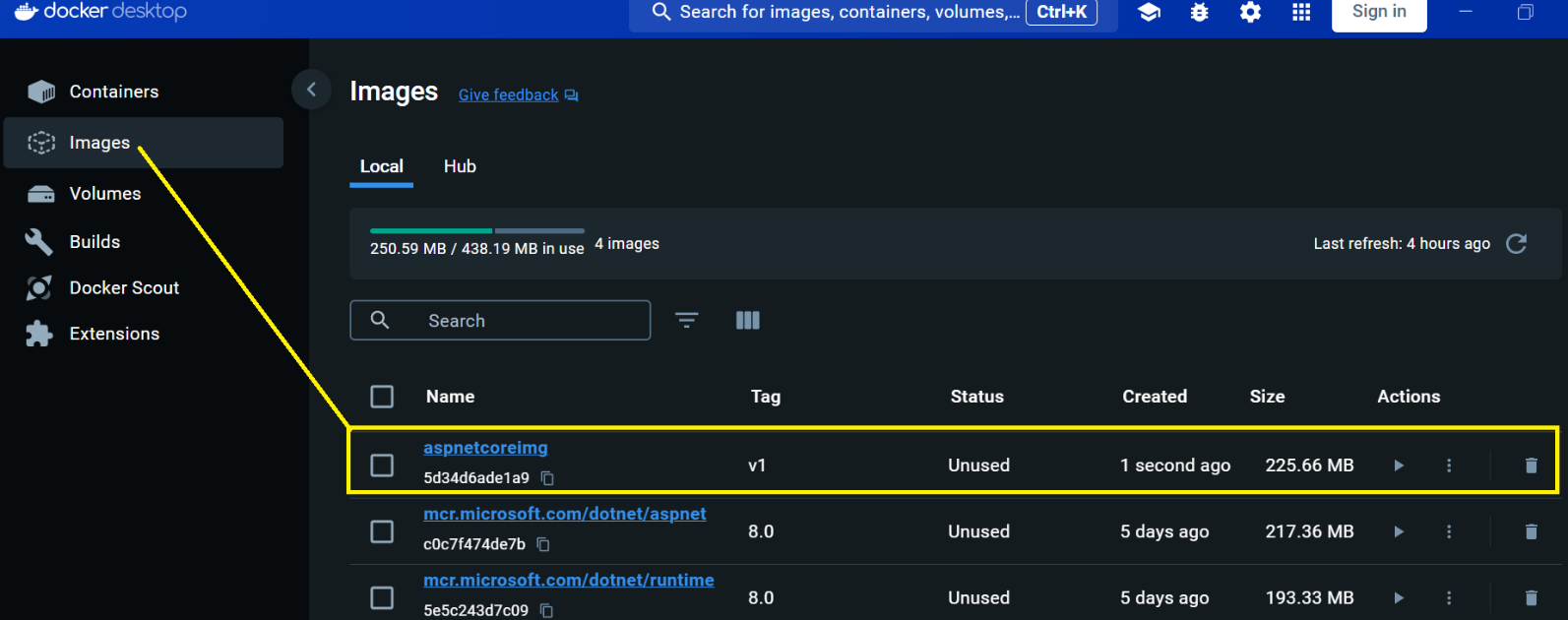
(If docker file is in some other location, then specify the path as follows) Build the image from some other location where we don’t have docker file.

