Chapter 9 Sample questions and answers

1. How to publish asp.net core web app in azure cloud?

   Step 1: Right click in project and select publish option.

   Step 2: Select **Azure** as **target**

   Step 3: Click Next and Select **specific target** and you will find **Azure App Services(Windows)** in right pane

   Step 4: After clicking Next you will find **App Service** in left pane and click (+) sign to create **new App Service instances** in right pane

   Step 5: After clicking (+) sign you will see new popup window to add **new Service Instance** which has following field like Name, Subscription, Resource group, Hosting Plan. Now just click in new hosting plan from (+) sign button in hosting plan section. In Hosting Plan section, you will have option to select **Location** and **Size.** Now you will select size option to **Free** and click **Ok**. And then click **Create.** Now app service instance is created in Azure.

   Step 6: Now you will see App Service Instance is created and select newly created instance and click Next.

   Step 7: You will redirect to API Management tab in right pane, this option for API management you can skip by selecting **checkbox** skip this step and click **Finish**.

   Follow youtube link: https://www.youtube.com/watch?v=MP4zatl3jF8

2. How to host asp.net core web app in IIS?

   Step 1: Make sure IIS is installed in your OS.

   Step 2: If not installed go to **Control Panel** from control panel you can find **Add and remove features** in right pane of window and click it you will find option of **Internet Information Service** with check box. Check features you required and install IIS.

   Step 3: Now open project in Visual Studio 2019 and right click in project and select **Publish.**

   Step 4: Now **publish** window appears and select **target** tab in right pane of window. After selecting it shows different option where to host, Now you can select **Folder** option to host and click Next.

   Step 5. After clicking next, it will have **location** tab in right pane, and browse button which will tells where you want to publish. Select folder and click Finish.

   Step 6: Click Publish, and you will find all published files inside folder.

   Step 7: Paste publish folder inside (C:\inetpub\wwwroot) Location if IIS installed properly.

   Step 8: To run asp.net core also install **dot net core windows hosting bundle** to work properly.

   Otherwise you will get 500.19 internal error.

   Follow url: https://www.youtube.com/watch?v=Q_A_t7KS5Ss

3. What is ASP.NET Core Module (ANCM)?

The ASP.NET Core Module is a native IIS module that plugs into the IIS pipeline, allowing ASP.NET Core applications to work with IIS. Run ASP.NET Core apps with IIS by either:

- Hosting an ASP.NET Core app inside of the IIS worker process (w3wp.exe), called the in-process hosting model.
- Forwarding web requests to a backend ASP.NET Core app running the Kestrel server, called the out-of-process hosting model.

There are trade-offs between each of the hosting models. By default, the in-process hosting model is used due to better performance and diagnostics.

4. How to use Kestrel in ASP.NET Core apps?

The **Microsoft.AspNetCore.Server.Kestrel** package is included in the **Microsoft.AspNetCore.App metapackage**.

ASP.NET Core project templates use Kestrel by default. In *Program.cs*, the template code calls CreateDefaultBuilder, which calls UseKestrel behind the scenes.

```
public static void Main(string[] args)
{
   CreateWebHostBuilder(args).Build().Run();
}
public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
   WebHost.CreateDefaultBuilder(args)
      .UseStartup<Startup>();
```

For more information on CreateDefaultBuilder and building the host, see the *Set up a host* section of ASP.NET Core Web Host.

To provide additional configuration after calling CreateDefaultBuilder, use ConfigureKestrel:

```
public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
   WebHost.CreateDefaultBuilder(args)
      .UseStartup<Startup>()
      .ConfigureKestrel((context, serverOptions) =>
      {
         // Set properties and call methods on serverOptions
      });
```

If the app doesn't call CreateDefaultBuilder to set up the host, call UseKestrel **before** calling ConfigureKestrel:

```
public static void Main(string[] args)

{
   var host = new WebHostBuilder()
      .UseContentRoot(Directory.GetCurrentDirectory())
      .UseKestrel()
      .UseIISIntegration()
      .UseStartup<Startup>()
      .ConfigureKestrel((context, serverOptions) =>
      {
         // Set properties and call methods on serverOptions
      })
      .Build();
   host.Run();

}
```

5. **What is Docker and Containerization? What is the difference between Virtual Machine and Docker Container? How to host asp.net core application in docker?**

Docker is the platform for deploying and building the applications which delivers the application in the packages over the operating system level virtualization. Docker requires the following components to run the application.

- Image
- Container

What is the docker image?

Docker image is the set of configuration and instructions to create the docker container, an image is created during building the application based on steps defined in the docker file of your application. The docker image is the read-only file which cannot be modified once it's created, but we can delete the image from docker.

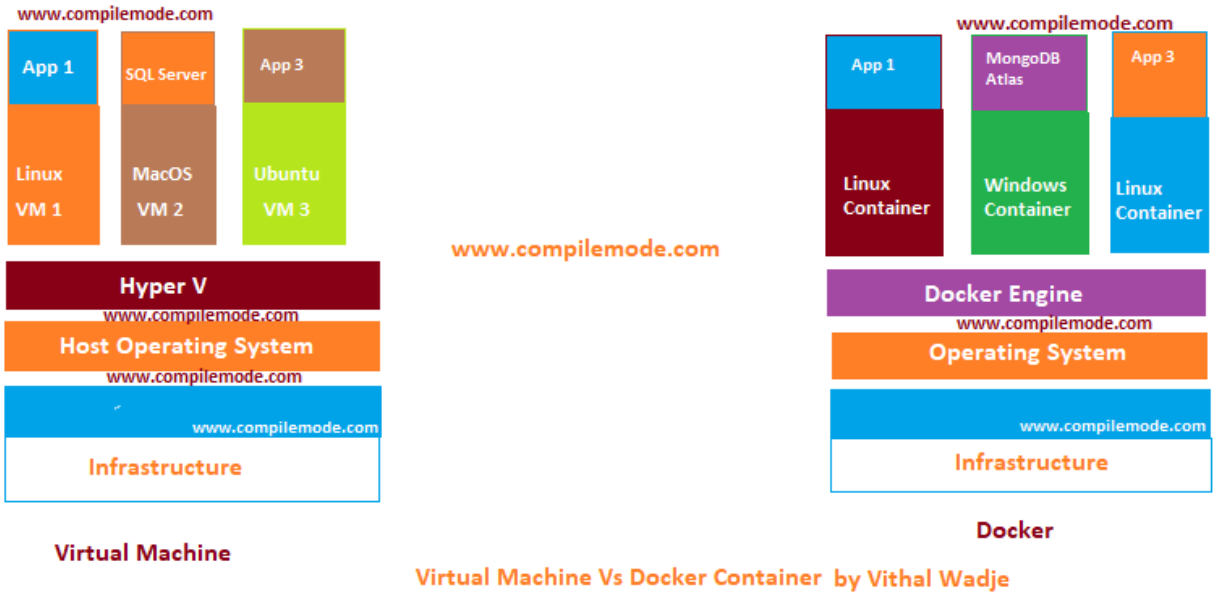
What is a Container?

