

# How to create my first ASP WebApp with .NET Framework 4.7.2 and how to deploy to Docker Desktop (in your local laptop)

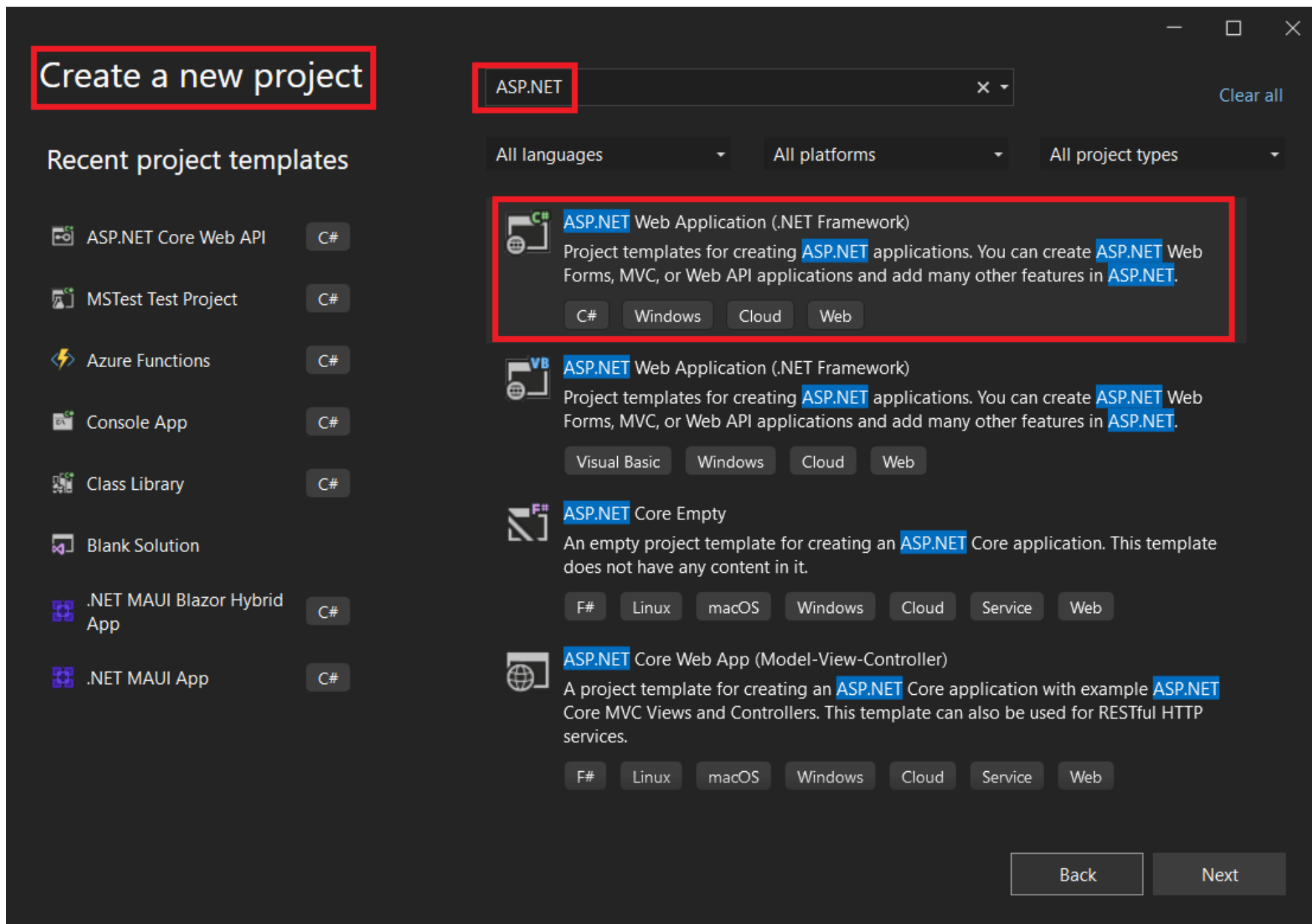
---

## 1. Run Visual Studio and create the WebApp

---

We run Visual Studio 2022 Community Edition and we create a new project

We select the project template



We input the location and the project name

# Configure your new project

## ASP.NET Web Application (.NET Framework)

C# Windows Cloud Web

Project name

WebApplication1

Location

C:\ASP WebApp .NET Framework 7.2\

Solution name ⓘ

WebApplication1

☒ Place solution and project in the same directory

Framework

.NET Framework 4.7.2

Project will be created in "C:\ASP WebApp .NET Framework 7.2\WebApplication1\"

Back Create

We chose the new project main features

# Create a new ASP.NET Web Application



## Empty

An empty project template for creating ASP.NET applications. This template does not have any content in it.