**docker -t <ImageName>: version\_number -f ./Dockerfile <current working directory path i.e .>**

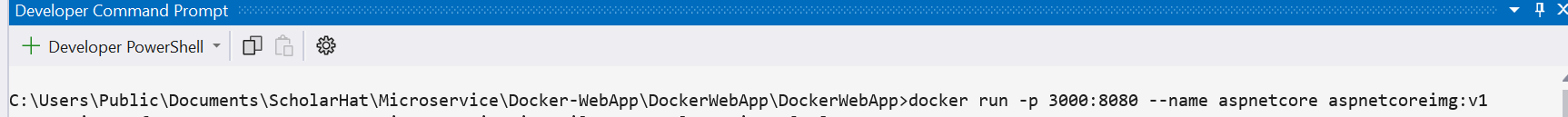
****

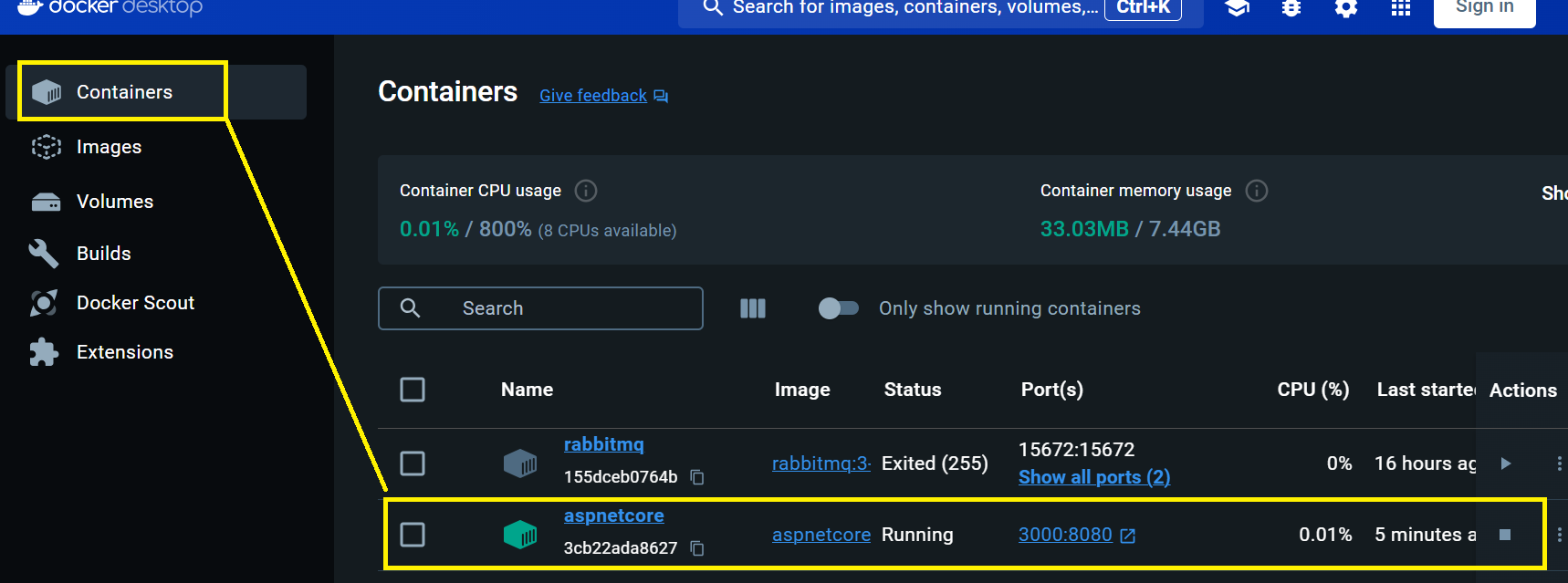
Here simply used the below command

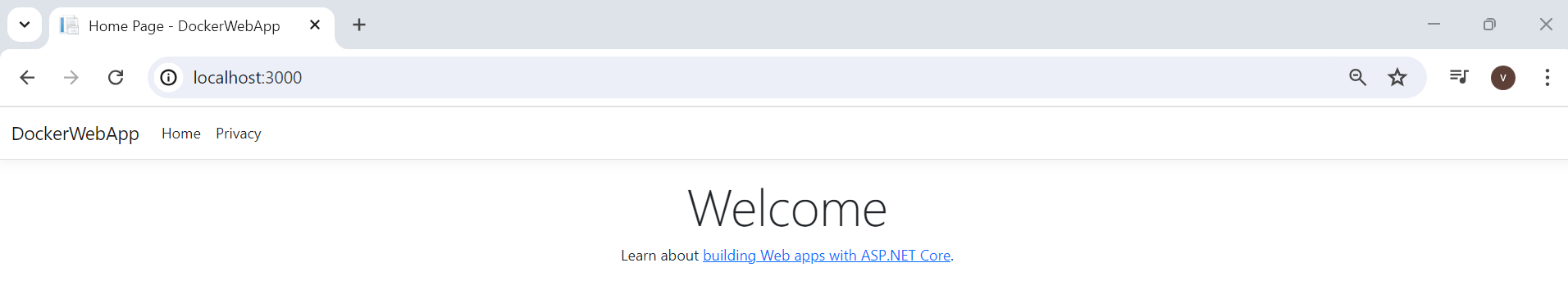




1. Run the below command for container run







# **4. SECTION – 2.2 : Container for SQL Server Image**

A screenshot of a computer

Description automatically generated

“ACCEPT\_EULA” => Accept User Level Agreement. (EULA)

5020 => External SQL Server Port.

1433 => Internal SQL Sever Port.

