# 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

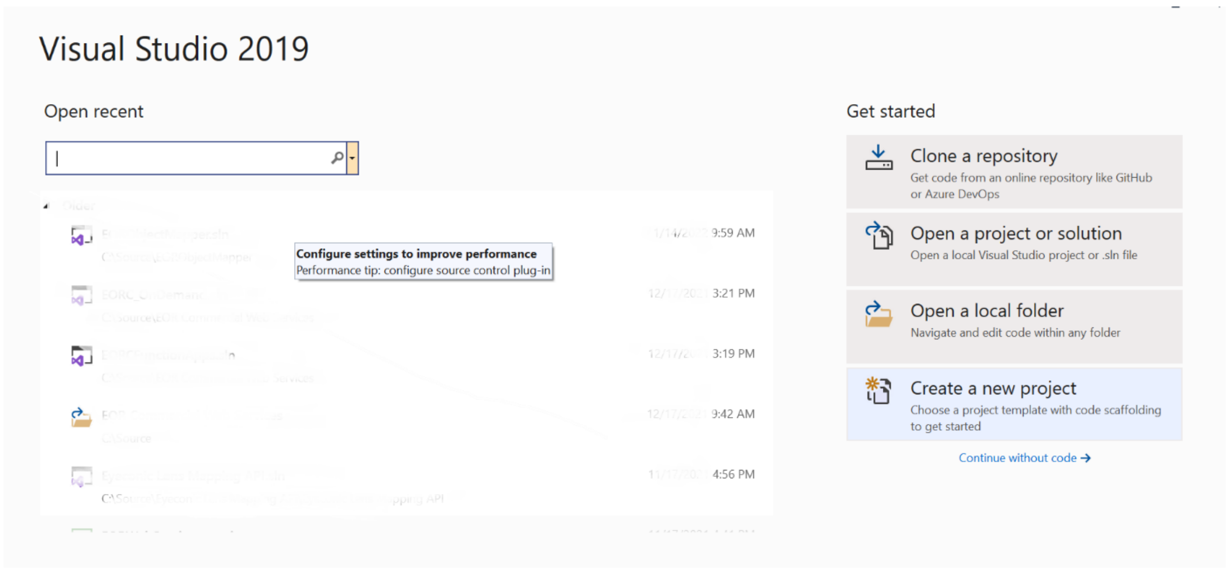
platform. .NET Core is also helpful for faster development, as well as supporting built-in dependency injection and a cloud-based environment configuration. .NET Core also has Docker support.

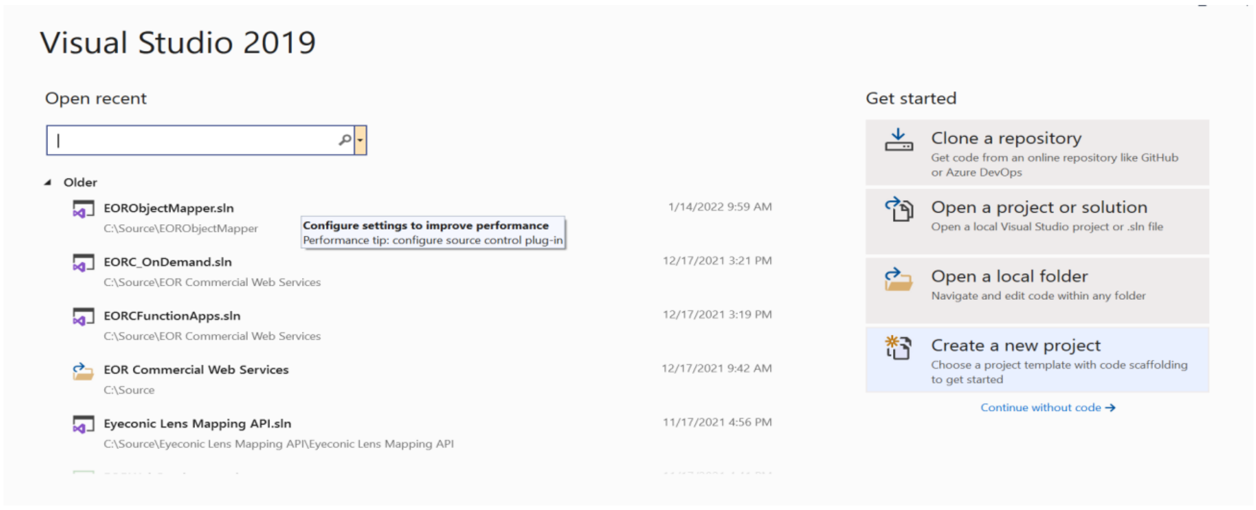
Why should we use Docker?

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.

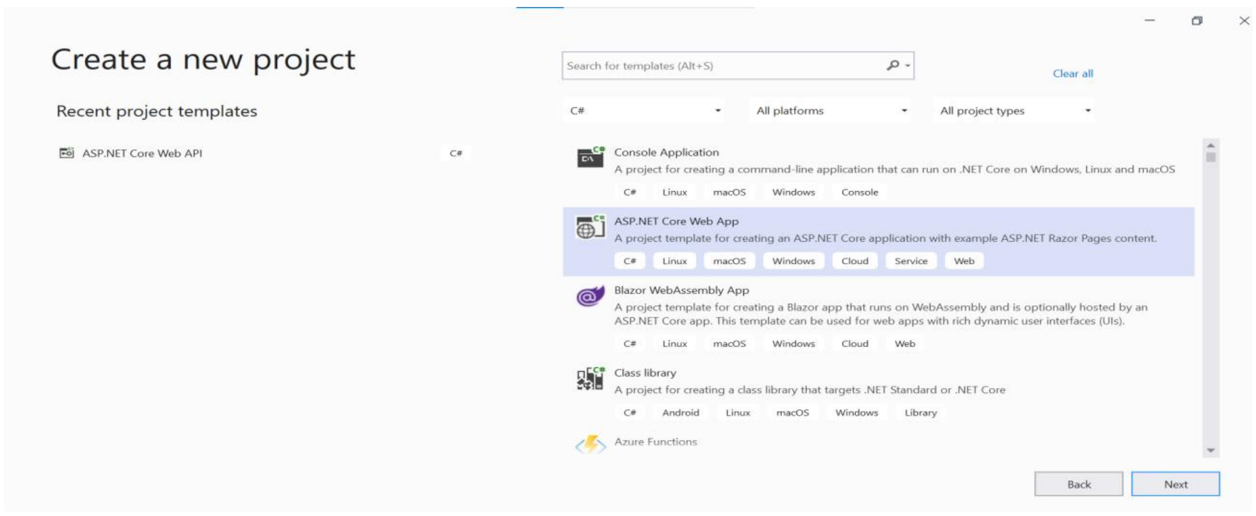
How to create a new application using .NET Core and then build and run it using Docker

