



Steps to create new .NET Core application and then build and run it using Docker

Srilatha Boga

Principal Full Stack Developer

Working at Visionworks, Iselin, New Jersey, United States.

Email: Srilatha0515@gmail.com

Dependencies for an application

- Windows 10 is required for Docker installation.
- Visual Studio 2017 or higher has built-in support for Docker, so this is highly recommended.
- .NET Core SDK
- Docker for Windows
- Docker Tools

Why should we use Microservices instead of a monolithic approach?

Microservices is an approach to develop small services that each run in its own process. We should develop Microservices instead of one service (a monolithic approach) for a multitude of benefits, including:

Microservices are smaller in size

Microservices are easier to develop, deploy, and debug, because a fix only needs to be deployed onto the Microservices with the bug, instead of across the board

Microservices can be scaled quickly and can be reused among different projects

Microservices work well with containers like Docker

Microservices are independent of each other, meaning that if one of the Microservices goes down, there is little risk of the full application shutting down.

Why should we use .NET Core?

.NET Core is a great development tool for many reasons, including that it's open source and is very helpful in developing high-performance and scalable systems. It supports cross-platform runtime, so if one can create a service using .NET Core, it can be run on any platform. .NET Core is also helpful for

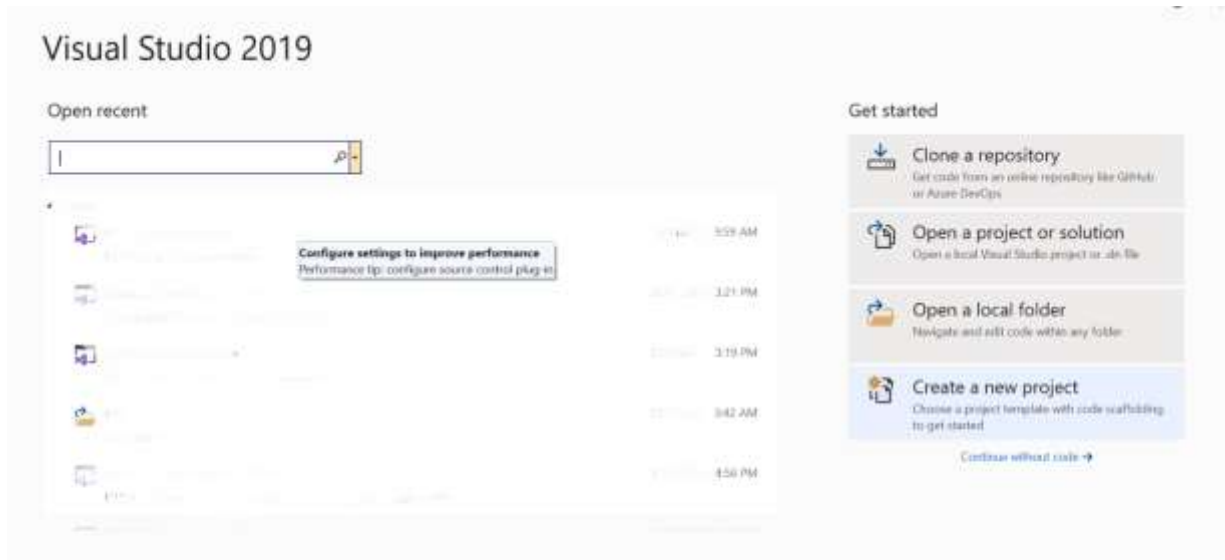
faster development, as well as supporting built-in dependency injection and a cloud-based environment configuration. .NET Core also has Docker support.

Why should we use Docker?