1. Pull the below image explicitly in command prompt.

A screenshot of a computer

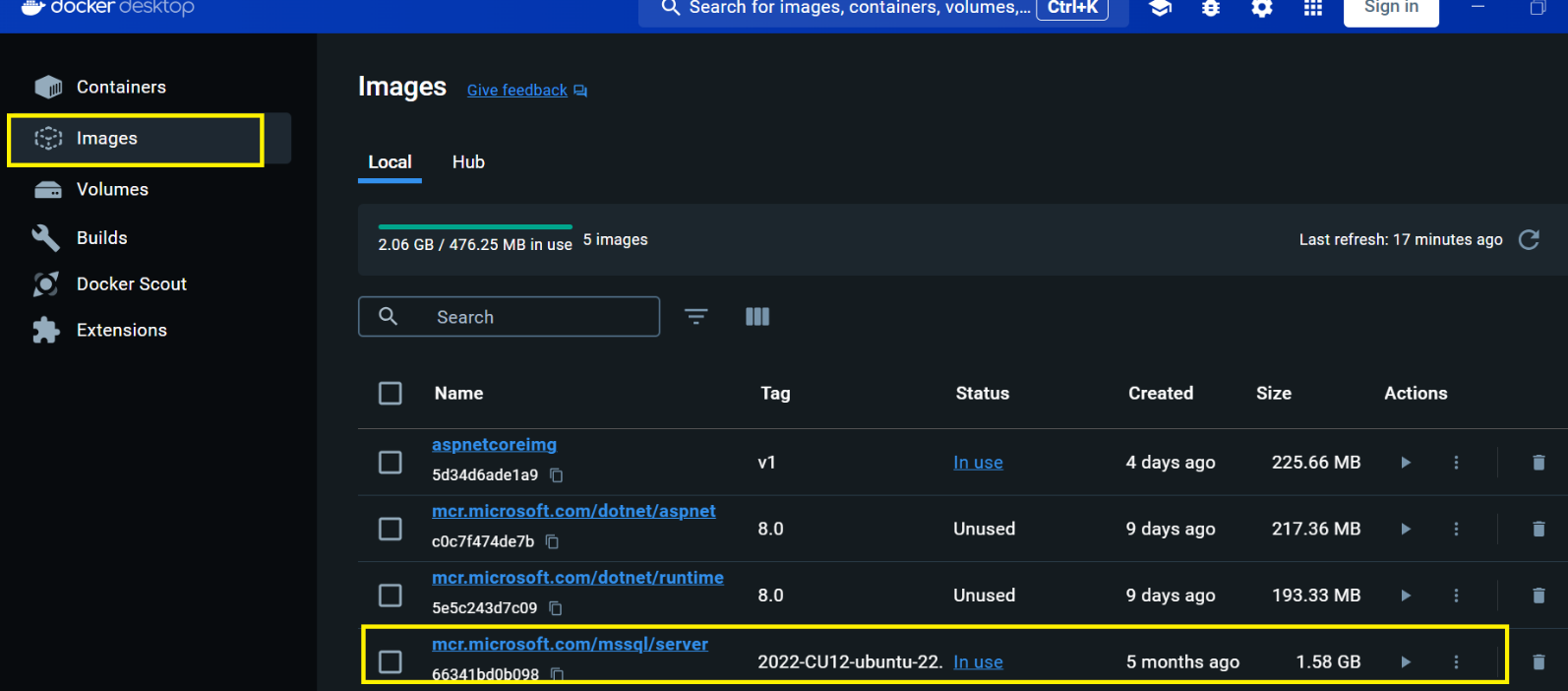
Description automatically generated

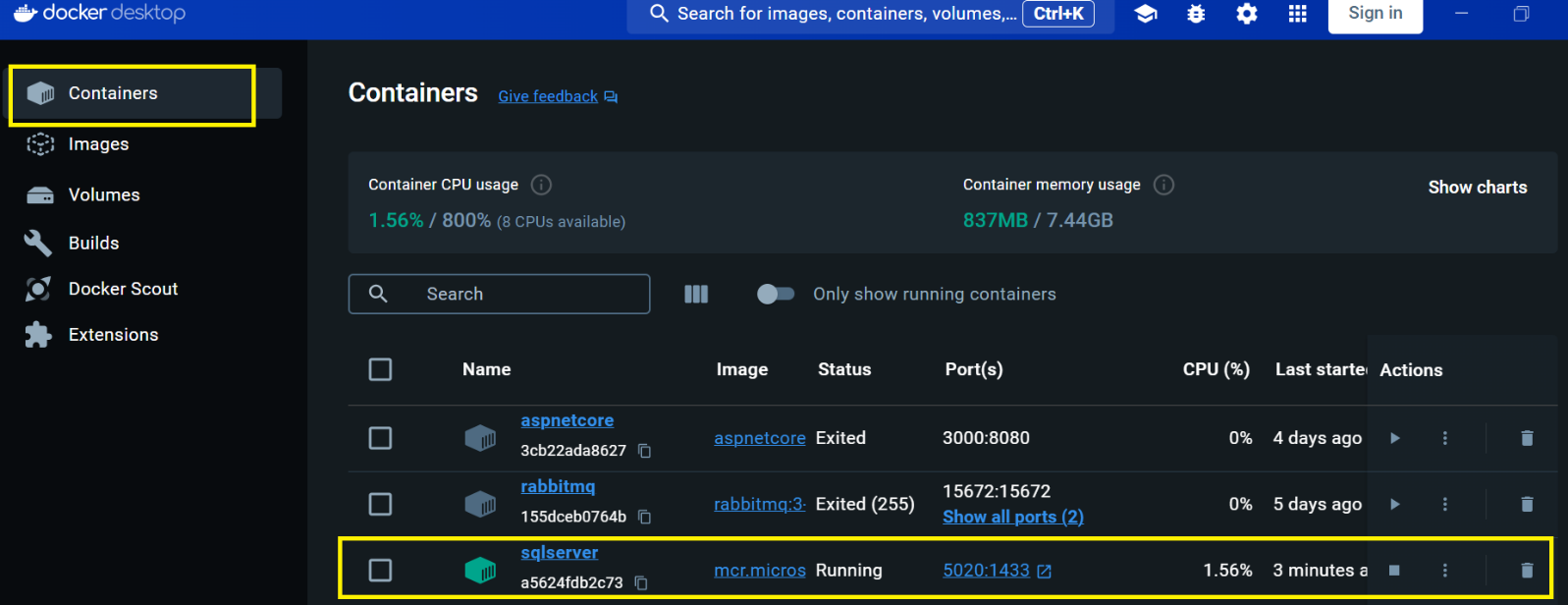
1. Run the container in command prompt.