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


Visual Studio 2019
Open recent
Configure settings to improve performance
Performance tip: configure source control plug-in
9:59 AM
3:21 PM
3:19 PM
9:42 AM
4:56 PM
Get started
Clone a repository
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →


Visual Studio 2019
Open recent
Older
EORObjectMapper.sln
C:\Source\EORObjectMapper
1/14/2022 9:59 AM
Configure settings to improve performance
Performance tip: configure source control plug-in
EORC_OnDemand.sln
C:\Source\EOR Commercial Web Services
12/17/2021 3:21 PM
EORCFunctionApps.sln
C:\Source\EOR Commercial Web Services
12/17/2021 3:19 PM
EOR Commercial Web Services
C:\Source
12/17/2021 9:42 AM
Eyeconic Lens Mapping API.sln
C:\Source\Eyeconic Lens Mapping API\Eyeconic Lens Mapping API
11/17/2021 4:56 PM
Get started
Clone a repository
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →

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



Create a new project

Recent project templates

ASP.NET Core Web API C#

Search for templates (Alt+S)

Clear all

C# | All platforms | All project types

Console Application
A project for creating a command-line application that can run on .NET Core on Windows, Linux and macOS
C# Linux macOS Windows Console

ASP.NET Core Web App
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
C# Linux macOS Windows Cloud Service Web

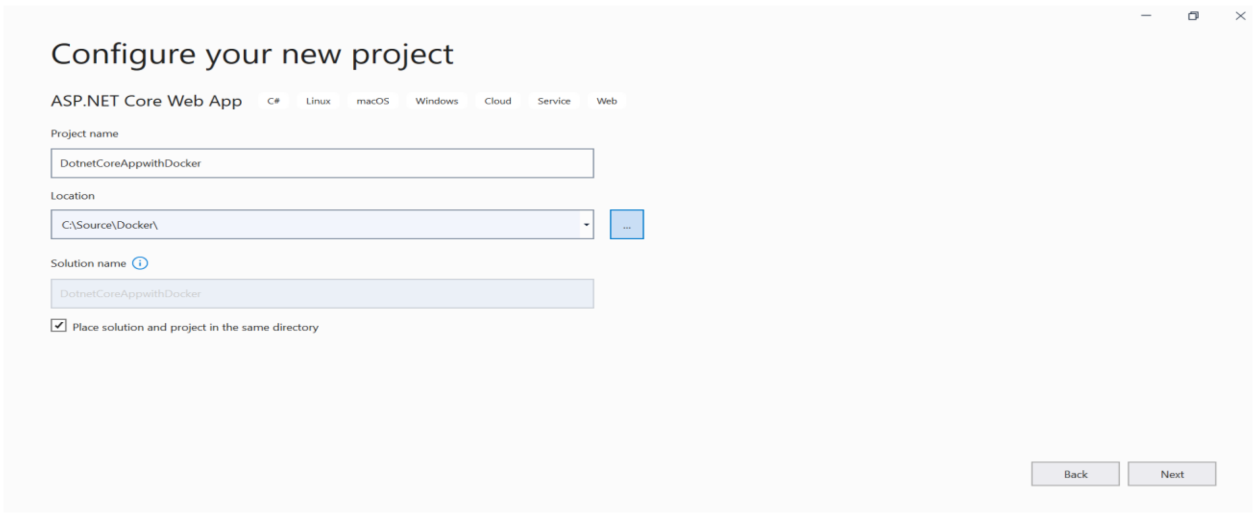
Blazor WebAssembly App
A project template for creating a Blazor app that runs on WebAssembly and is optionally hosted by an ASP.NET Core app. This template can be used for web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Cloud Web

Class library
A project for creating a class library that targets .NET Standard or .NET Core
C# Android Linux macOS Windows Library

Azure Functions

Back Next

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



Configure your new project

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Project name
DotnetCoreAppwithDocker

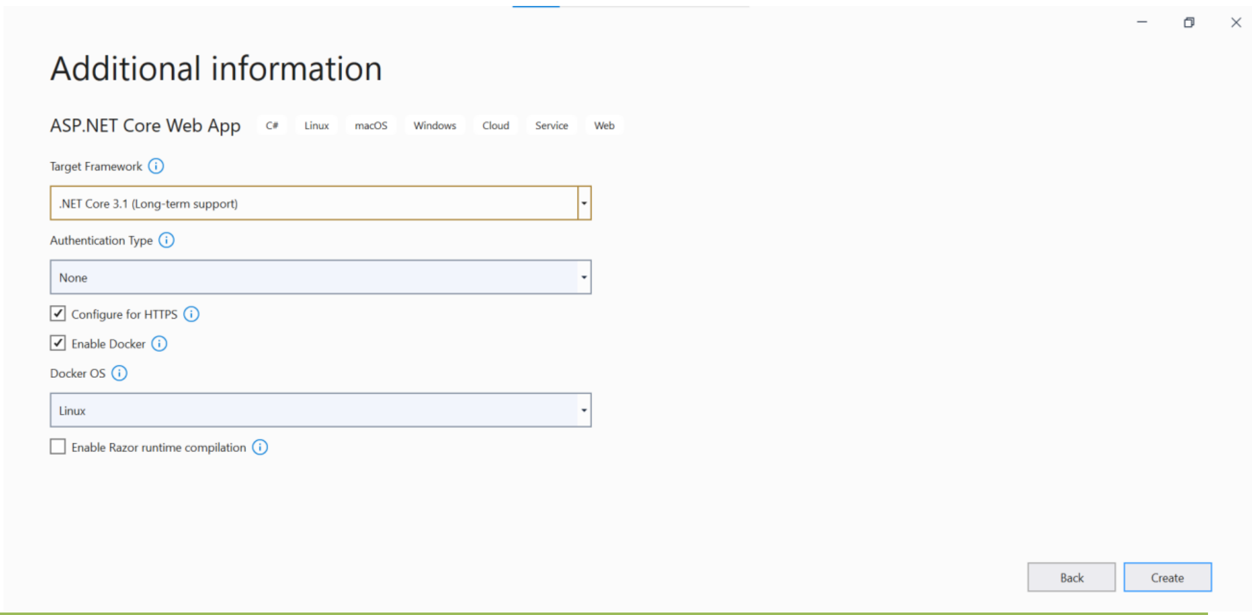
Location
C:\Source\Docker\

Solution name
DotnetCoreAppwithDocker

☑ Place solution and project in the same directory

Back Next

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



Additional information

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Target Framework
.NET Core 3.1 (Long-term support)