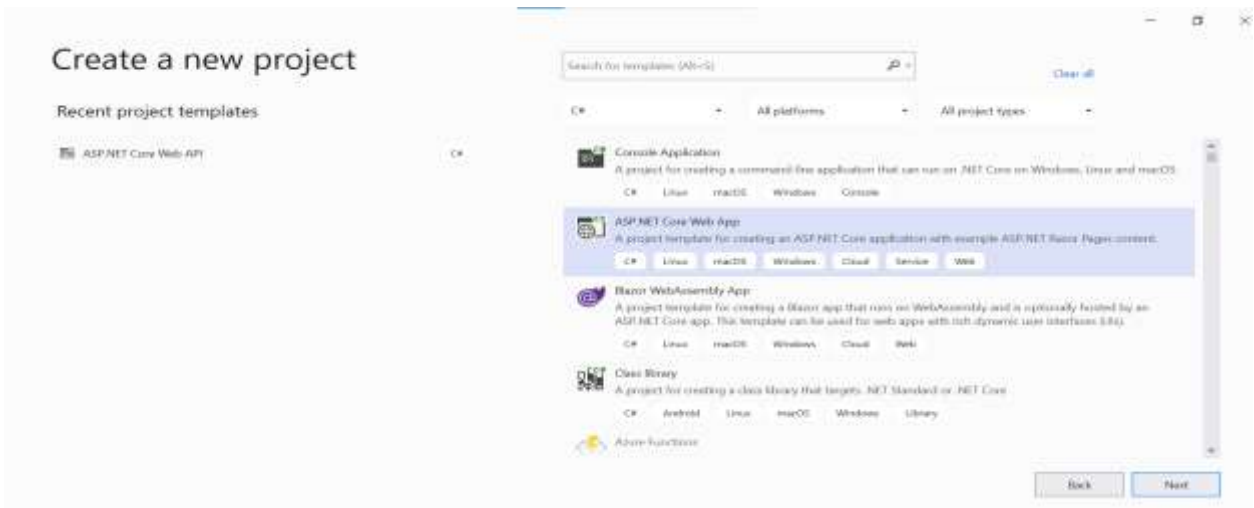
Docker is a tool that makes it easier to create, deploy, and run applications by using a containerization approach. These containers are lightweight and take less time to start than traditional servers. These containers also increase performance and lower cost, while offering proper resource management. Another benefit to using Docker is that a user no longer need to pre-allocate RAM to each container.

How to create a new application using .NET Core and then build and run it using Docker

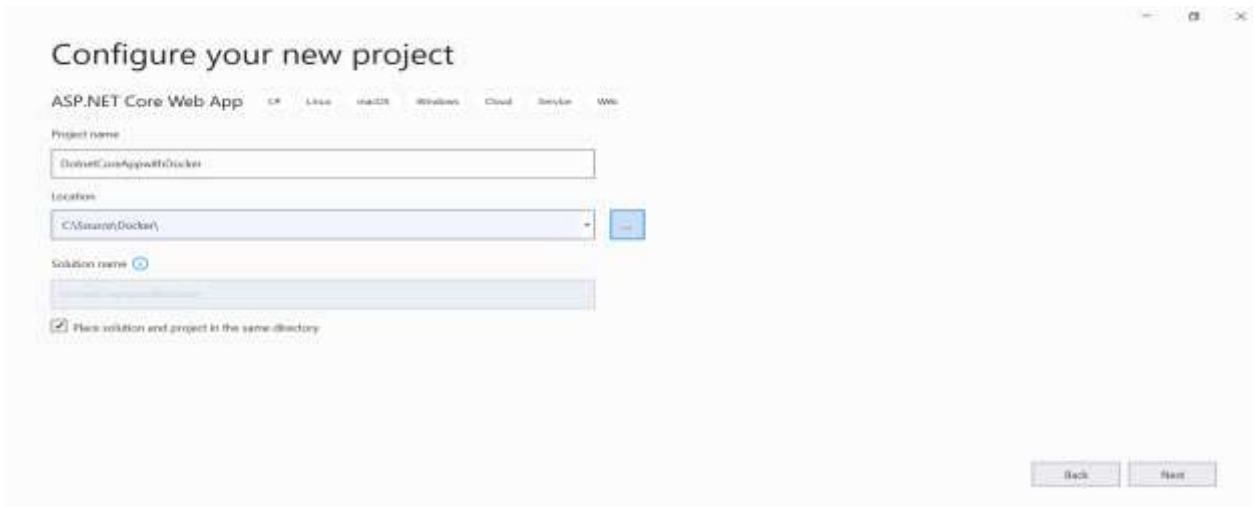
Step 1: Open Visual Studio 2019, then create a new project