A black screen with white text

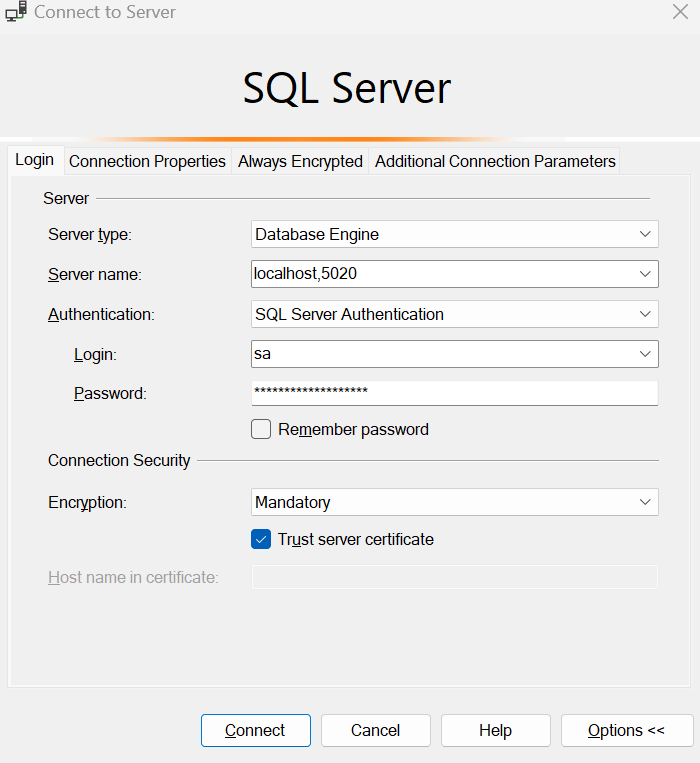
Description automatically generated

**Note :** Run the SQL Server inside a container for Dev / Test / Stage environment and not for production environment.



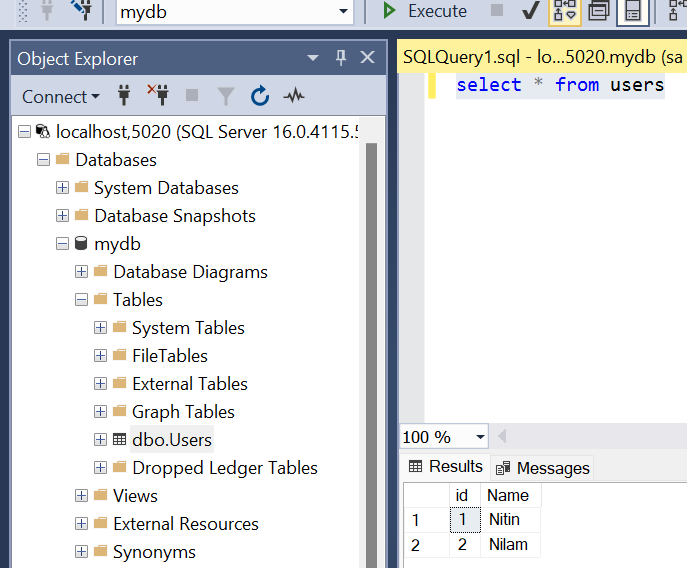


1. Connect through SQL Server Management Studio with below details.



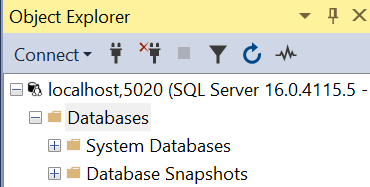
1. Create the Database named **mydb** and create the table named **Users** and insert some records in table.

If we delete the sqlserver container, then Read and Write layer also deleted.



A screenshot of a computer

Description automatically generated

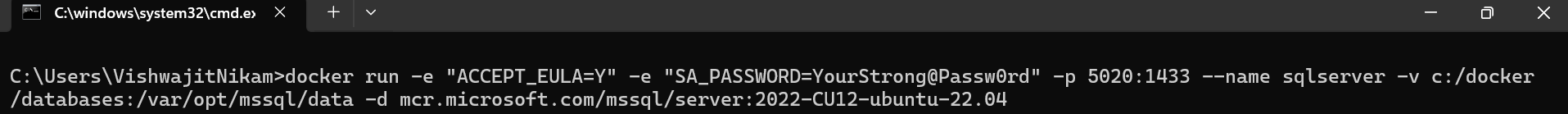


If run the docker command for sqlserver container again, it will recreate everything again from scratch.

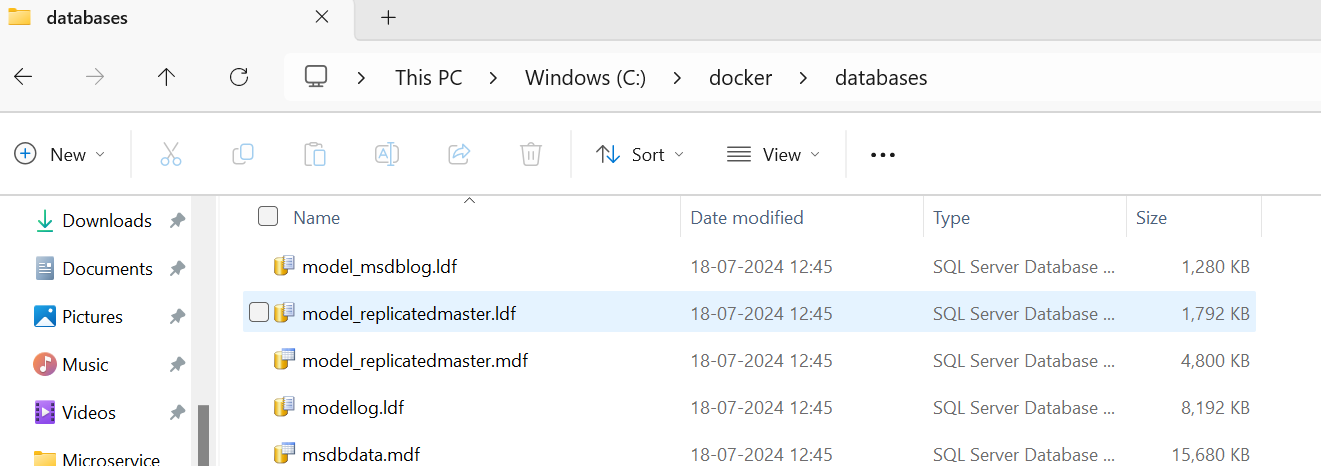
A screen shot of a computer

Description automatically generated

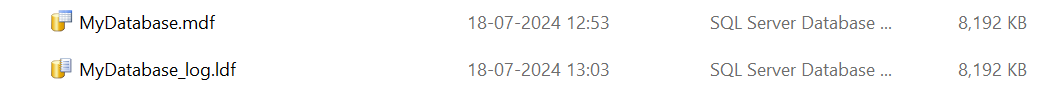
1. Now delete the sqlserver docker container from docker desktop, and and setup and run the below docker container (**Volume** based) for sqlserver with same externa port mapping in command prompt. (In above screenshot we use 5030. We can use any port as external port mapping.)