## Web Forms

A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.



## MVC

A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.



## Web API

A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.



## Single Page Application

A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

### Authentication

None

### Add folders & core references

- ☐ Web Forms
- ☒ MVC
- ☐ Web API

### Advanced

- ☒ Configure for HTTPS
- ☐ Docker support  
(Requires [Docker Desktop](#))
- ☐ Also create a project for unit tests

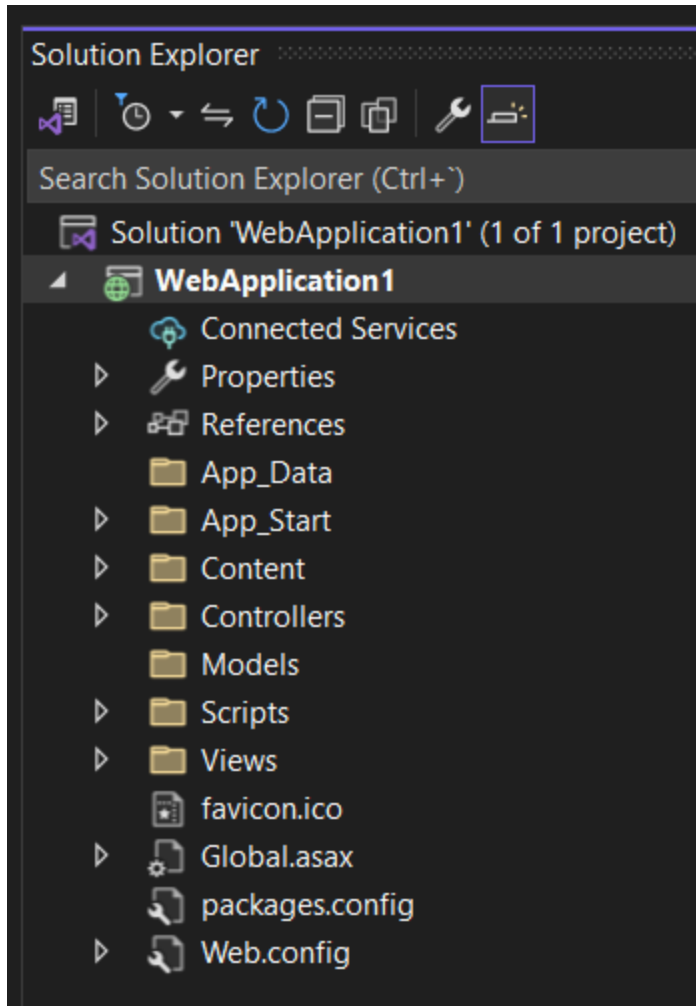
WebApplication1.Tests

Back

Create

## 2. Project folder structure and main files description

## 2.1. Default Folder Structure

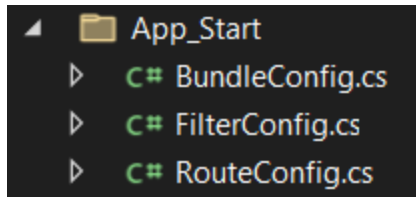


### App\_Data

Used to store application data files like local databases, XML files, etc.

It's a secure folder that is not accessible directly via URLs.

### App\_Start



Contains class files which are executed at application start.

Common files include **RouteConfig.cs** for routing configuration, **BundleConfig.cs** for bundling and minification of CSS and JS files, etc.

## Content

Houses **CSS** files and other static content like **images**.

## Controllers

Contains Controller classes.

Controllers respond to HTTP requests and execute appropriate actions.

## Models

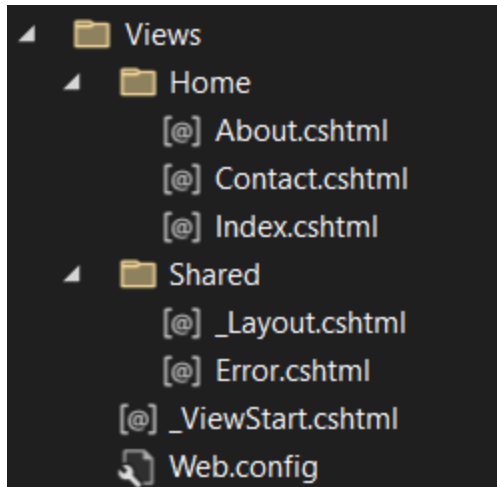
Contains classes representing data models.