Authentication Type
None

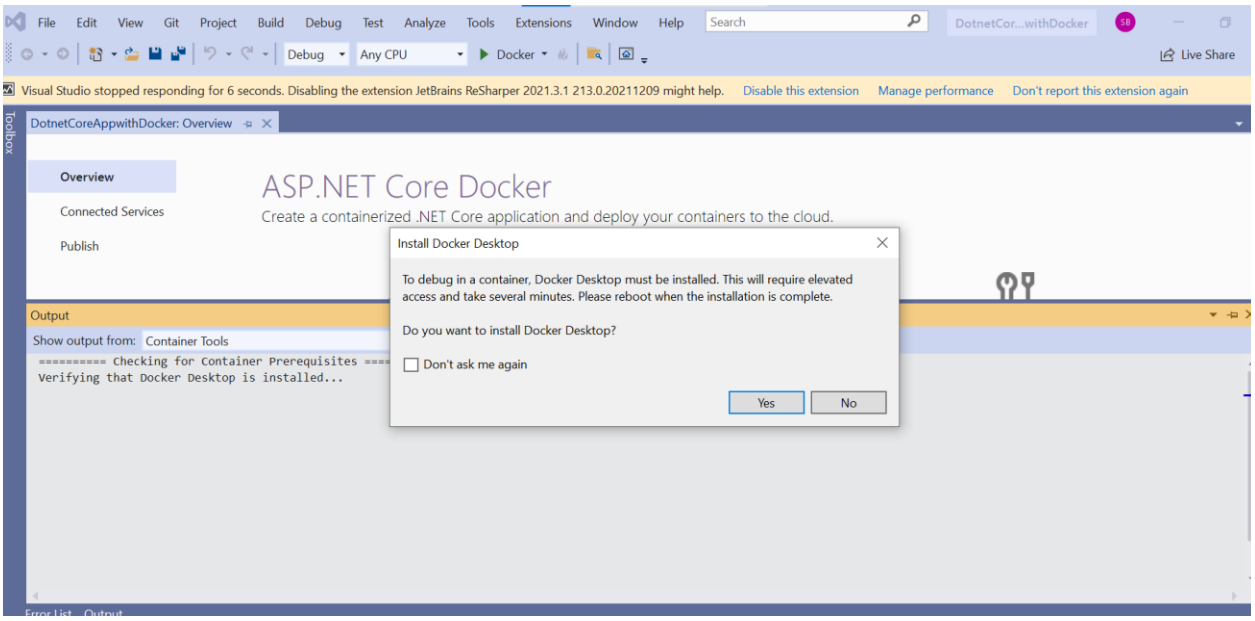
☑ Configure for HTTPS
☑ Enable Docker

Docker OS
Linux

☐ Enable Razor runtime compilation

Back 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



Visual Studio stopped responding for 6 seconds. Disabling the extension JetBrains ReSharper 2021.3.1 213.0.20211209 might help. Disable this extension Manage performance Don't report this extension again

ASP.NET Core Docker

Create a containerized .NET Core application and deploy your containers to the cloud.

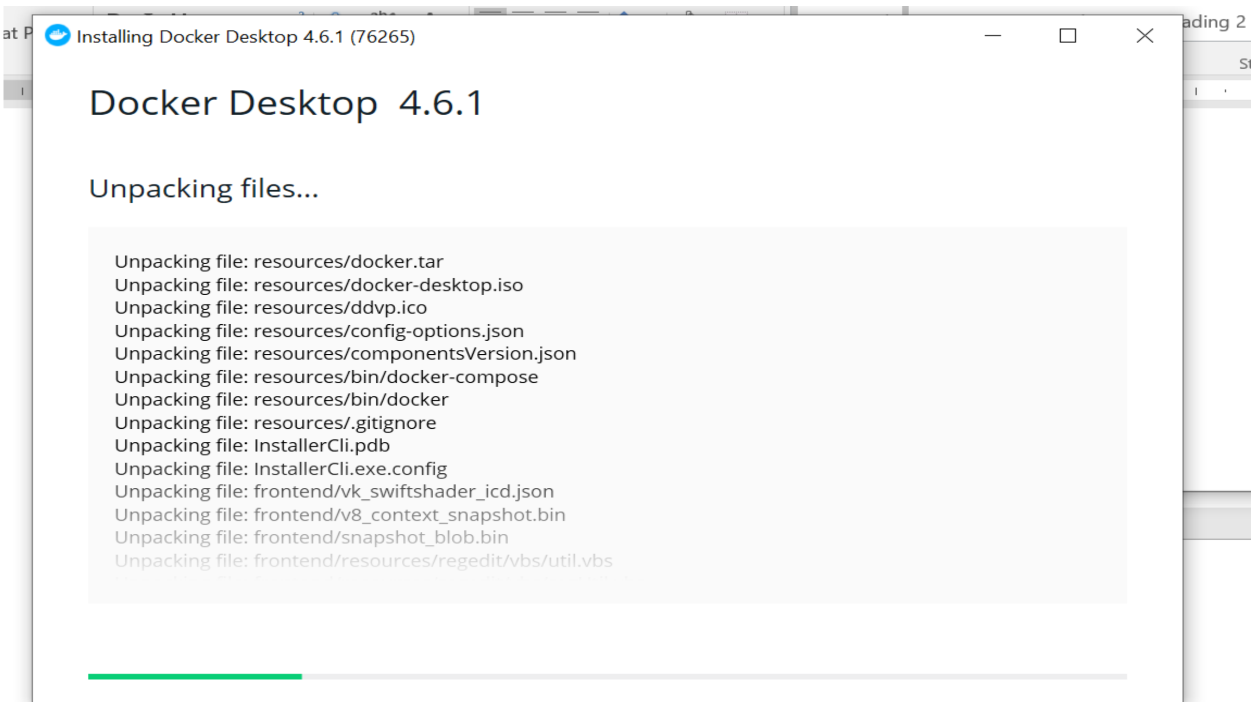
```
========== Checking for Container Prerequisites ====
Verifying that Docker Desktop is installed...
```

Install Docker Desktop

To debug in a container, Docker Desktop must be installed. This will require elevated access and take several minutes. Please reboot when the installation is complete.

Do you want to install Docker Desktop?