1. It generates the database in the following folder path.

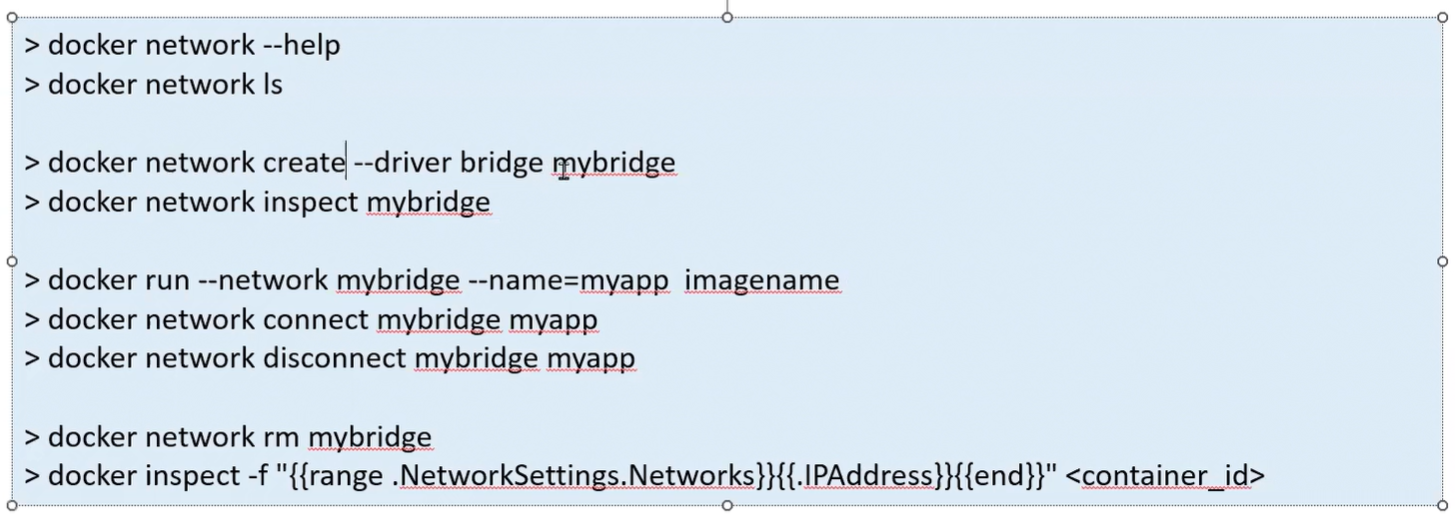


Connect with localhost,5020 with same credential as we did before in Step 3 and 4. The new database is created in the same path. (C:/docker/databases)



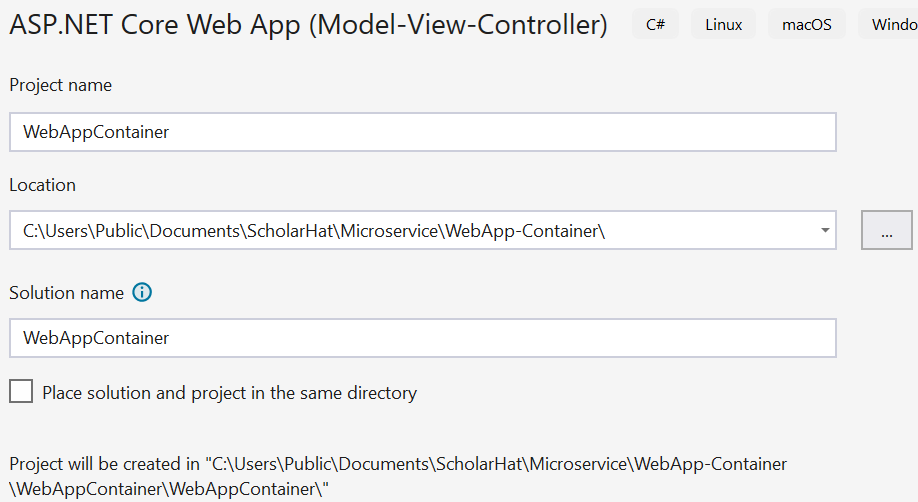
Now if you delete the sqlserver container from docker desktop, the above folder keep remains as it is.

If setup and run the sqlserver volume container again with the same external port mapping ( same as per Step 5 command) in command prompt ,then can access the database again.



# **4. SECTION – 3 : Run 2 Containers for SQL Server and Web App**

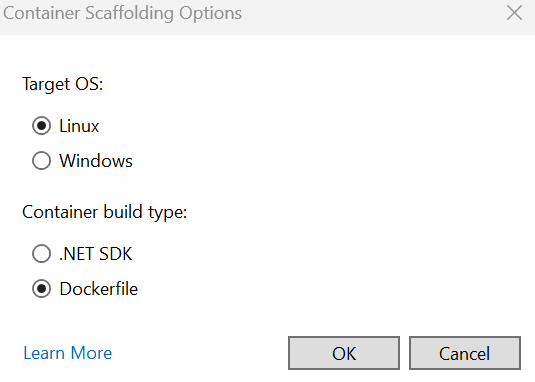
1. Create ASP.Net Core MVC application (Choose ASP.Net Core MVC Template) 🡪 Next



A screenshot of a computer

Description automatically generated

Right Click on Project 🡪 Add 🡪 Docker Support



Automatically created the images and container in docker desktop. Now delete the container.