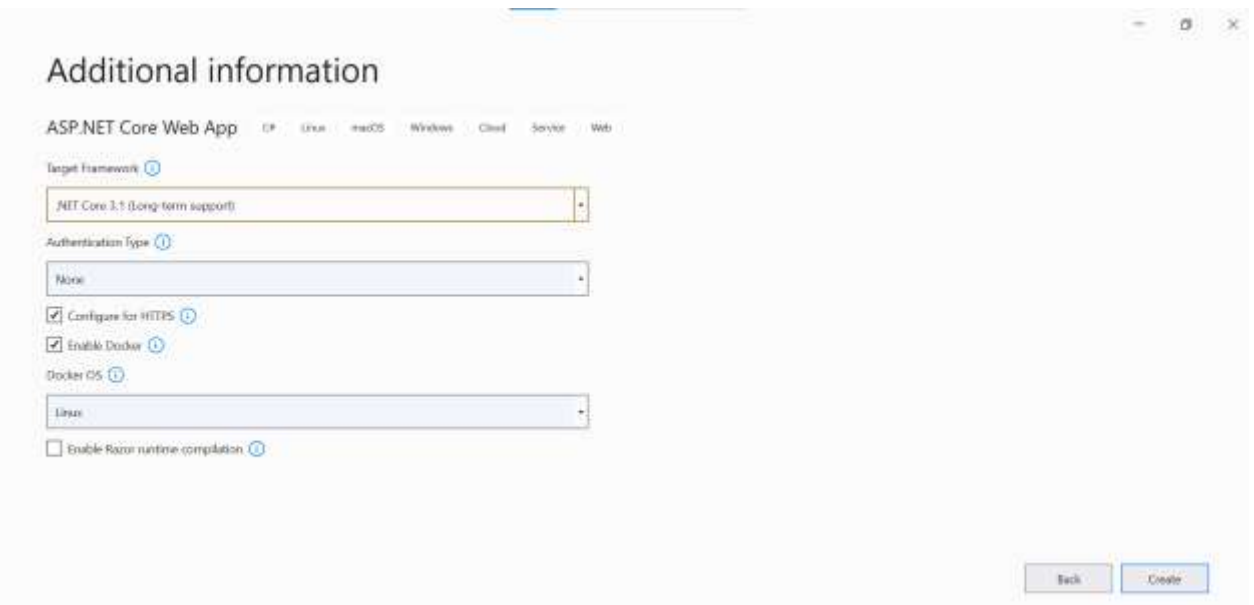
Step 2: Select ASP.Net Core Web App, then click on Next



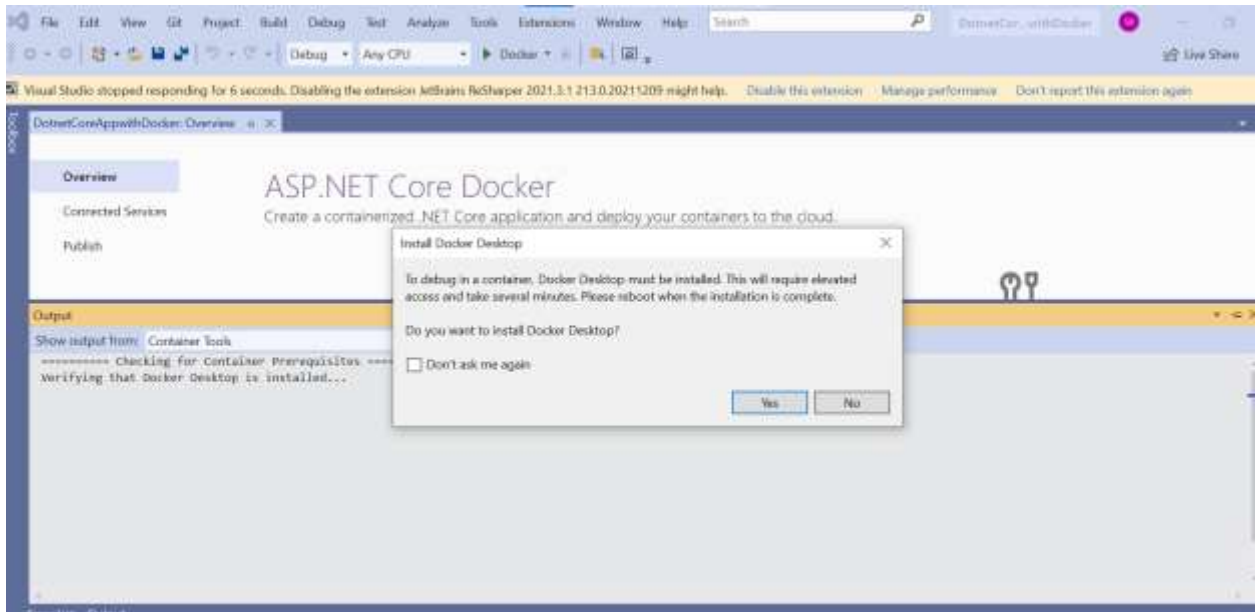
Step 3: Provide the Project Name and Location to save the project