☐ Don't ask me again

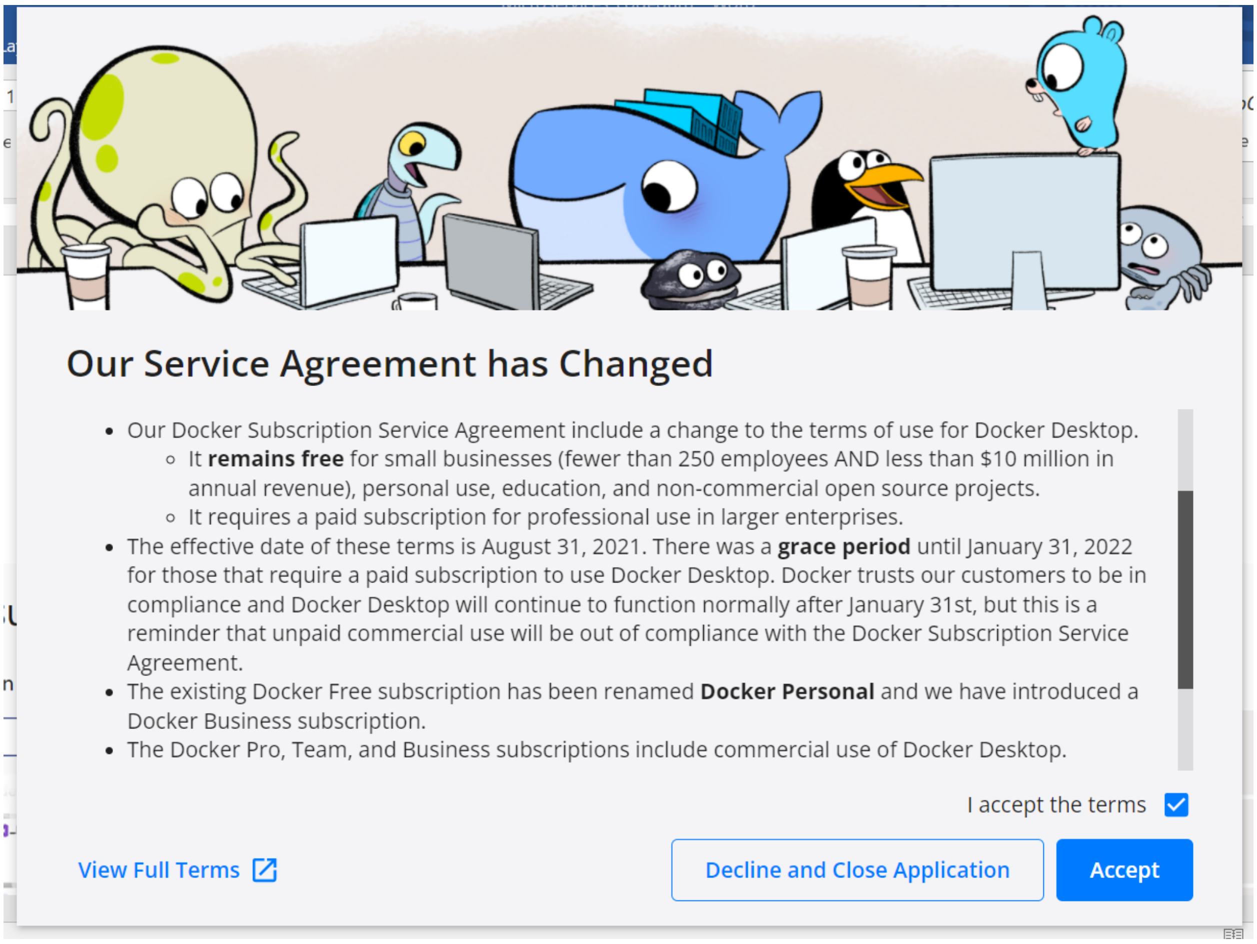


Installing Docker Desktop 4.6.1 (76265)

Docker Desktop 4.6.1

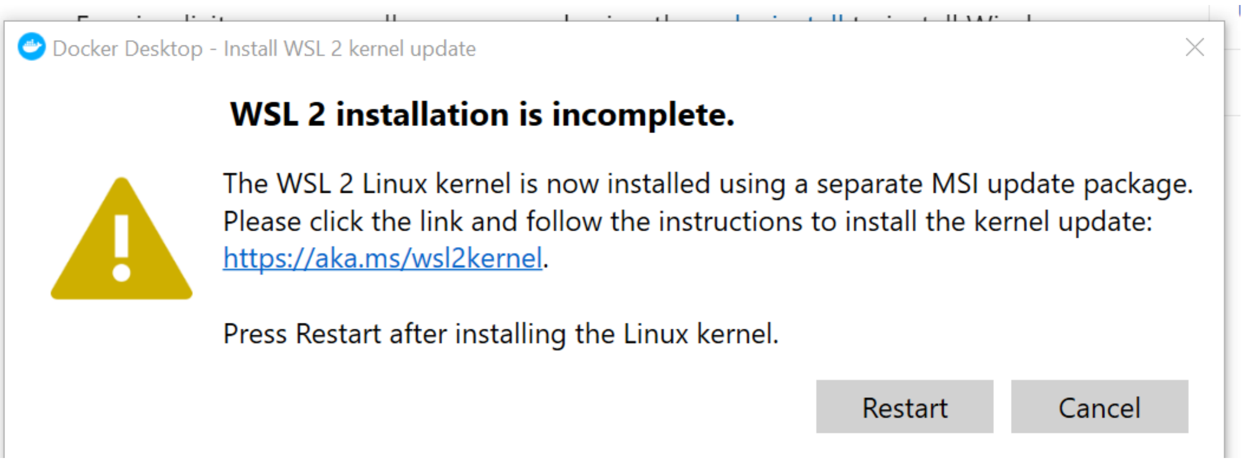
Unpacking files...

```
Unpacking file: resources/docker.tar
Unpacking file: resources/docker-desktop.iso
Unpacking file: resources/ddvp.ico
Unpacking file: resources/config-options.json
Unpacking file: resources/componentsVersion.json
Unpacking file: resources/bin/docker-compose
Unpacking file: resources/bin/docker
Unpacking file: resources/.gitignore
Unpacking file: InstallerCli.pdb
Unpacking file: InstallerCli.exe.config
Unpacking file: frontend/vk_swiftshader_icd.json
Unpacking file: frontend/v8_context_snapshot.bin
Unpacking file: frontend/snapshot_blob.bin
Unpacking file: frontend/resources/regedit/vbs/util.vbs
```


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.