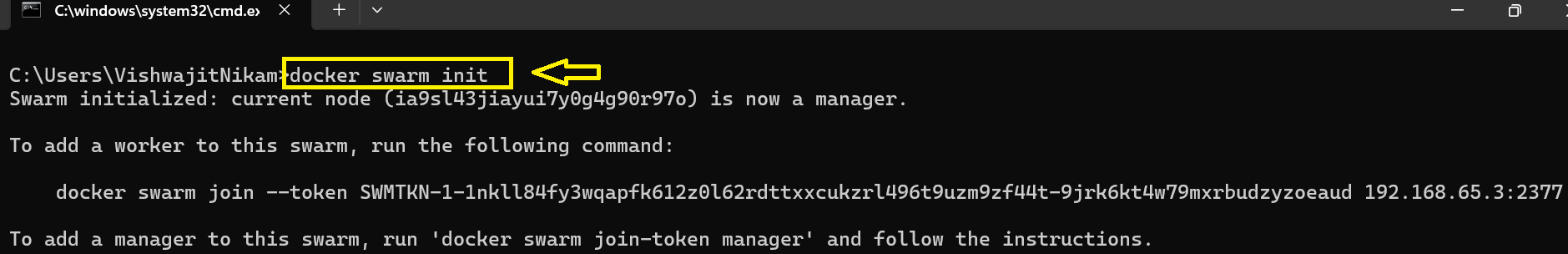
Right click on Project 🡪 Add 🡪 Container Orchestrator Support

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

1. Enable the docker swarm by run the below command in command prompt.



Open the docker-compose.yml file.



One subnet for **ipvm4** setting and another subnet for **ipvm6** settings.

Latest devices **ipvm6** settings.

Right click on Project 🡪 Open in Terminal 🡪 Select the **Developer Command Prompt** from the Developer Power Shell dropdown.

A screenshot of a computer

Description automatically generated

Add the details as per the docker-compose.yml file to connect to SSMS.



Create New MyDb database under the above connection and run the schema from the Database > SQL-Schema folder from solution and under the MyDb database.

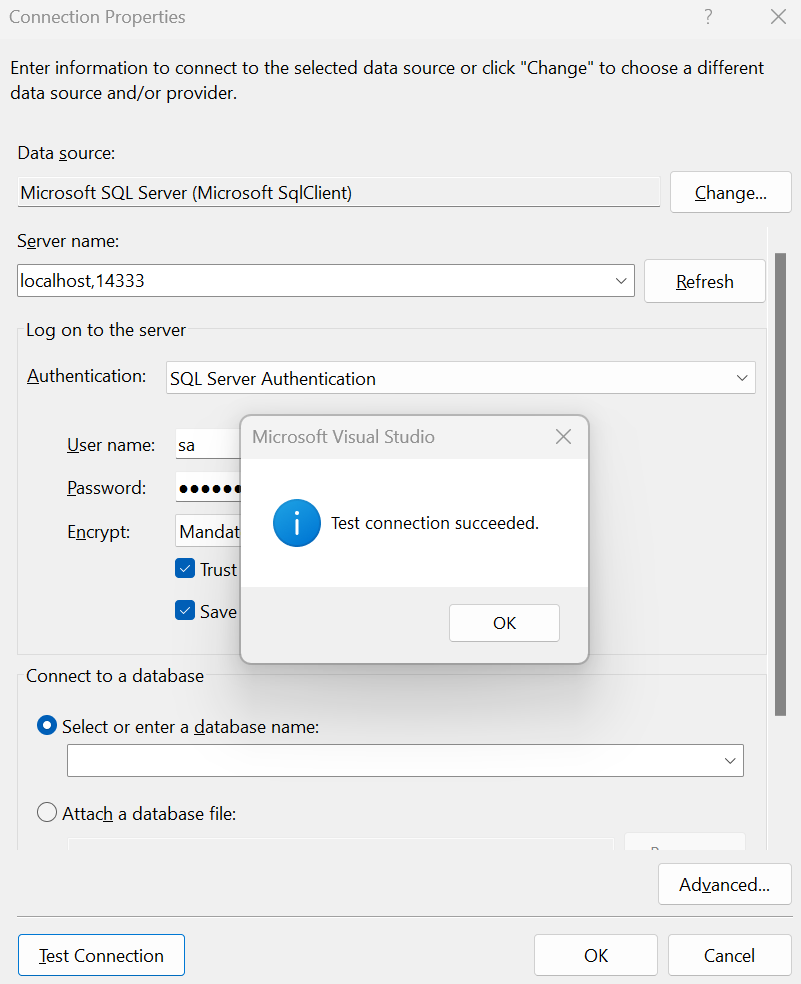
Follow the Data First Approach.