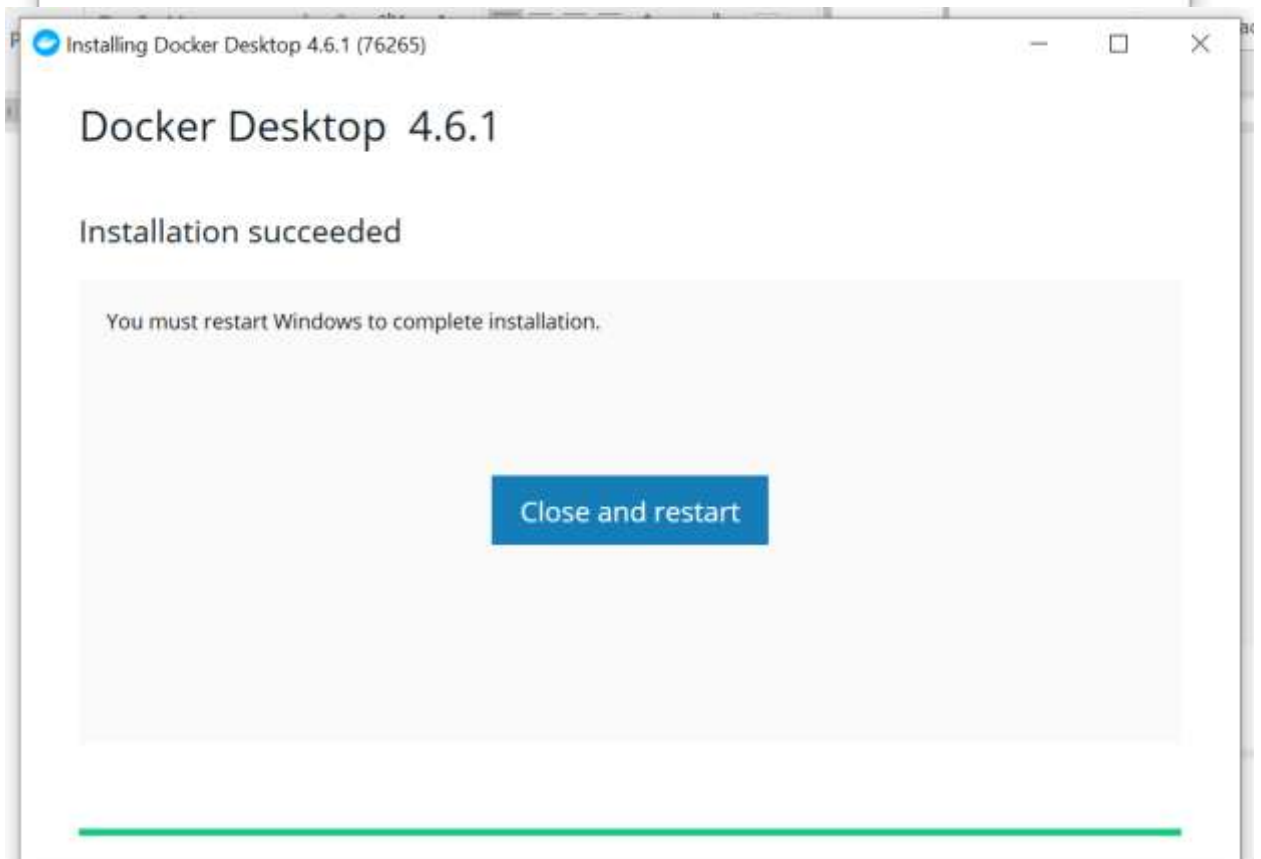



Step 4: Select Target Framework as ".Net Core 3.1" or higher. Also Enable Docker Checkbox then click on "Create"



Step 5(optional): Once the project is created it will prompt to Install Docker, if the Docker is not installed on the machine yet








Our Service Agreement has Changed

- Our Docker Subscription Service Agreement include a change to the terms of use for Docker Desktop.
 - It **remains free** for small businesses (fewer than 250 employees AND less than \$10 million in annual revenue), personal use, education, and non-commercial open source projects.
 - It requires a paid subscription for professional use in larger enterprises.
- The effective date of these terms is August 31, 2021. There was a **grace period** until January 31, 2022 for those that require a paid subscription to use Docker Desktop. Docker trusts our customers to be in compliance and Docker Desktop will continue to function normally after January 31st, but this is a reminder that unpaid commercial use will be out of compliance with the Docker Subscription Service Agreement.
- The existing Docker Free subscription has been renamed **Docker Personal** and we have introduced a Docker Business subscription.
- The Docker Pro, Team, and Business subscriptions include commercial use of Docker Desktop.

I accept the terms

[View Full Terms](#) 

[Decline and Close Application](#) [Accept](#)