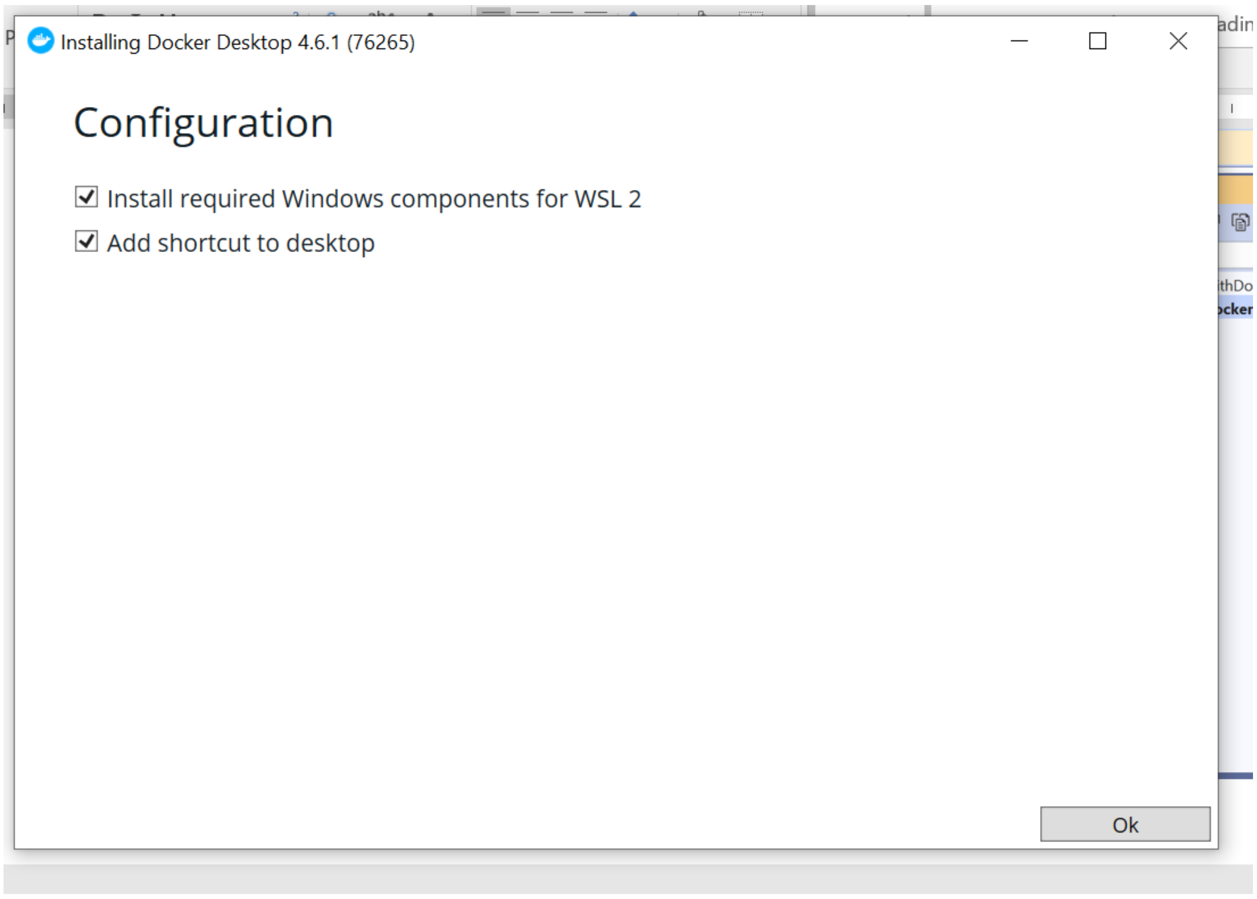
I accept the terms ✓


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.

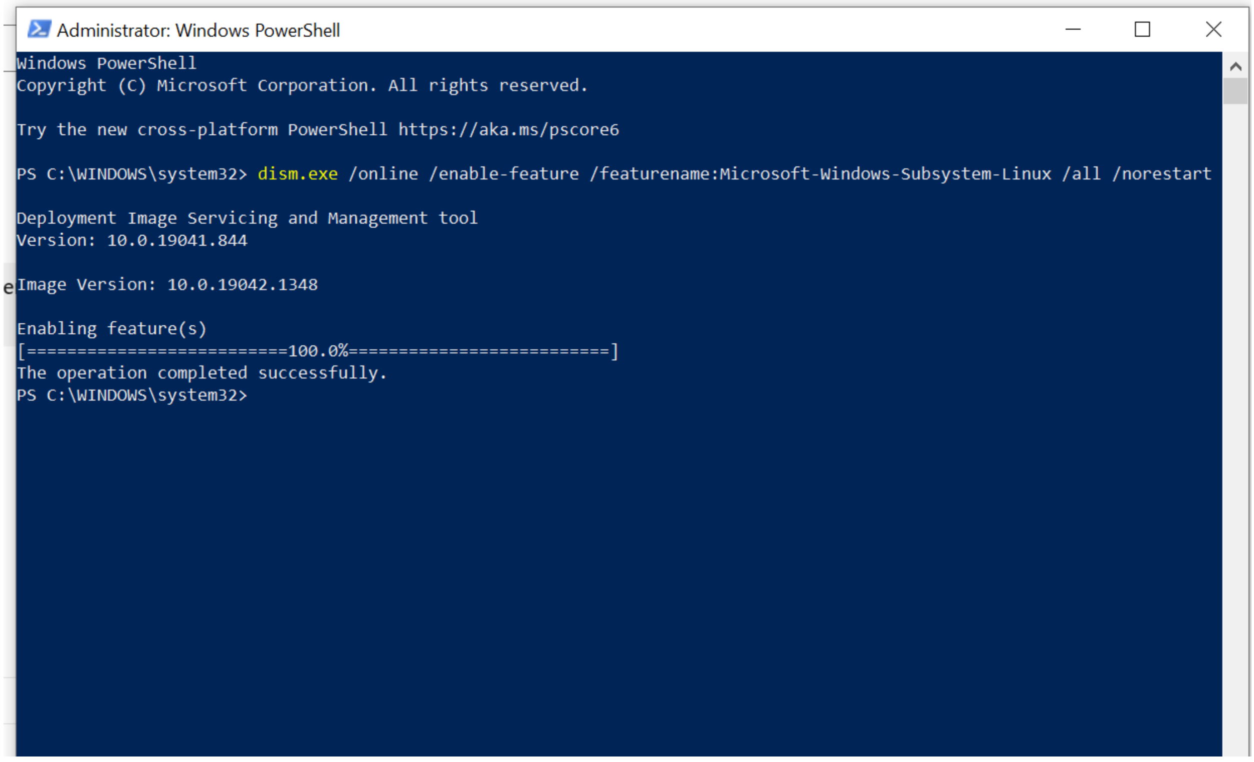
Press Restart after installing the Linux kernel.