1. Right click on Project 🡪 EF Core Power Tools 🡪 Reverse Engineer

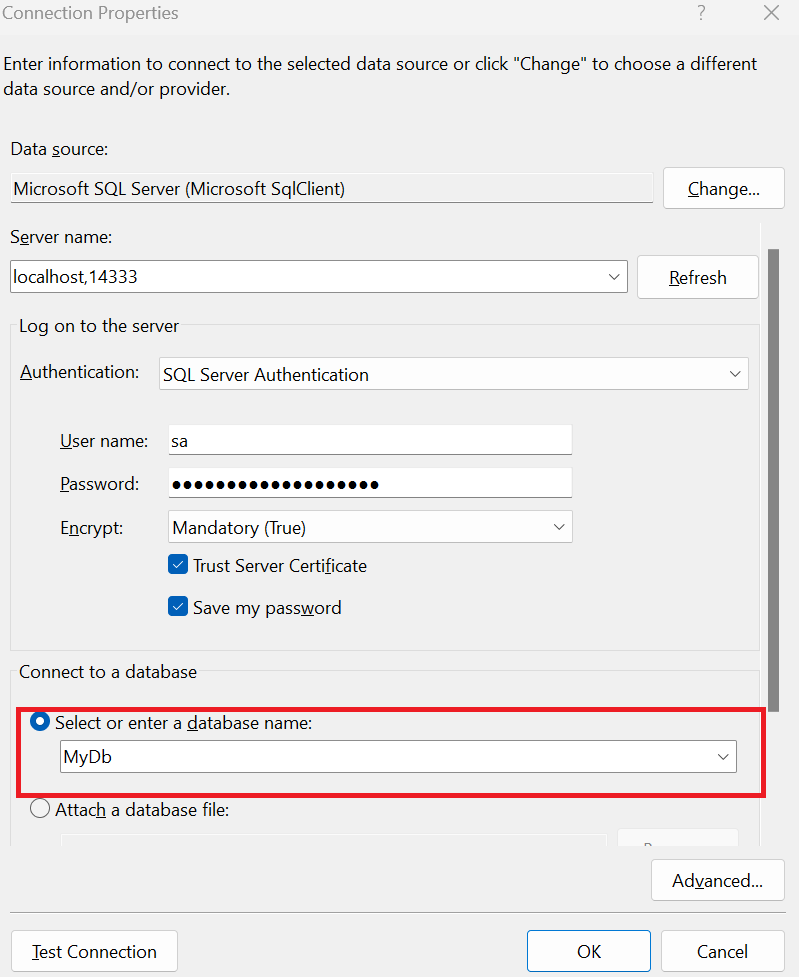
A screenshot of a computer error

Description automatically generated

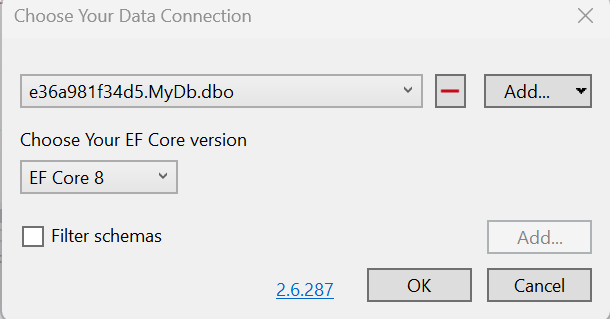
1. Click on Add 🡪 Select Add Database Connection and enter the same connection details as per the docker-compose.yml file as we did to connect to SSMS.



Select Or enter a database name as (MyDb) as we created in SSMS.



Click Ok



Click Ok

A screenshot of a computer

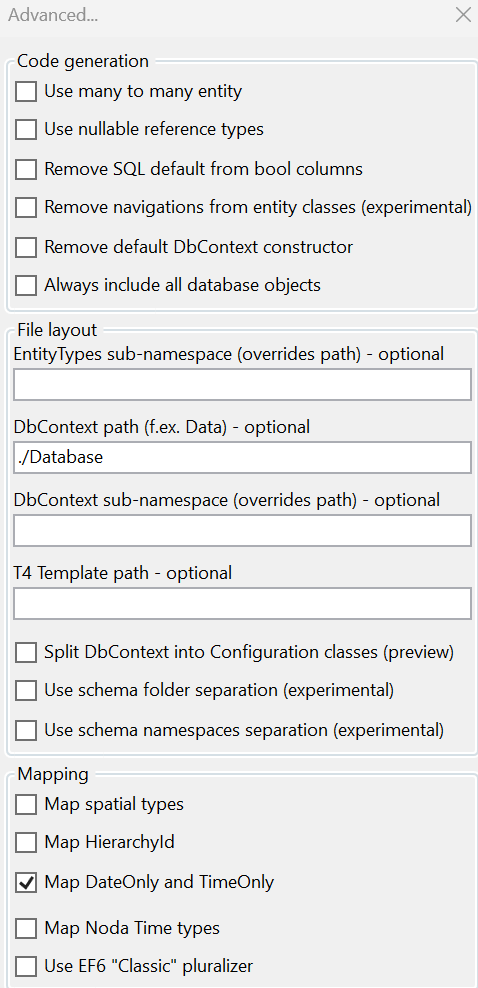
Description automatically generated

Click Ok

A screenshot of a computer

Description automatically generated

Click 🡪 Advanced



🡪 Ok 🡪 Ok button of previous window.

1. Add the database connection from docker-compose.yml file into launchsetting.json file of WebAppContainer Project.

"Container (Dockerfile)": {

"commandName": "Docker",

"launchBrowser": true,

"launchUrl": "{Scheme}://{ServiceHost}:{ServicePort}",

"environmentVariables": {

"ASPNETCORE\_HTTPS\_PORTS": "8081",

"ASPNETCORE\_HTTP\_PORTS": "8080",

"DB\_CONNECTION": "Server=db,1433;Database=MyDb;User Id=sa;Password=YourStrong@Passw0rd;TrustServerCertificate=True"

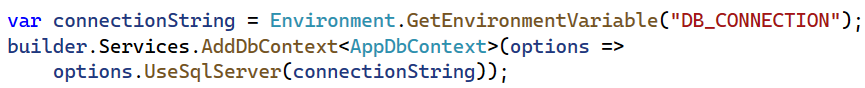
},

"publishAllPorts": true,

"useSSL": true

}

1. Add services to the container as below in Program.cs file of WebAppContainer project.

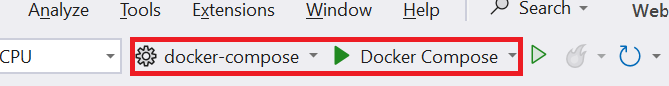


1. Add UserController.cs file and add the action methods Index and Create (GET/POST) verb and the respective views. (Index action method use List template for view through wizard and Create method use Create template for view through wizard)
2. Set the Default routing for User as below