The Container is an independent isolated process of an operating system that has its own networking and file system to run the image or application. The container is created on the docker engine based on the image configuration.

Difference between Virtual Machine and Docker Container?

The virtual machine (Linux, ubuntu) is installed on some other operating system (windows) which shares the same lifecycle such as running and shutting at the same time. Virtual machine acts as a real PC on an actual operating system which has full features such as RAM, Hard Disk, networking etc. In this case, often virtual machines are called guest PC and the actual PC which runs the VM is called host system machine.

The container is the minimal and smaller part of a virtual machine which does not use the entire operating system as a virtual machine. The container cuts the unnecessary components of the VM and creates the isolated virtualized environment called a container, which is faster than virtual machine. The container does not require the host system as a virtual machine instead, it works on the docker engine.



The preceding diagram gives more understanding about the virtual machine and docker container, now let's start building and running the ASP.NET Core application in docker step by step.

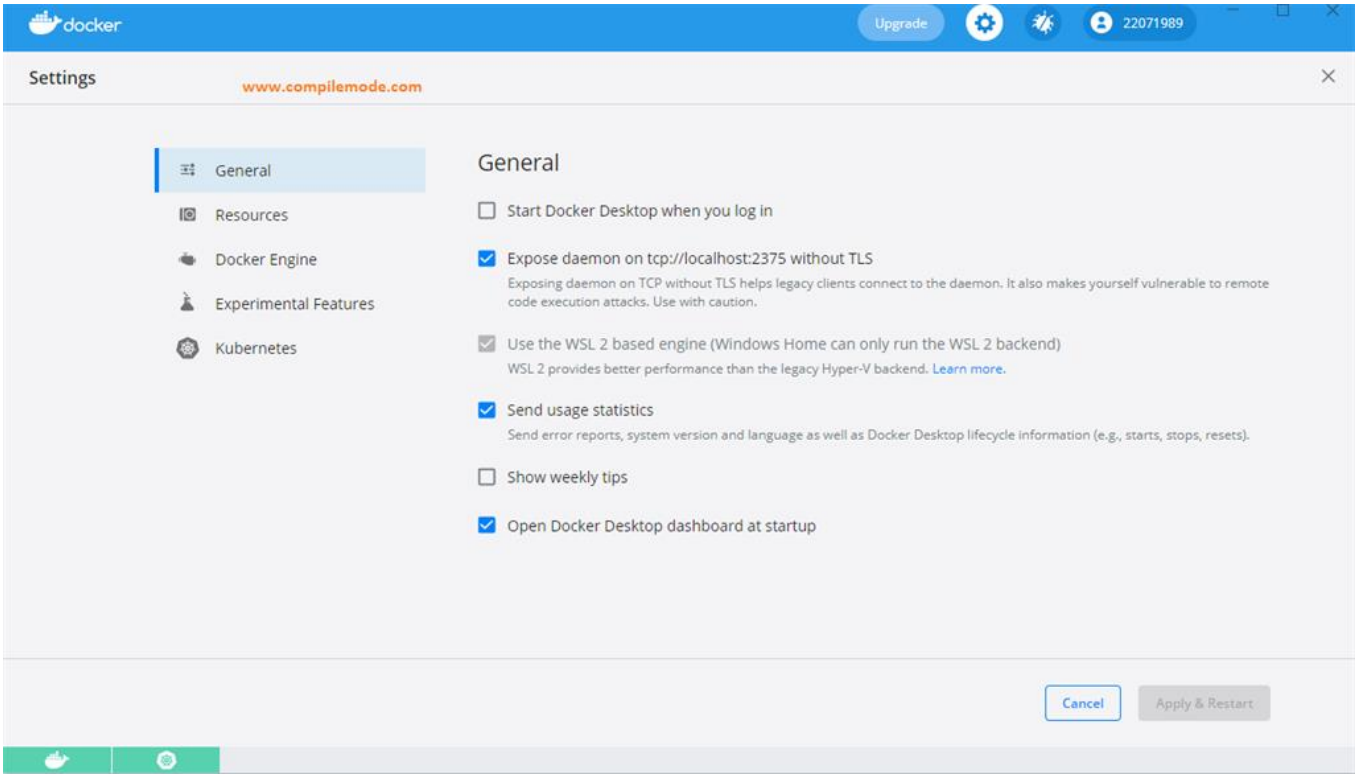
Step 1: Install Prerequisites

You need to install the docker desktop client based on your machine (PC) as per your operating system type. Use the following link to download the docker desktop client.

Url: https://www.docker.com/products/docker-desktop
Install WSL on Windows 10 | Microsoft Docs(https://docs.microsoft.com/en-us/windows/wsl/install-win10#step-4---download-the-linux-kernel-update-package)