These models often correspond to database tables.

## Scripts

Stores JavaScript files and libraries, such as jQuery, Bootstrap JS, etc.

## Views



Holds the Razor view files for the application. It's typically structured into subfolders corresponding to the controllers.

**\_ViewStart.cshtml** - Contains common view settings; executed before the rendering of each view.

**\_Layout.cshtml** - Common layout file (like a template) used by other views.

## Fonts

(Optional) For custom fonts used in your application.

## bin

Contains the compiled assemblies (DLLs) including your application's code.

## 2.2. Main Files

### Global.asax

Allows you to write code that responds to system-level events, such as application start and end.

### Web.config

The main configuration file for the application. It includes settings for databases, security configurations, custom error pages, etc.

## Package.config

Lists the NuGet packages used in the project.

Other Files and Folders

## Properties

Contains the AssemblyInfo.cs file which includes metadata like version number, company name, etc.

## .csproj file

The project file which includes references to all the other files and folders in your project. It's used by Visual Studio to build your application.

## Solution (.sln) file

If your application is part of a larger solution, this file ties together the various projects.

# 3. Workflow after running the WebApp

---

This is the workflow summary:

**Global.asax** -> **App\_Start/RouterConfig.cs** -> <https://localhost:44327/Home/Index> -> Call Index action inside the Home controller  
**Controllers/HomeController.cs** -> Call Home view (**Views/Home/Index.cshtml**) -> the Index.cshtml view is Rendered inside the  
**Views/Shared/\_Layout.cshtml** as defined in **\_ViewStart.cshtml**

## 3.1. When we run the application the first file we start is Global.asax

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.Mvc;
```



```
using System.Web.Optimization;
using System.Web.Routing;

namespace WebApplication1
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
        }
    }
}
```

The **Application\_Start** method is a special method in ASP.NET.

**It is automatically called by the framework when the web application starts.**

This method is typically used to perform application-level **initializations**, such as configuring routes, filters, and bundles:

### **AreaRegistration.RegisterAllAreas()**

This line registers all areas in the application. Areas are an ASP.NET MVC feature that helps organize a large application into smaller functional groupings.

### **FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters)**

This line registers global filters. Filters in ASP.NET MVC can be used for cross-cutting concerns like logging, exception handling, and authorization.

Global filters are applied to all controllers and actions.

### **RouteConfig.RegisterRoutes(RouteTable.Routes)**

This configures the URL routing for the application. Routing is how ASP.NET MVC matches a URI to an action method in a controller.

RegisterRoutes method sets up the routes that the application recognizes.

### **BundleConfig.RegisterBundles(BundleTable.Bundles)**

This registers bundles for JavaScript and CSS files.

Bundling and minification are features that help improve request load time by reducing the number of requests to the server and reducing the size of requested assets (like CSS and JavaScript).

Overall, this MvcApplication class is central to setting up important aspects of the ASP.NET MVC web application when it starts, ensuring that areas, filters, routes, and bundles are all registered and ready to use.

## **3.2. The Global.asax calls the App\_Start/RouterConfig.cs file**

In this file we configure the default entry point in my web application.

In this case we call the Index action inside the Home controller

This is the web application entry point: <https://localhost:44327/Home/Index>

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace WebApplication1
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
```

```

routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
);
}
}
}

```

### 3.3. The WebApp default entry point automatically calls the Controllers/HomeController.cs file

When the application starts with the default route: <https://localhost:44327/Home/Index>

automatically the Index action inside the Home controller is invoked

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace WebApplication1.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}

```

### 3.4. The Index action inside the HomeController calls the Views/Home/Index.cshtml view

```
@{
    ViewBag.Title = "Home Page";
}

<main>
    <section class="row" aria-labelledby="aspnetTitle">
        <h1 id="title">ASP.NET</h1>
        <p class="lead">ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and Jav
        <p><a href="https://asp.net" class="btn btn-primary btn-md">Learn more &raquo;</a></p>
    </section>