A computer screen shot of a computer

Description automatically generated

1. Run the project through Container (Dockerfile) profile.

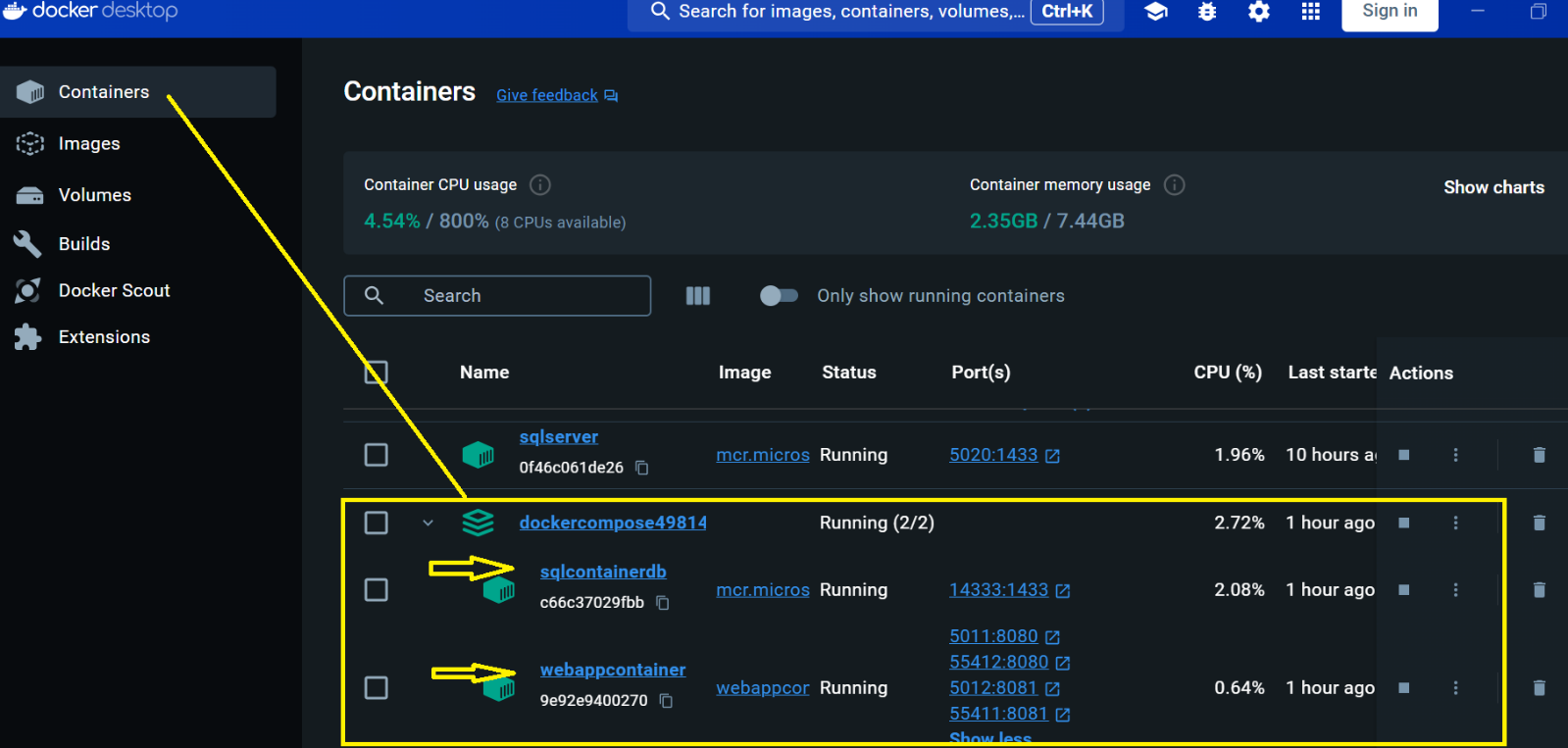


**Note :** During connect to the SSMS, In Additional Connection Parameter Tab add the following parameter. Because we add the same parameter in docker-compose.yml file and launchsetting.json

TrustServerCertificate=True

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated A white background with black text

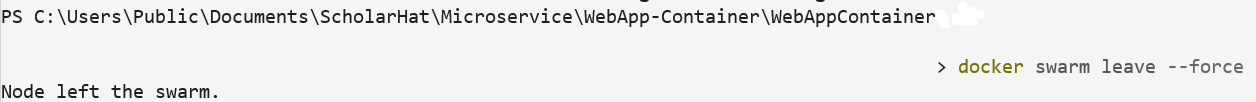
Description automatically generated

1. Delete the running container from Docker Desktop
2. Right click on Solution 🡪 Add 🡪 docker-stack.yml (Stack need the ready made image. It doent contains the build process like docker-compose, doesn’t contain container\_name and mention the driver as driver: overlay)
3. Close the Docker Desktop..

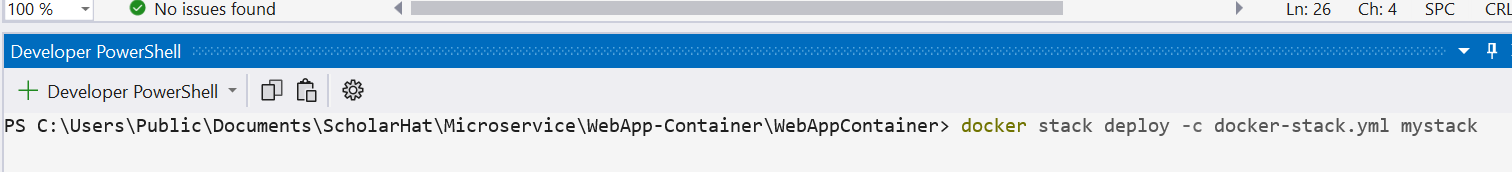
A screenshot of a computer

Description automatically generated

1. Right Click Solution 🡪 Open in Terminal 🡪 Select Developer Command Prompt option from the Developer Power Shell dropdown







Where -c => configuration

Docker-stack.yml => stack file.

mystack => stack file name.