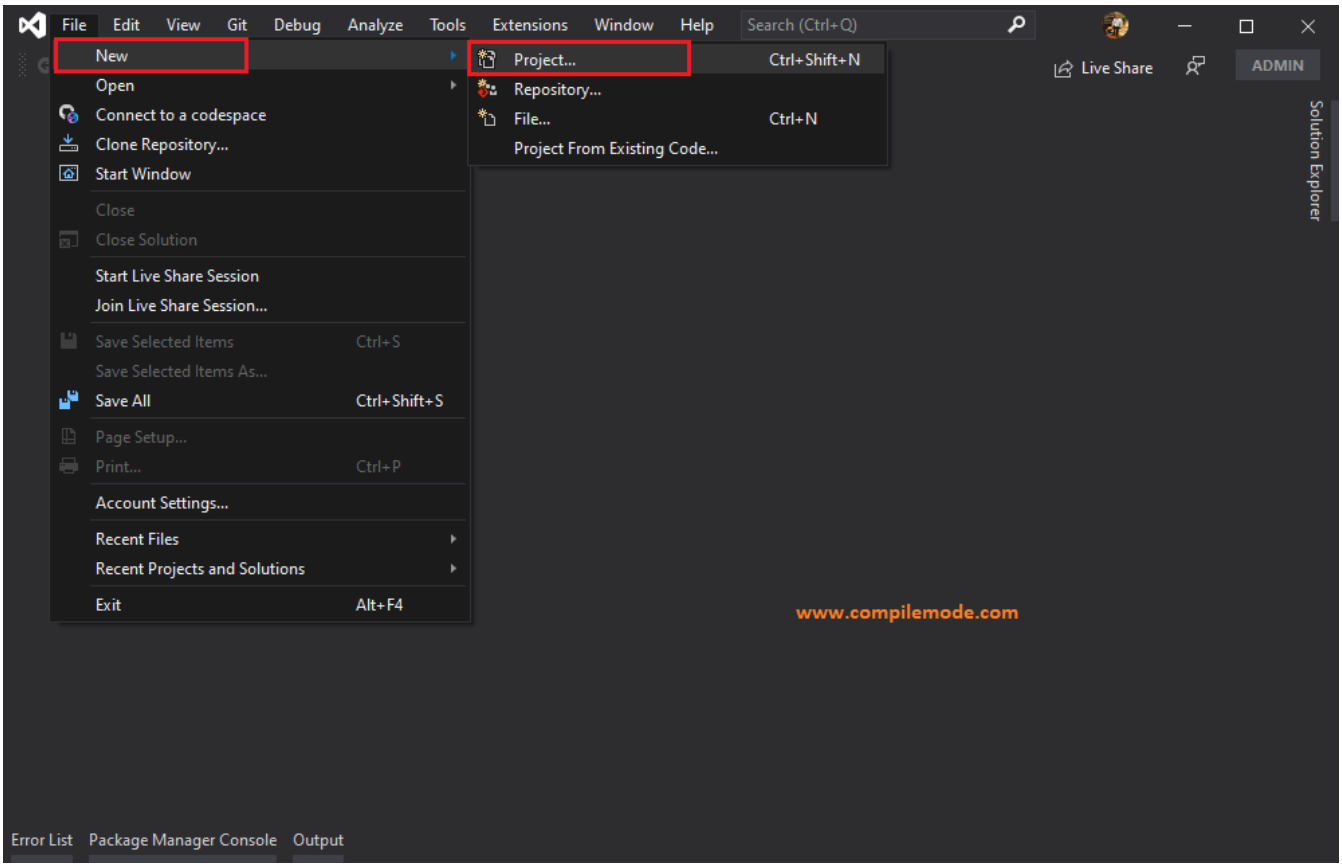
To work docker perfectly.

The installed docker desktop client looks like as follows.