    <div class="row">
        <section class="col-md-4" aria-labelledby="gettingStartedTitle">
            <h2 id="gettingStartedTitle">Getting started</h2>
            <p>
                ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that
                enables a clean separation of concerns and gives you full control over markup
                for enjoyable, agile development.
            </p>
            <p><a class="btn btn-outline-dark" href="https://go.microsoft.com/fwlink/?LinkId=301865">Learn more &raquo;</a></p>
        </section>
        <section class="col-md-4" aria-labelledby="librariesTitle">
            <h2 id="librariesTitle">Get more libraries</h2>
            <p>NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visua
            <p><a class="btn btn-outline-dark" href="https://go.microsoft.com/fwlink/?LinkId=301866">Learn more &raquo;</a></p>
        </section>
        <section class="col-md-4" aria-labelledby="hostingTitle">
            <h2 id="hostingTitle">Web Hosting</h2>
            <p>You can easily find a web hosting company that offers the right mix of features and price for your applications.</p>
            <p><a class="btn btn-outline-dark" href="https://go.microsoft.com/fwlink/?LinkId=301867">Learn more &raquo;</a></p>
        </section>
    </div>
</main>
```

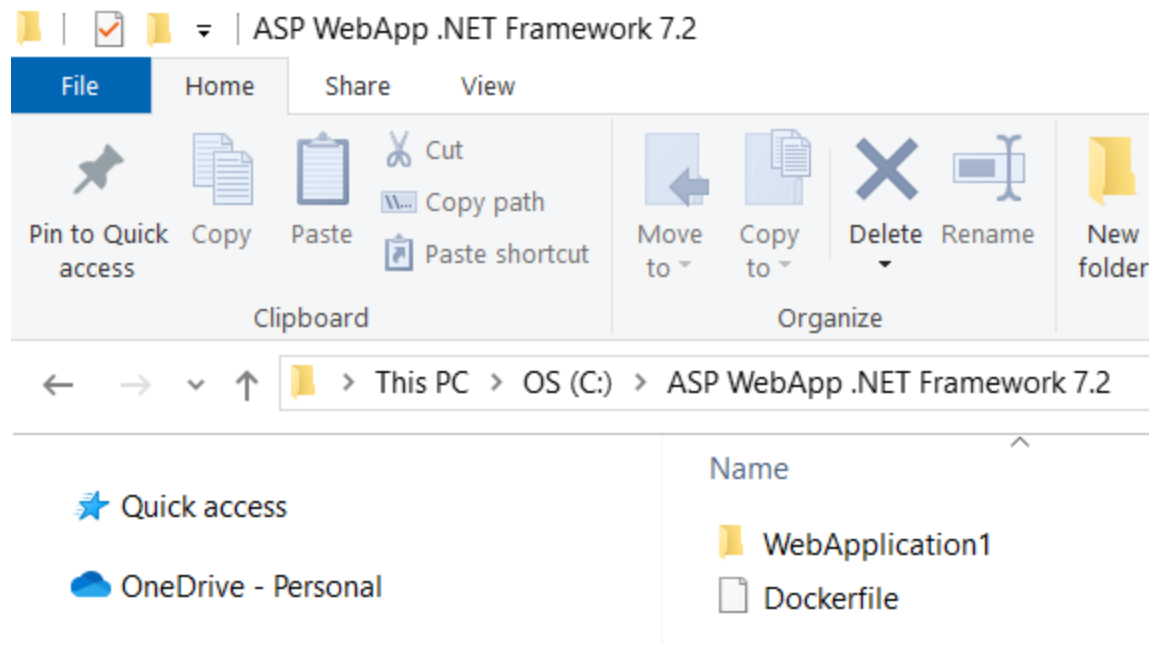
### 3.5. The Views/Home/Index.cshtml view is rendered inside the Views/Shared/\_Layout.cshtml view

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>@ViewBag.Title - My ASP.NET Application</title>
  @Styles.Render("~/Content/css")
  @Scripts.Render("~/bundles/modernizr")
</head>
<body>
  <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-dark bg-dark">
    <div class="container">
      @Html.ActionLink("Application name", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
      <button type="button" class="navbar-toggler" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" title="Toggle"
        aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse d-sm-inline-flex justify-content-between">
        <ul class="navbar-nav flex-grow-1">
          <li>@Html.ActionLink("Home", "Index", "Home", new { area = "" }, new { @class = "nav-link" })</li>
          <li>@Html.ActionLink("About", "About", "Home", new { area = "" }, new { @class = "nav-link" })</li>
          <li>@Html.ActionLink("Contact", "Contact", "Home", new { area = "" }, new { @class = "nav-link" })</li>
        </ul>
      </div>
    </div>
  </nav>
  <div class="container body-content">
    @RenderBody()
    <hr />
    <footer>
      <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
    </footer>
  </div>
```

```
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required: false)
</body>
</html>
```