Docker Desktop - Install WSL 2 kernel update

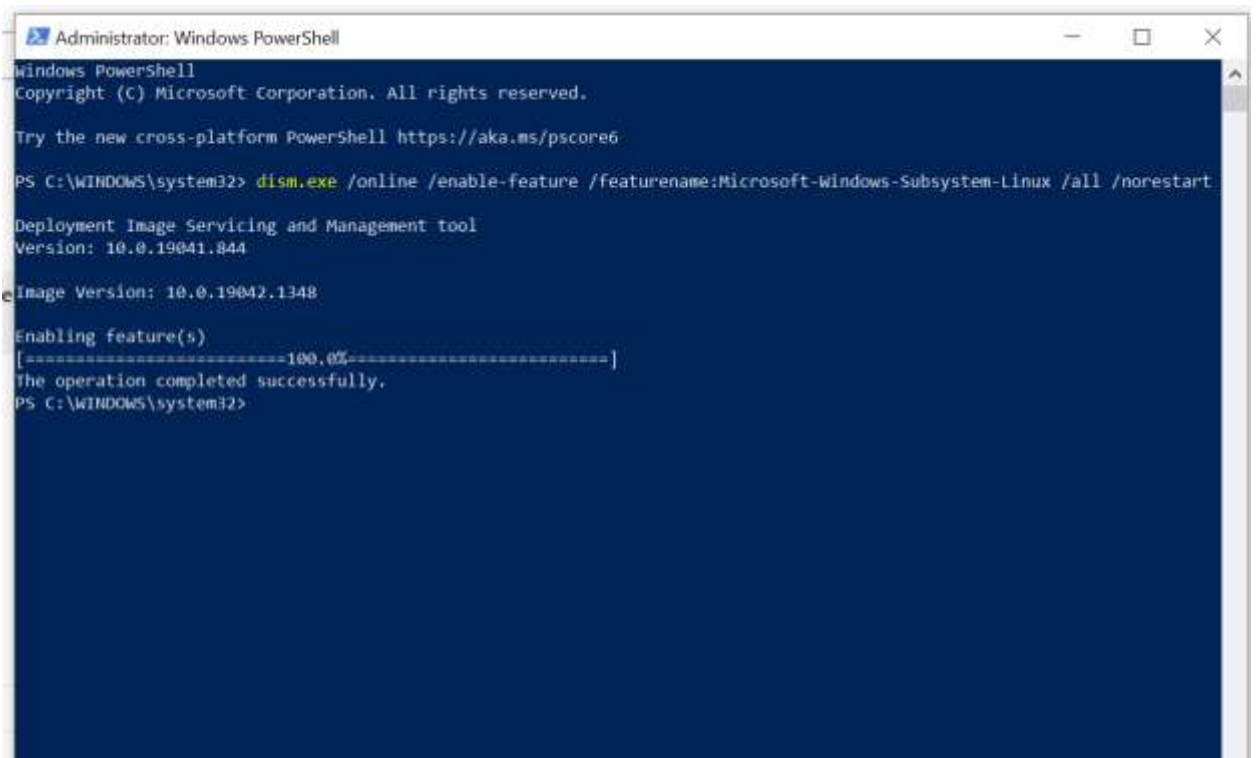
WSL 2 installation is incomplete.



The WSL 2 Linux kernel is now installed using a separate MSI update package. Please click the link and follow the instructions to install the kernel update:
<https://aka.ms/wsl2kernel>.

Press Restart after installing the Linux kernel.