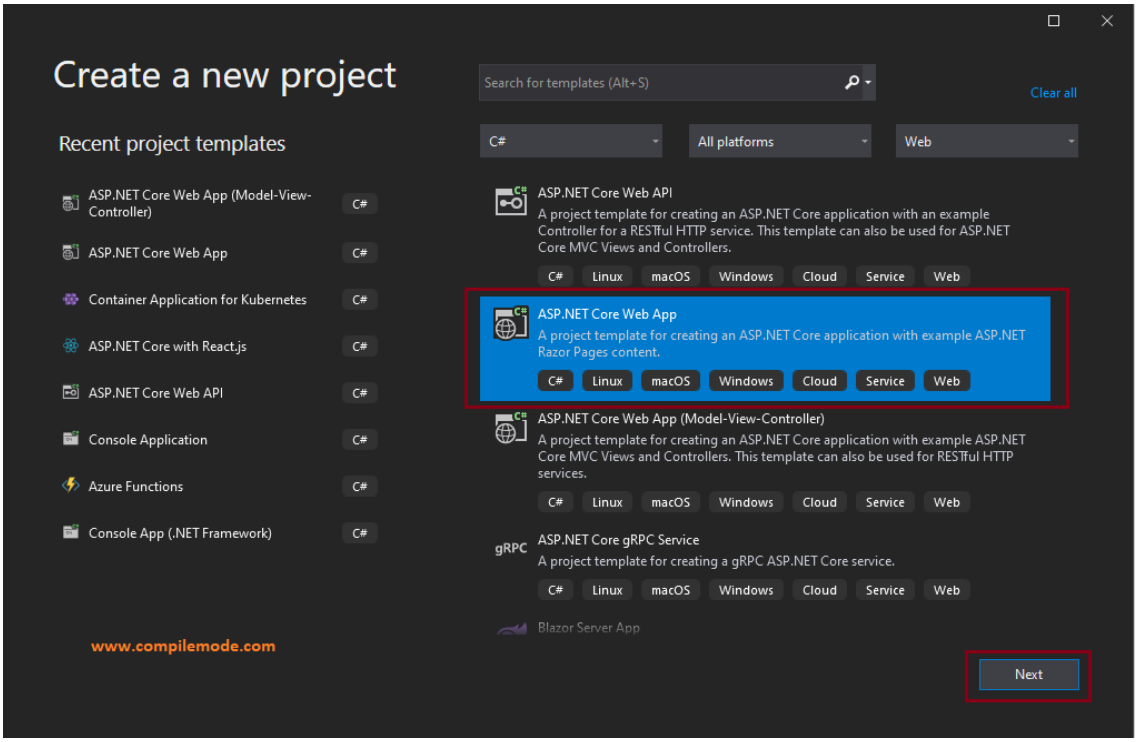


Step 2 : Create the ASP.NET Core Application

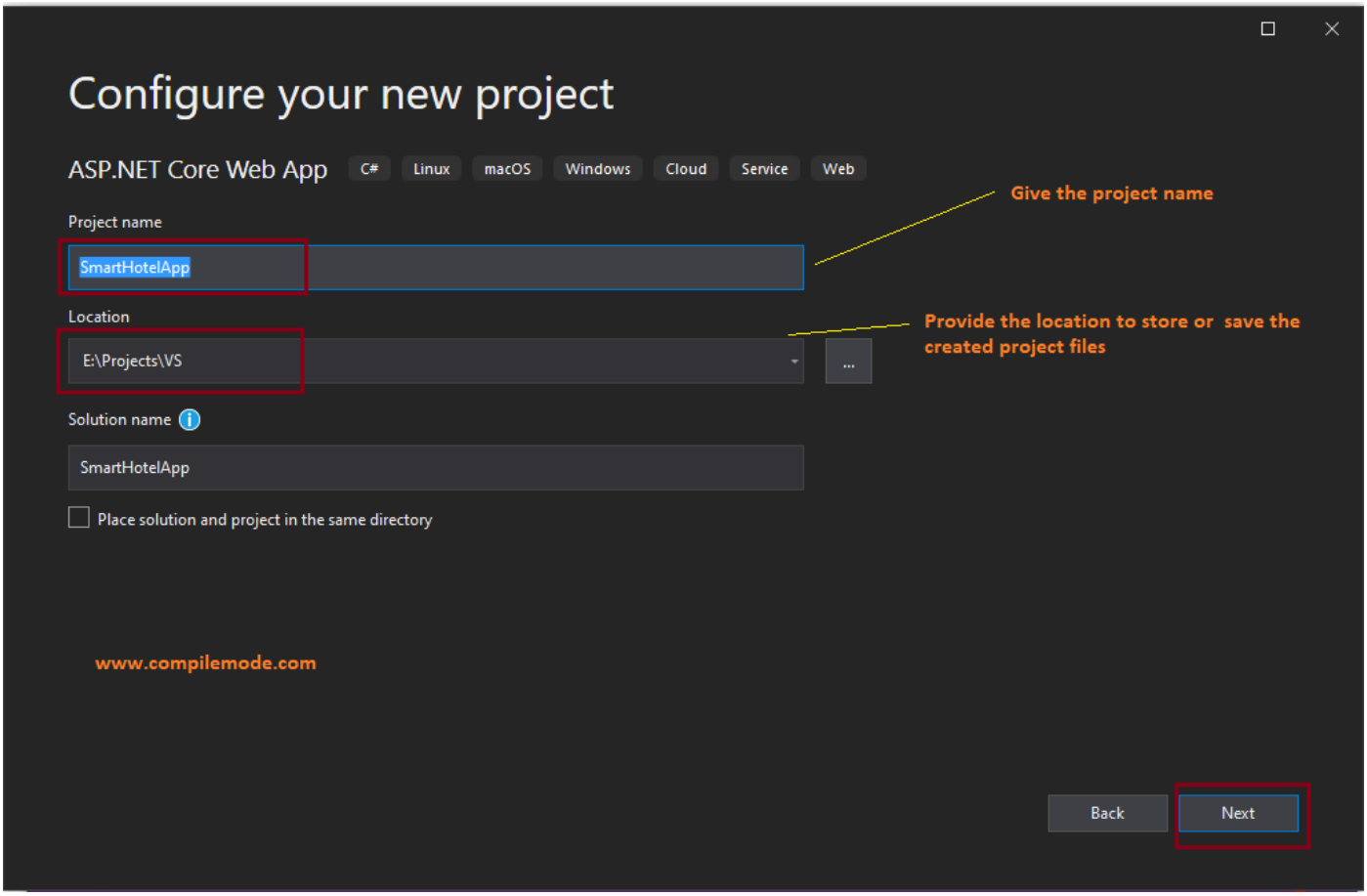
1. Open Visual Studio (I am using 2019)
2. Once Visual studio open then click on continue without code(If you are using VS 2019)
3. Then from Visual Studio Menu, click on File => New Project , as shown in the following image



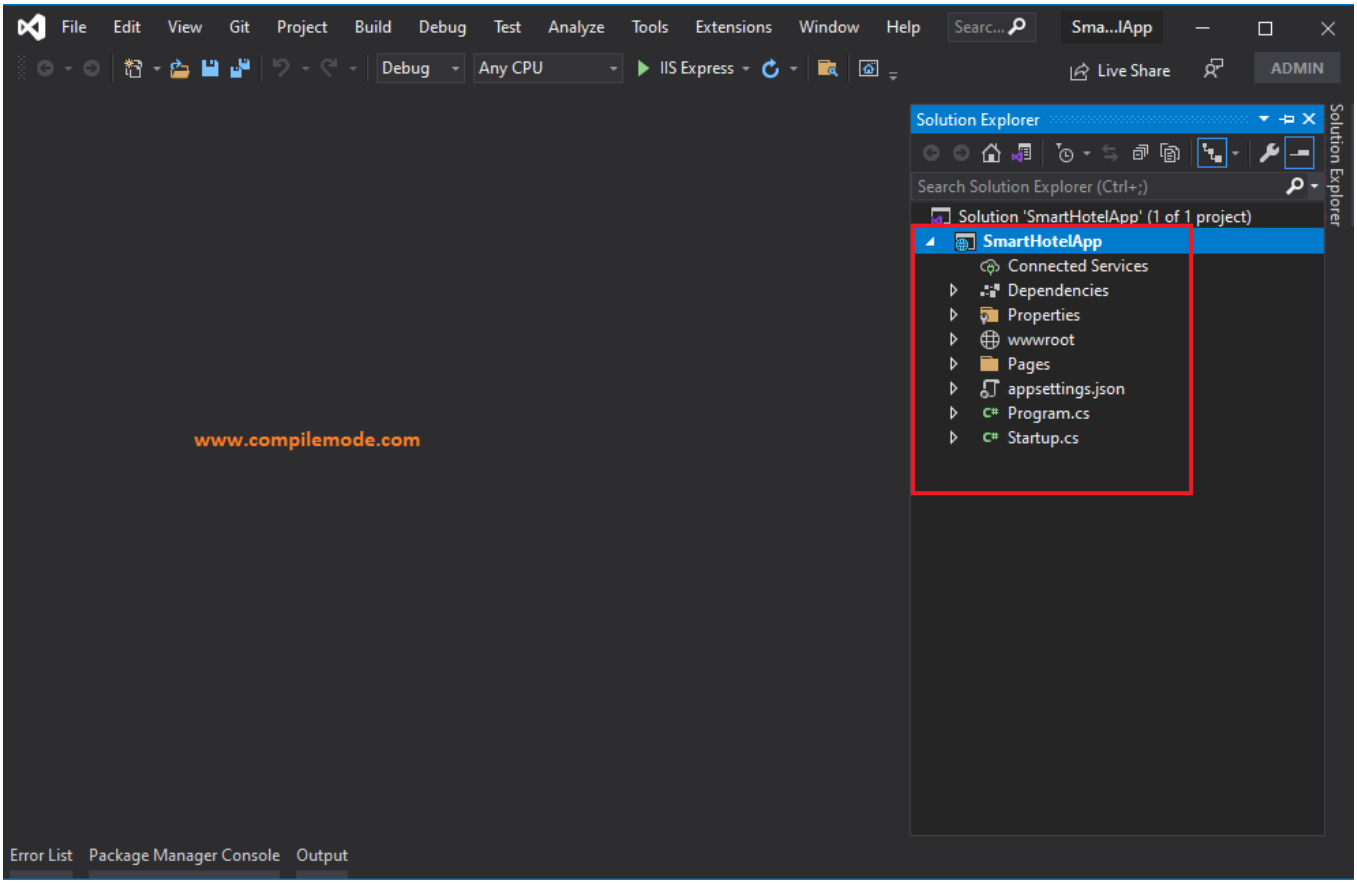
After clicking on the New Project, following window will get appears, Choose the project template ASP.NET Core Web App as shown in the following image,



After Choosing the project template click on next button, provide the project name and storage location of the project on your PC