## 4. Adding docker support to the WebApp application

We create a Dockerfile as shown in the following picture



This is the **Dockerfile** source code

```
# Use a Windows image with .NET Framework 4.7.2
FROM mcr.microsoft.com/dotnet/framework/aspnet:4.7.2-windowsservercore-ltsc2019

# Set the working directory
WORKDIR /inetpub/wwwroot
```

```
# Copy the application files from your host to your container
```

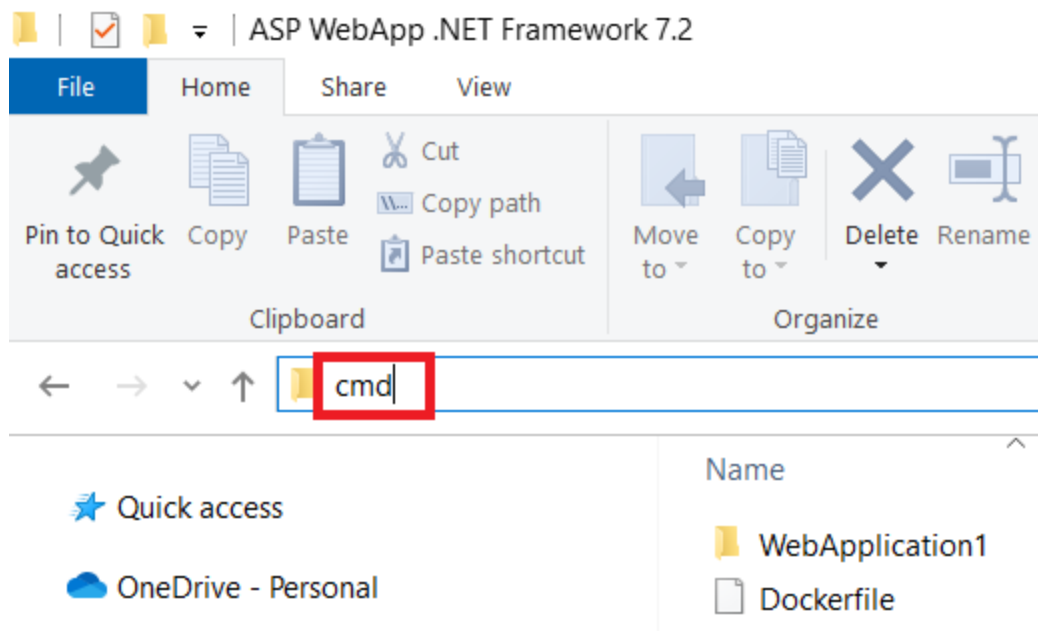
```
COPY ./WebApplication1 .
```

```
# Expose the port the app runs on
```

```
EXPOSE 80
```

```
# The final command to run your application will be in the base image
```