Installing Docker Desktop 4.6.1 (76265)

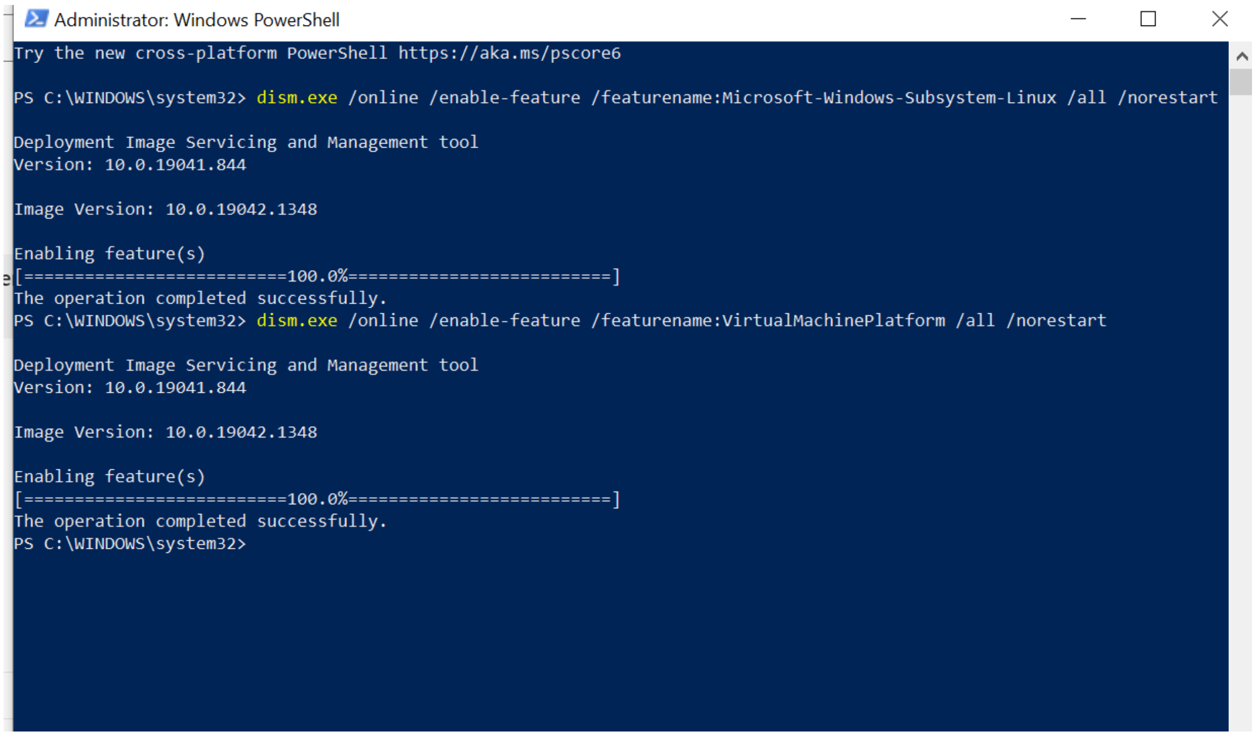
Configuration

- [x] Install required Windows components for WSL 2
- [x] Add shortcut to desktop

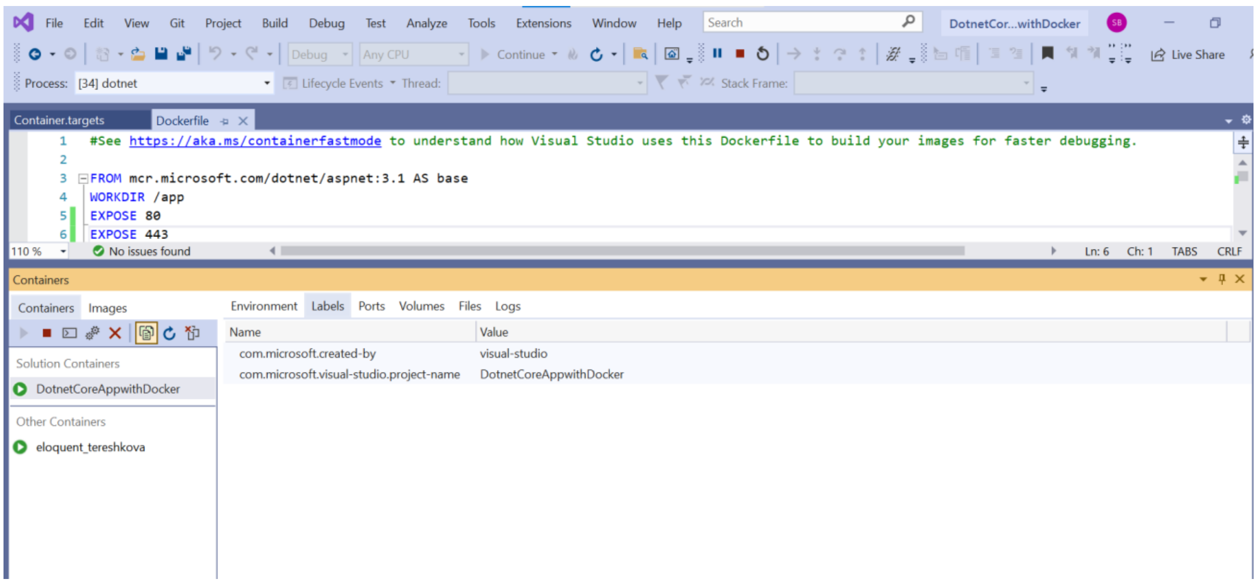


Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

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>
```



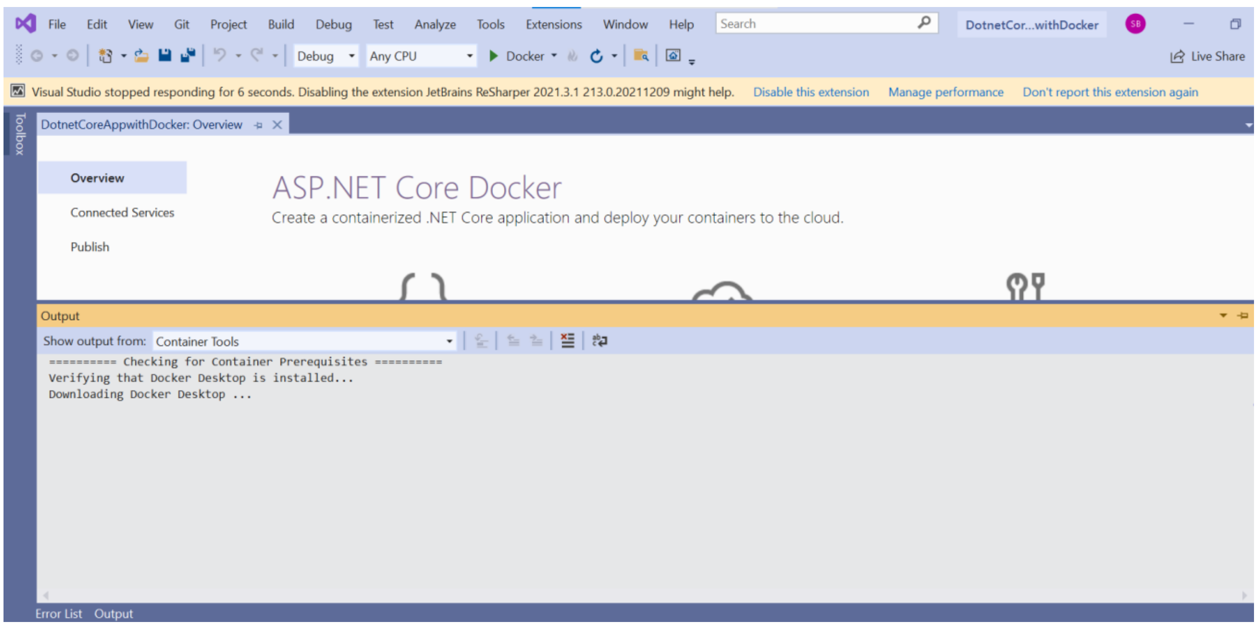
```
Try the new cross-platform PowerShell https://aka.ms/pscore6