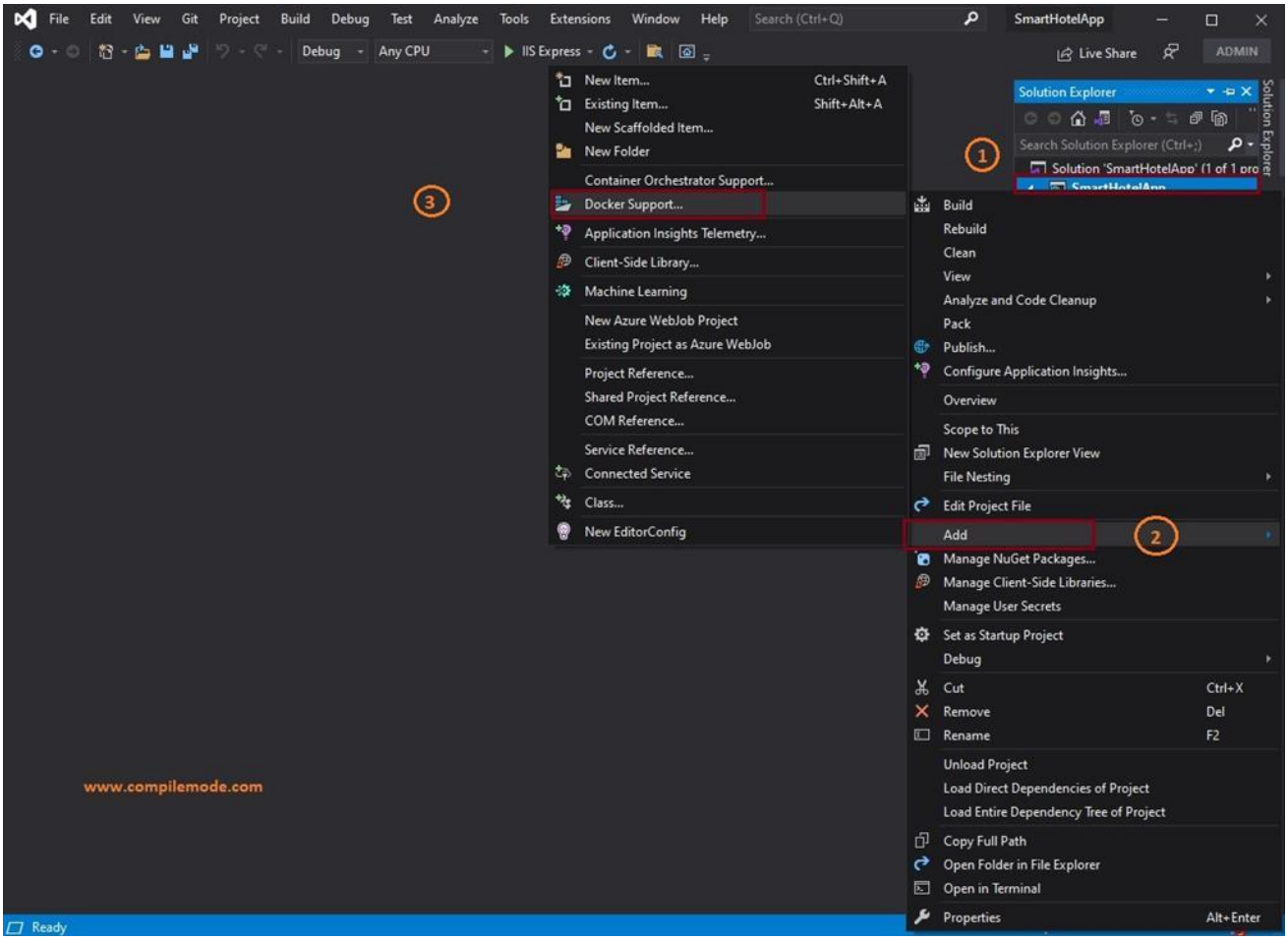
Now providing the required details, click on create button it will create the ASP.NET Core web application and created ASP.NET Core web application looks like as follows in your visual studio.



Step 3: Add Docker Support

Docker support can be enabled in two ways from visual studio, at the time of creating the application and after creating the application. To make it easier to make understand for beginners, in this tutorial we will enable the docker support after creating the ASP.NET core application.

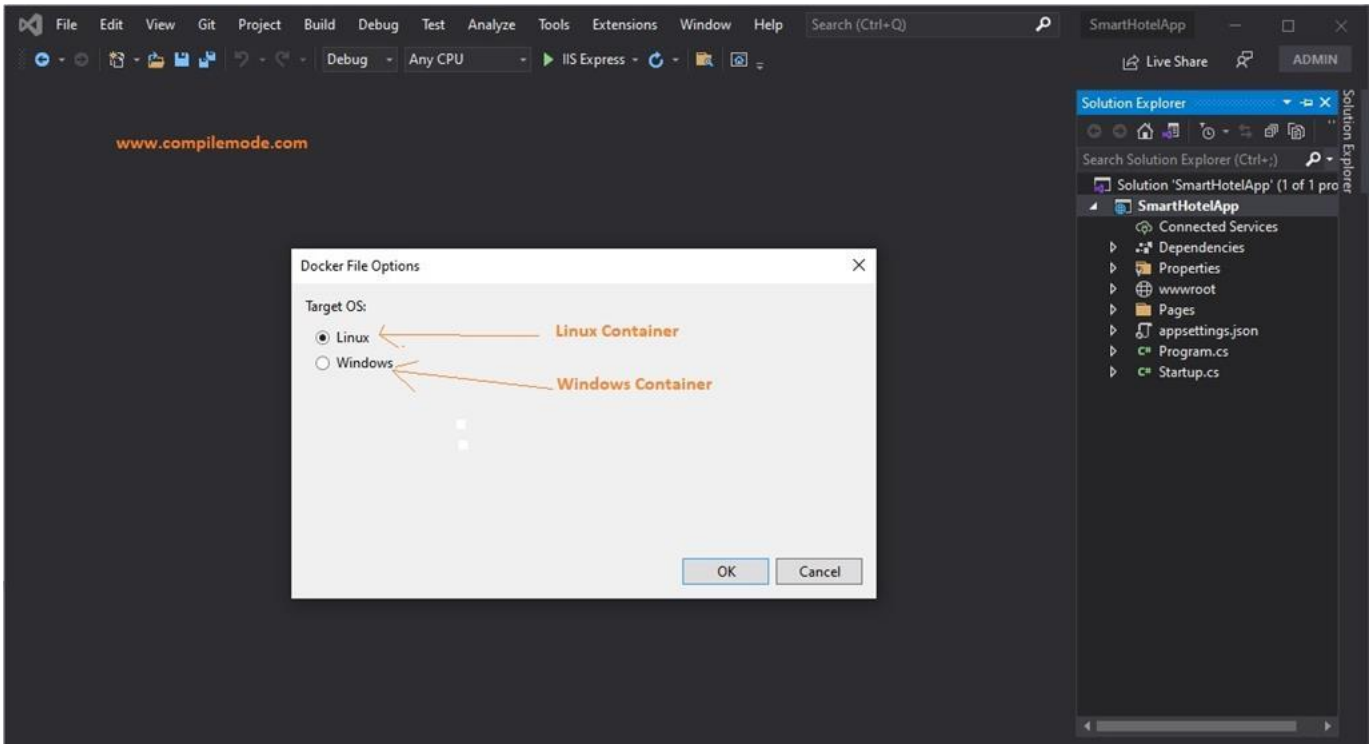
Right click on the solution explorer of your existing project and follow the steps which are shown in the following image.