[Restart](#) [Cancel](#)




```
Administrator: Windows PowerShell
Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\WINDOWS\system32> dism.exe /online /enable-feature /Featurename:Microsoft-Windows-Subsystem-Linux /all /norestart

Deployment Image Servicing and Management tool
Version: 10.0.19041.844

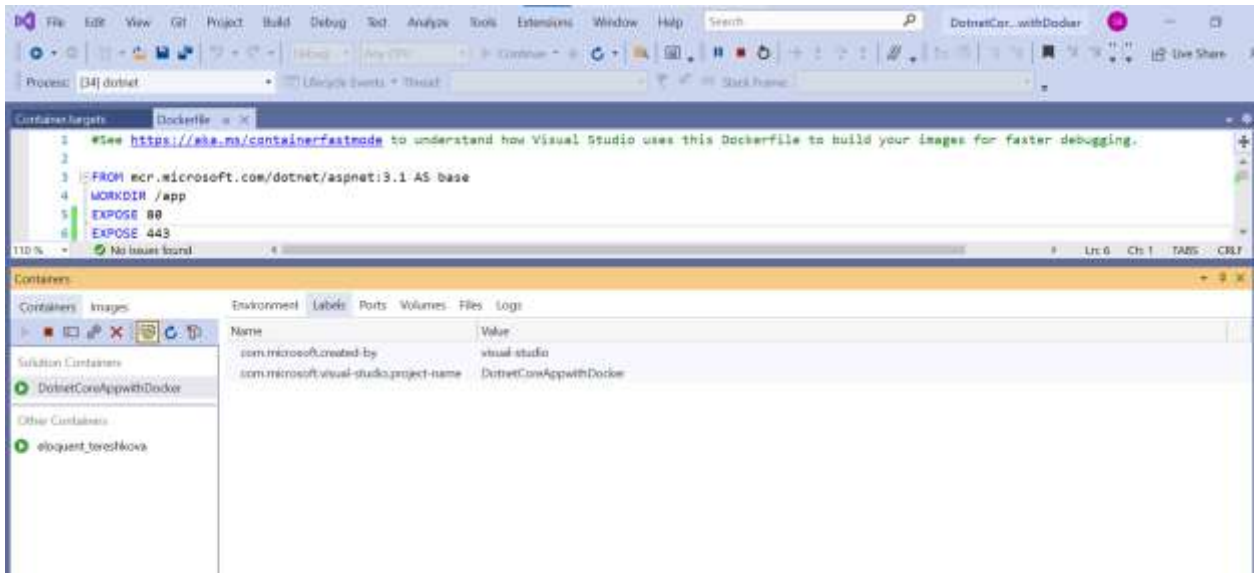
Image Version: 10.0.19042.1348

Enabling feature(s)
[=====100.0%=====]
The operation completed successfully.