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>
```



```
#See https://aka.ms/containerfastmode to understand how Visual Studio uses this Dockerfile to build your images for faster debugging.

FROM mcr.microsoft.com/dotnet/aspnet:3.1 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443
```

| Name | Value |
|---|---|
| com.microsoft.created-by | visual-studio |
| com.microsoft.visual-studio.project-name | DotnetCoreAppwithDocker |

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


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
DotnetCor...withDocker
Debug
Any CPU
Docker
Live Share
Toolbox
DotnetCoreAppwithDocker: Overview
Overview
Connected Services
Publish
Show output from: Container Tools
Error List Output

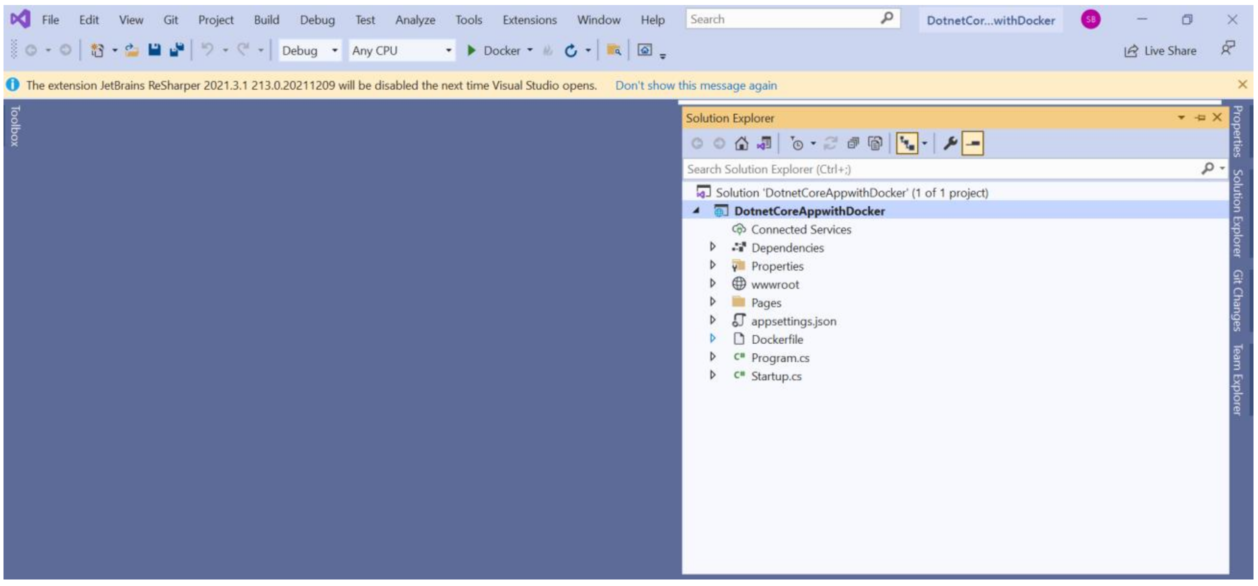
Visual Studio stopped responding for 6 seconds. Disabling the extension JetBrains ReSharper 2021.3.1 213.0.20211209 might help. Disable this extension Manage performance Don't report this extension again

ASP.NET Core Docker

Create a containerized .NET Core application and deploy your containers to the cloud.

Output