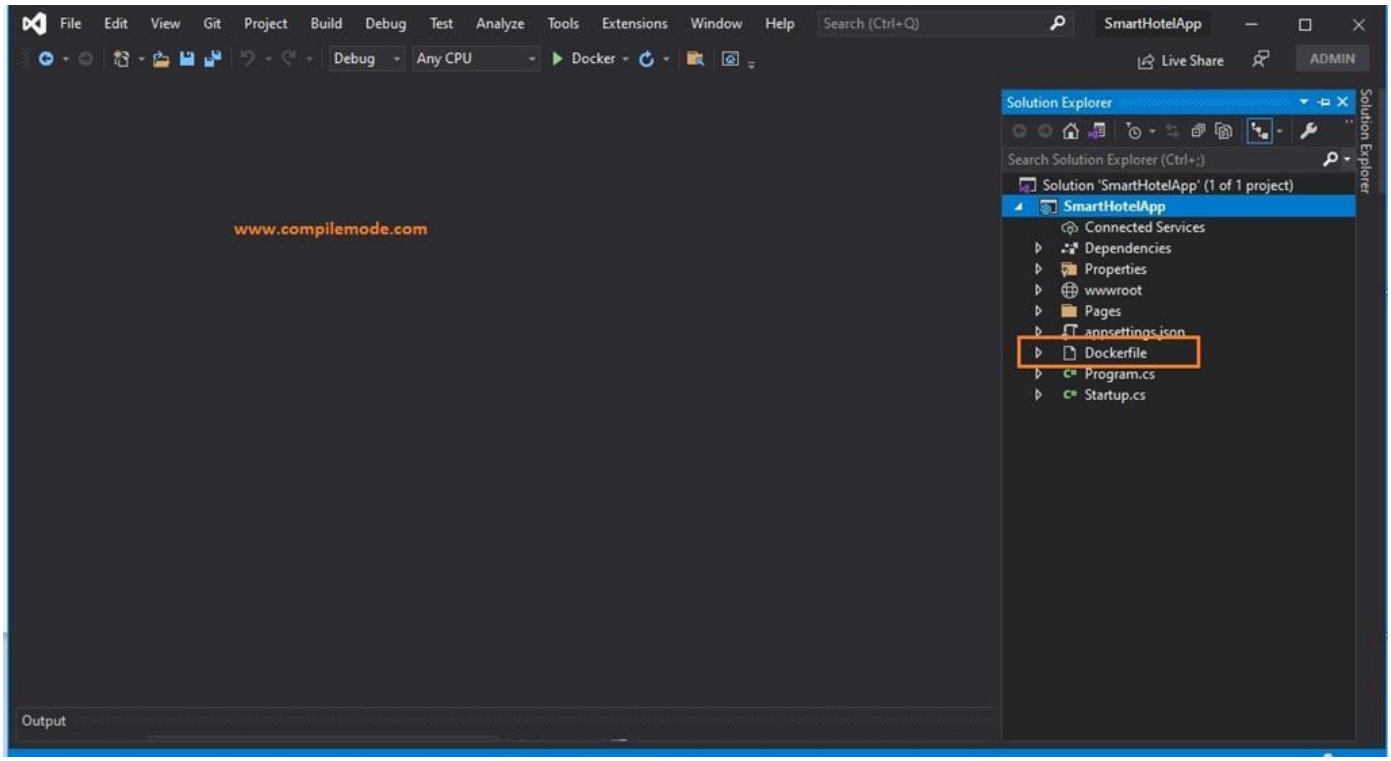
Once you click on the Docker Support option, it will prompt the screen which is shown in the step 4.

Step 4: Choose the Container (Docker File)

Choose the target operating system on which type of container you want to run the application, and it will create the docker image. Let's choose the Linux option which creates the image size smaller than compared to the windows container image.



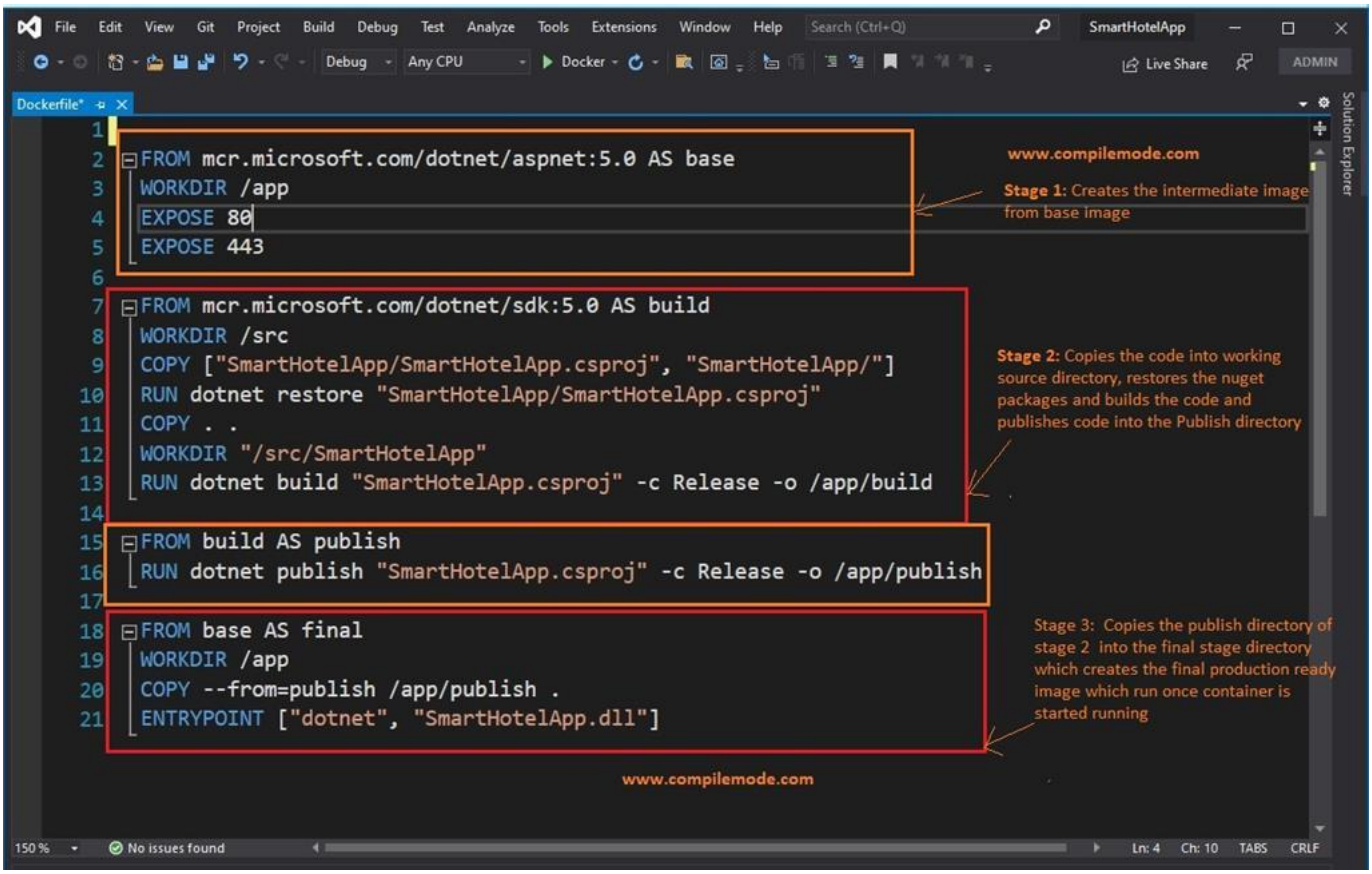
In the preceding image click on the OK button, it will create the Docker file in your project solution as shown in the following image.