PS C:\WINDOWS\system32> dism.exe /online /enable-feature /Featurename:VirtualMachinePlatform /all /norestart

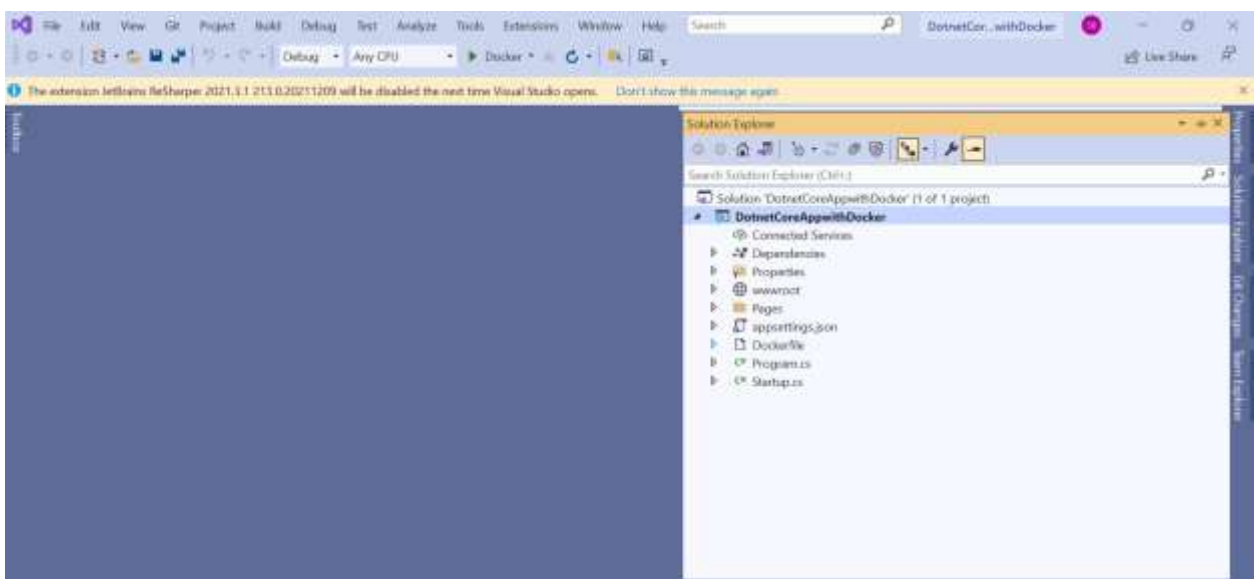
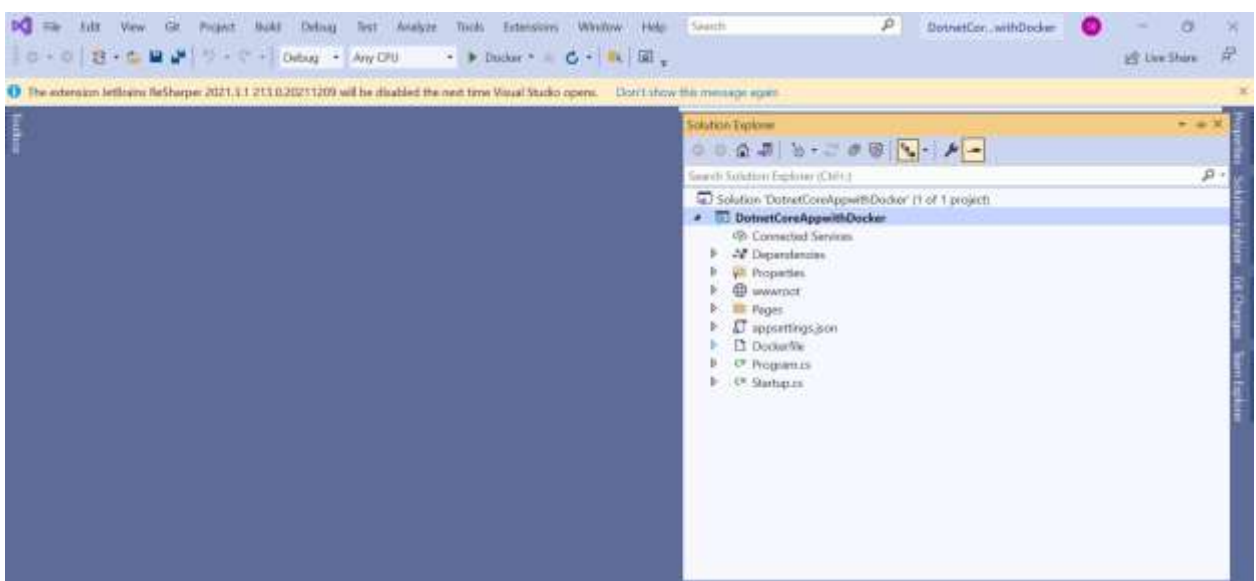
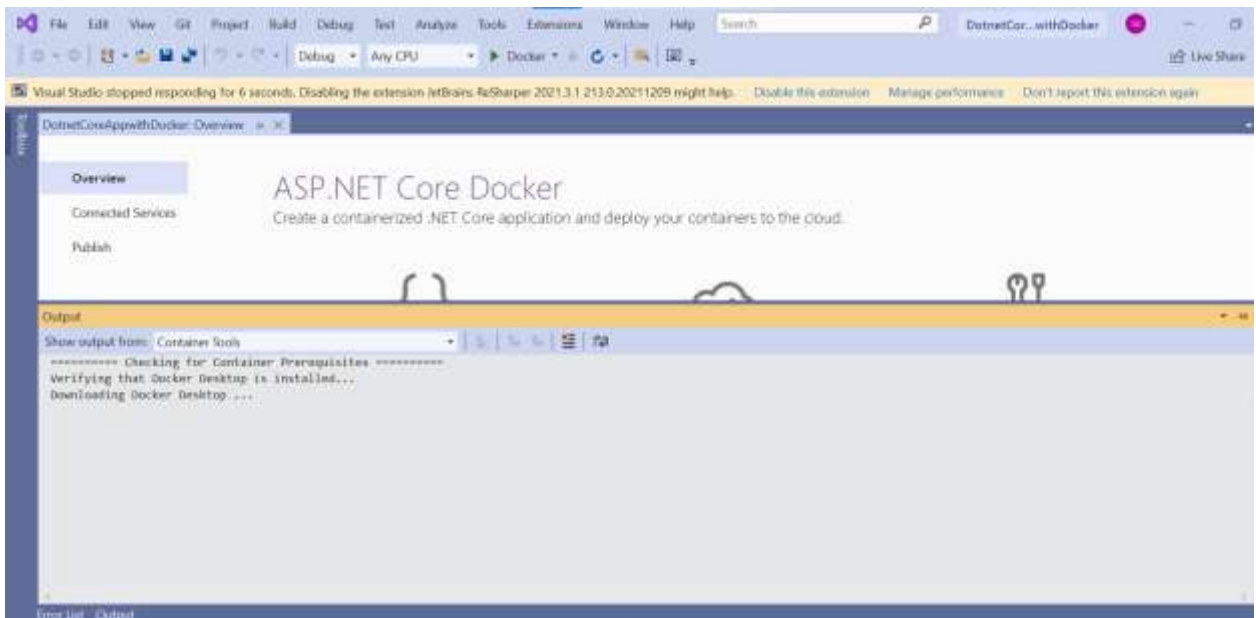
Deployment Image Servicing and Management tool
Version: 10.0.19041.844

Image Version: 10.0.19042.1348

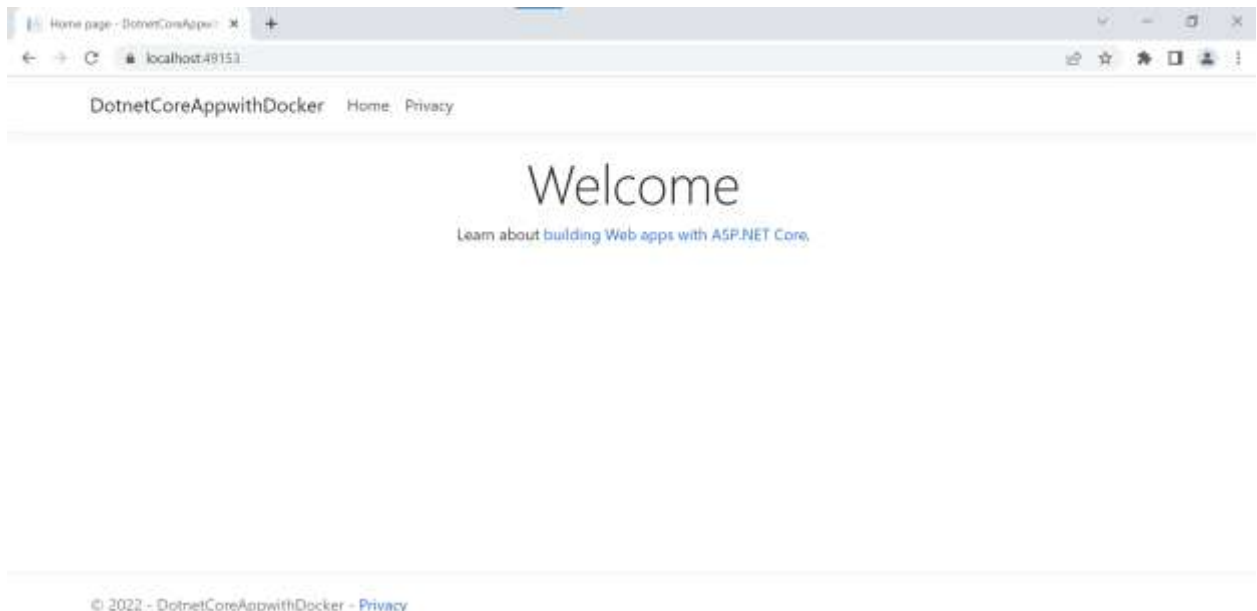
Enabling feature(s)
[=====100.0%=====]
The operation completed successfully.
PS C:\WINDOWS\system32>
```