We type **cmd** to open a command prompt window



We create the docker image with the following command

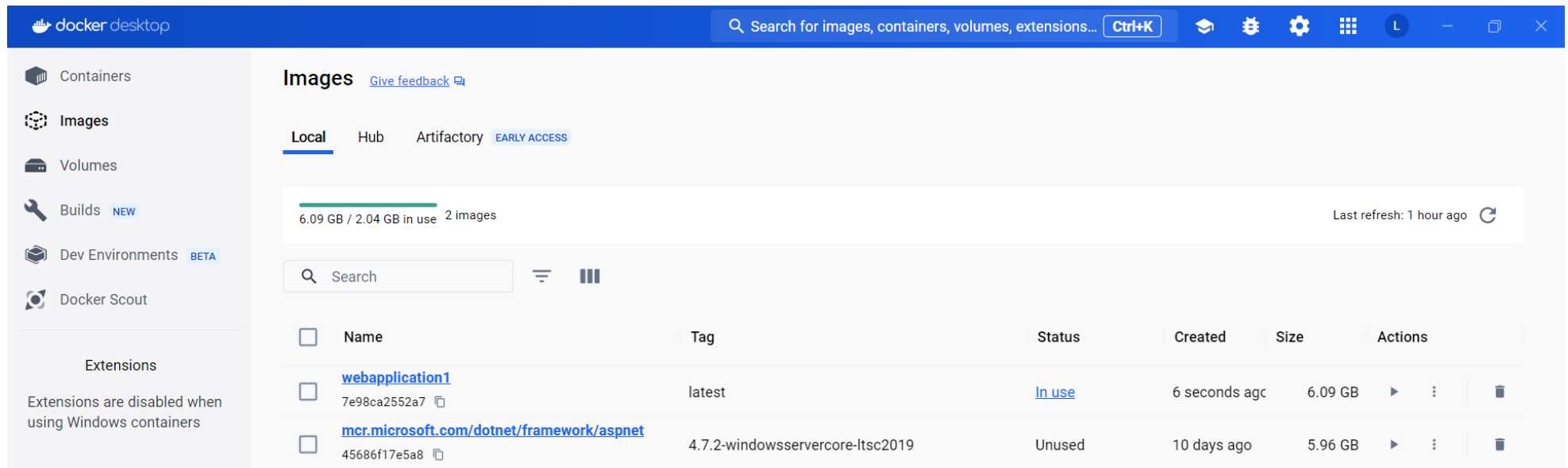
```
docker build -t webapplication1 .
```

```
C:\ASP WebApp .NET Framework 7.2>docker build -t webapplication1 .
Sending build context to Docker daemon 129.8MB
Step 1/4 : FROM mcr.microsoft.com/dotnet/framework/aspnet:4.7.2-windowsservercore-ltsc2019
--> 45686f17e5a8
Step 2/4 : WORKDIR /inetpub/wwwroot
--> Running in 17c65644de41
Removing intermediate container 17c65644de41
--> 81f862387752
Step 3/4 : COPY ./WebApplication1 .
--> 11a78789f40a
Step 4/4 : EXPOSE 80
--> Running in 53b4b9f4998f
Removing intermediate container 53b4b9f4998f
--> 7e98ca2552a7
Successfully built 7e98ca2552a7
Successfully tagged webapplication1:latest

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
```

We also verify the docker image in Docker Desktop





## 5. Run your application with protocol HTTP

Now we can run the docker image with this command

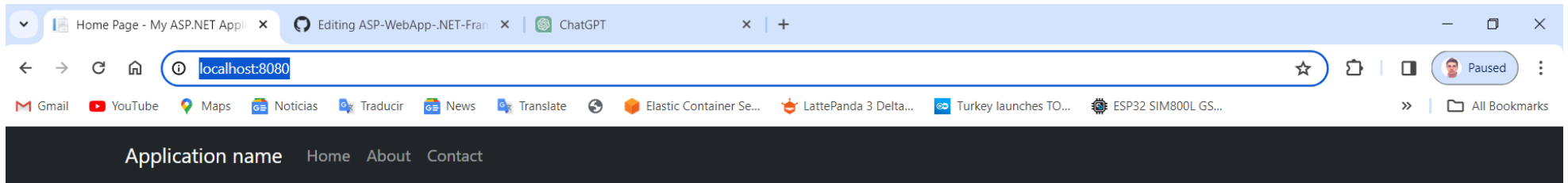
```
docker run -d -p 8080:80 --name myapp webapplication1
```

```
C:\ASP WebApp .NET Framework 7.2>docker run -d -p 8080:80 --name myapp webapplication1
5057e92ceee90b8f9ada9a213a5f438dafdf2348f625ca0acbdbcbf77a227976
```

We verify the application running navigating in the internet web browser to this endpoint:

<http://localhost:8080/>

We see the application running



# ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

[Learn more »](#)

## Getting started

ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.

[Learn more »](#)

## Get more libraries

NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.

[Learn more »](#)


## Web Hosting

You can easily find a web hosting company that offers the right mix of features and price for your applications.






[Learn more »](#)


© 2024 - My ASP.NET Application


We can also verify the running docker container in Docker Desktop


 docker desktop


Ctrl+K


     L — □ ×


 Containers

 Images

 Volumes

 Builds NEW

 Dev Environments BETA



 Docker Scout




Extensions

Extensions are disabled when using Windows containers

## Containers

[Give feedback](#)

  Only show running containers

<input type="checkbox"/>	Name	Image	Status	Port(s)	Last started	Actions
<input type="checkbox"/>	 <b>myapp</b> 5057e92ceee9 	<a href="#">webapplication1</a>	Running	<a href="#">8080:80</a> 	9 minutes ago	