Step 5: Open the Docker File

Double click on the docker file in which you will see auto generated code from visual studio which has pre-configured steps to create and run the docker image in multiple stages. The visual studio creates a docker file in your project which has multi-stages build features which make sure the final image remains smaller in size, which helps to become a container more efficient and faster. The container images were created in the following stages.

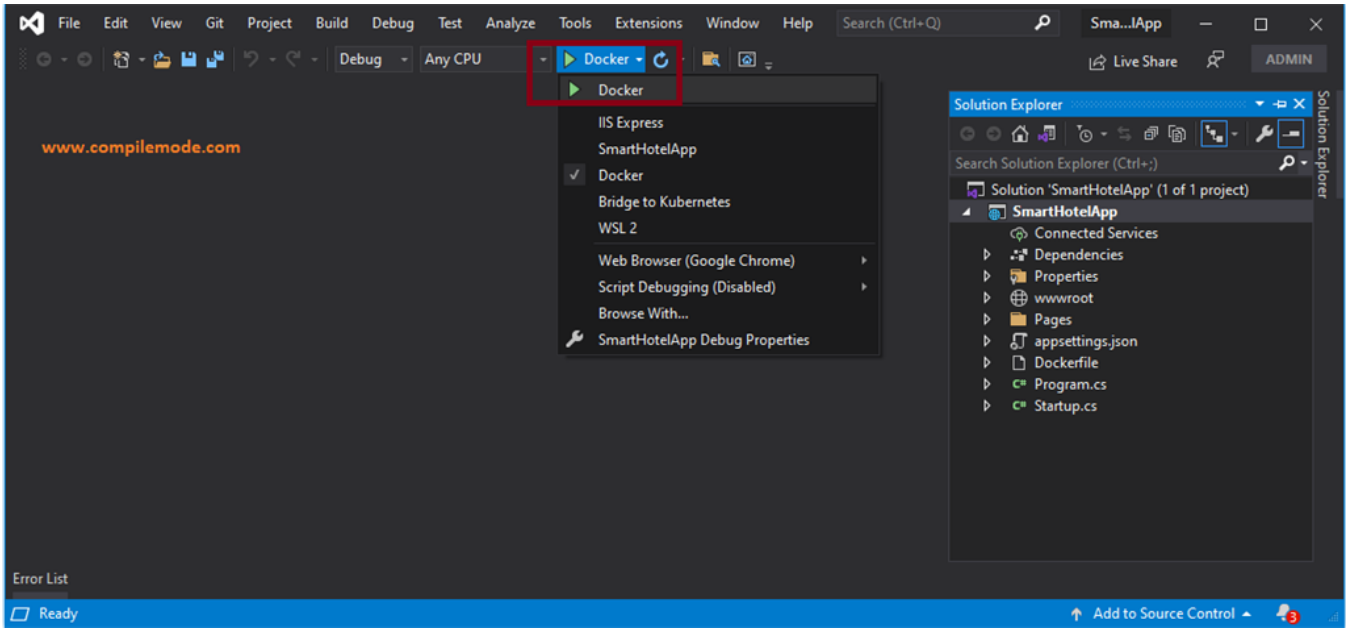
- Base
- Build
- Publish
- Final Image



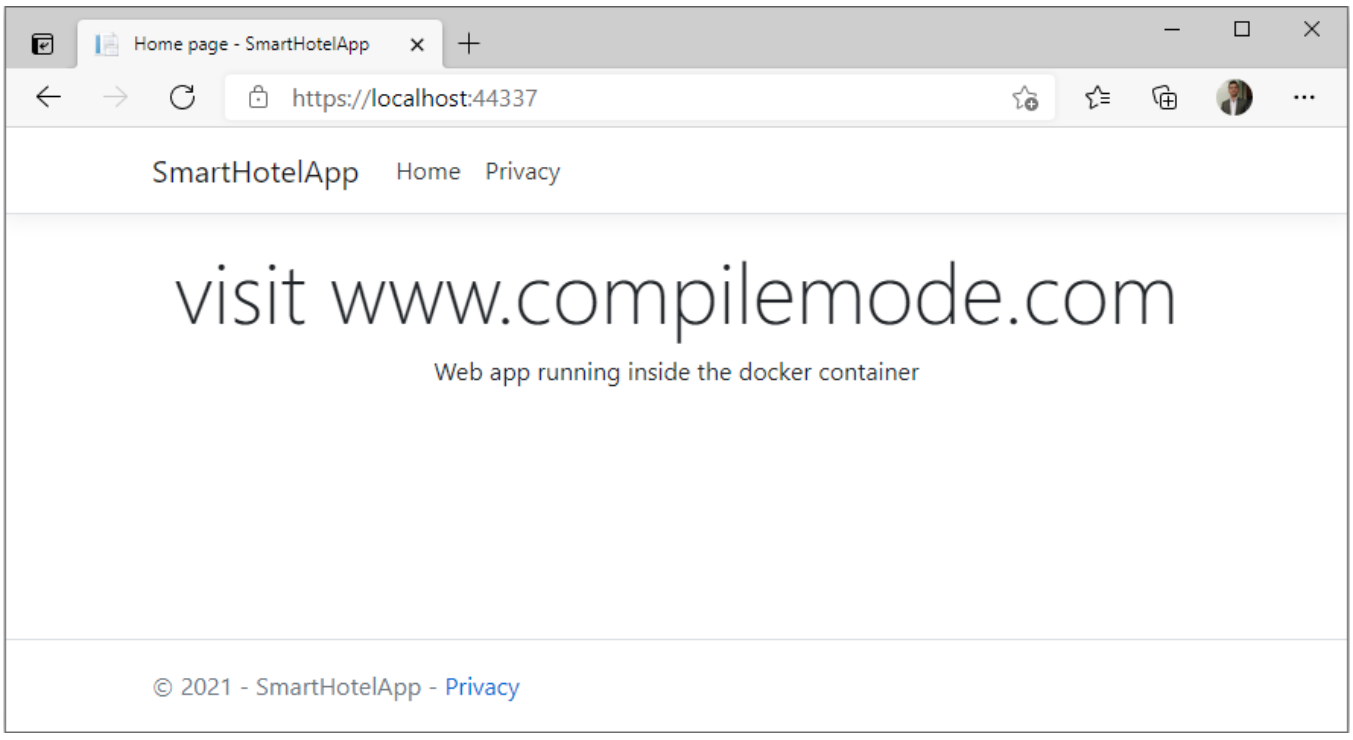
We will learn more about the base image and docker file stages in a separate article.