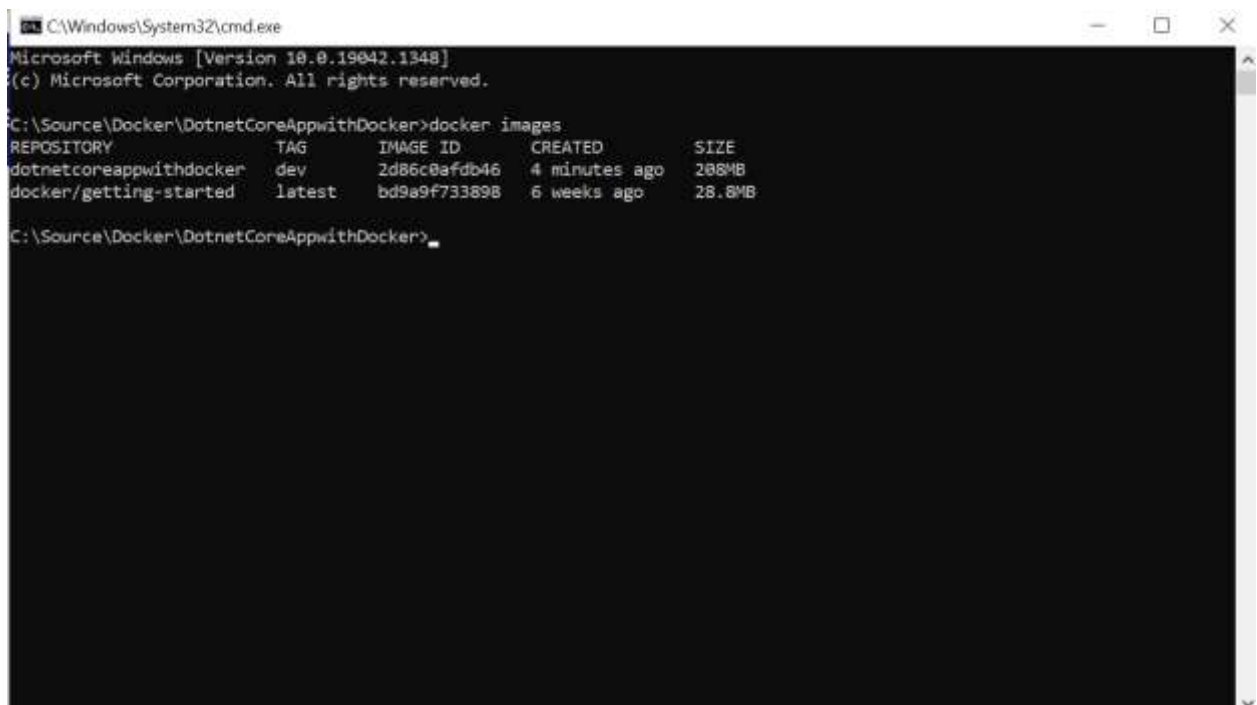
Step 6: Solution will be created with a Dockerfile in the file list



Step 7: Build the solution, application will open in the browser



Step 8: To check the docker images, cmd to the project folder and type the command "docker images"



Here are some of the commands for the docker images

To restore packages: dotnet restore

To publish the application code: dotnet publish -o obj/Docker/publish

To build the image: docker build -t imagename

Check running containers: dockerps

Check all containers: dockerps -a