```
========== Checking for Container Prerequisites ==========
Verifying that Docker Desktop is installed...
Downloading Docker Desktop ...
```

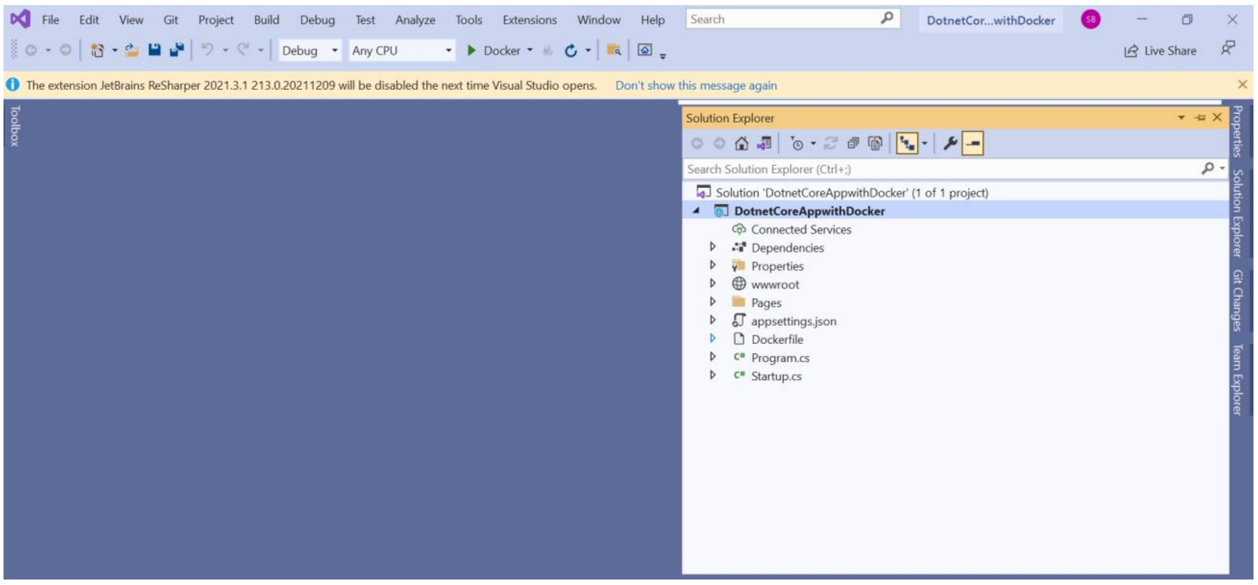


The extension JetBrains ReSharper 2021.3.1 213.0.20211209 will be disabled the next time Visual Studio opens. Don't show this message again

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'DotnetCoreAppwithDocker' (1 of 1 project)
- DotnetCoreAppwithDocker
  - Connected Services
  - Dependencies
  - Properties
  - wwwroot
  - Pages
  - appsettings.json
  - Dockerfile
  - Program.cs
  - Startup.cs



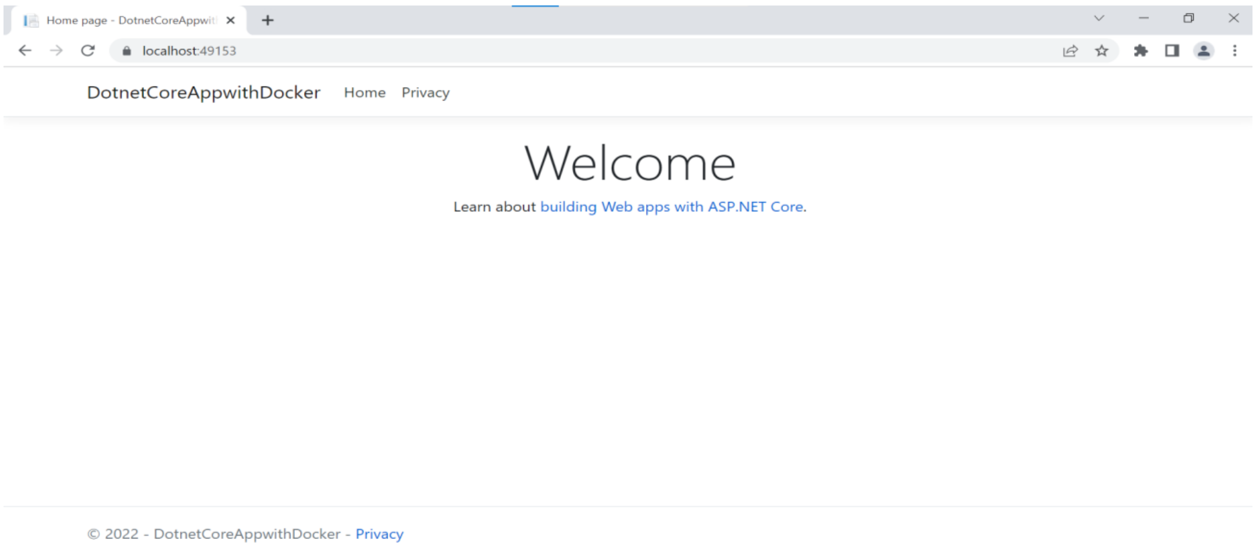
The extension JetBrains ReSharper 2021.3.1 213.0.20211209 will be disabled the next time Visual Studio opens. Don't show this message again

Solution Explorer

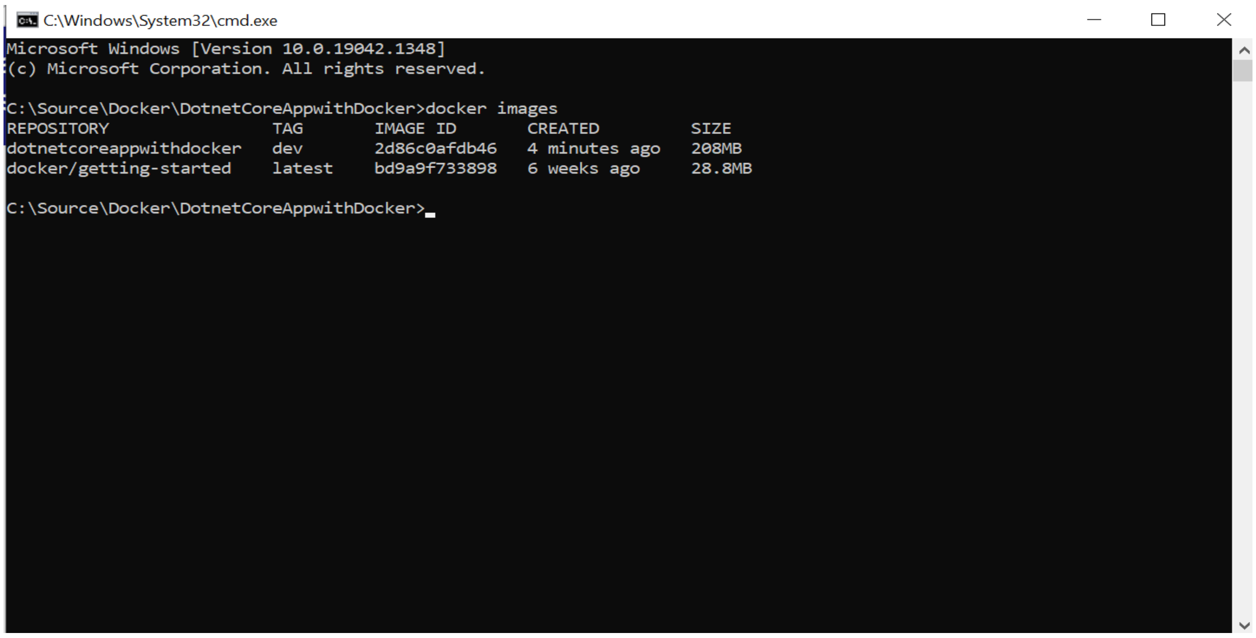
Search Solution Explorer (Ctrl+;)

Solution 'DotnetCoreAppwithDocker' (1 of 1 project)
- DotnetCoreAppwithDocker
  - Connected Services
  - Dependencies
  - Properties
  - wwwroot
  - Pages
  - appsettings.json
  - Dockerfile
  - Program.cs
  - Startup.cs

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


Home page - DotnetCoreAppwit
localhost:49153
DotnetCoreAppwithDocker
Home
Privacy
Welcome
Learn about building Web apps with ASP.NET Core.
© 2022 - DotnetCoreAppwithDocker - Privacy

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



C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Source\Docker\DotnetCoreAppwithDocker>docker images
REPOSITORY                 TAG       IMAGE ID       CREATED          SIZE
dotnetcoreappwithdocker    dev       2d86c0afdb46   4 minutes ago    208MB
docker/getting-started     latest    bd9a9f733898   6 weeks ago      28.8MB

C:\Source\Docker\DotnetCoreAppwithDocker>
```

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