Step 6: Run the image in Docker Container

Choose the docker option to run the application as shown in the following image.



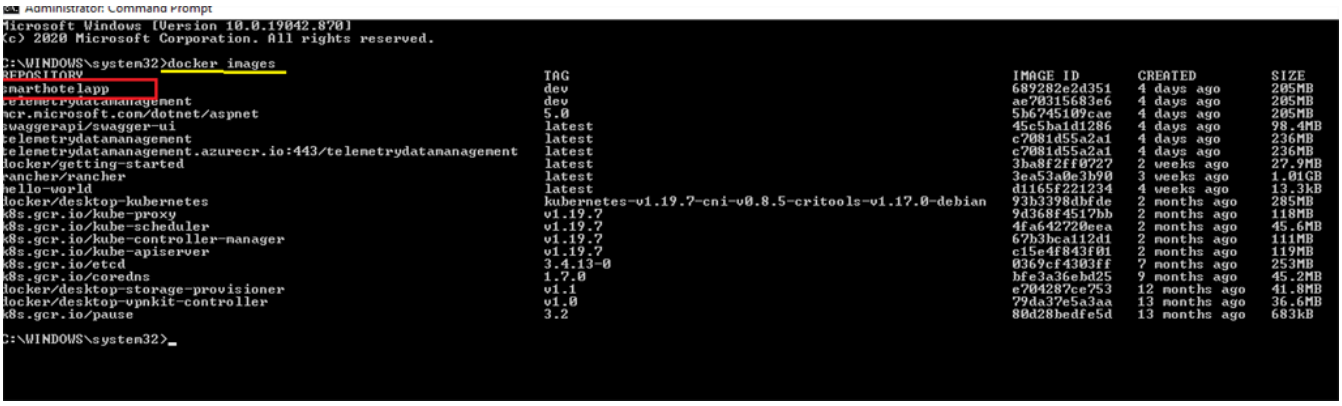
After clicking on the docker option, it will build code, create a docker image as well as a docker container and run the application inside the docker container without using the docker commands on the windows command prompt. The application opens the browser as follows.



In the preceding, our application is running inside the Linux Docker container instead of IIS express. We can view the created images by using the docker command or navigating to the docker desktop dashboard. Let's view images using the docker command. Open your windows command prompt and use the following command on the command prompt and press enter.

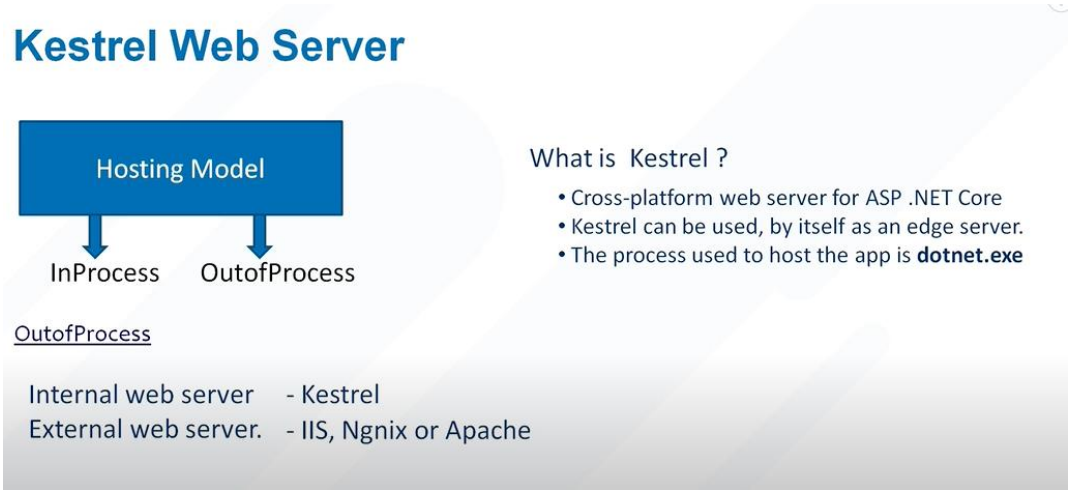
Docker Images

The preceding docker command will display the list of available docker images as follows.



About **Nginx** Please go through: https://www.youtube.com/watch?v=mtXE1LMQZEY

6. What is Kestrel Web Server? What is Difference between IIS and Kestrel web server?



Differences

- IIS web server only supports Windows while kestrel web server supports all platforms.
- IIS web servers have HTTP access logs which is not the case in kestrel.
- IIS allows port sharing/multiple apps running at the same time while kestrel doesn't.
- IIS has SSL certificates while kestrel only works with internal SSL traffic between the reverse proxy web servers and .Net.
- IIS has windows authentication while kestrel doesn't.
- IIS has a management console which isn't there in kestrel.
- Process activation is there in case of IIS web servers while it isn't there in kestrel.
- IIS allows application initialization while kestrel doesn't.
- IIS has configuration API while kestrel doesn't.
- IIS allows request filters and limits while kestrel doesn't.
- IIS has IP and domain limitations while kestrel doesn't.
- IIS follows HTTP redirect rules while kestrel doesn't.
- IIS allows WebSocket protocol while kestrel allows middleware.
- IIS does response output caching while kestrel doesn't.
- IIS has FTP servers while kestrel doesn't.