

Tech Spec

Microsoft's .NET Core hosting models



Table of contents

- 1. Glossary of Terms**
- 2. What is a web server? Why do we use it?**
- 3. What is ASP.Net Cores Kestrel (high level)**
 - a. The Pros and Cons**

4. *Choosing a hosting model in ASP.NET core*
5. *What is a proxy server?*
 - a. *The benefits.*
6. *IIS*
 - a. *What is IIS*
 - b. *Why are we using IIS specifically?*
 - c. *What are the benefits of using them?*
7. *NGINX*
 - a. *What is NGINX*
 - b. *Why are we using NGINX specifically?*
 - c. *What are the benefits of using them?*
8. *Which version?*
9. *Overreaching policy. Have a policy of updating, and look at "when does this technology stop getting support"*
10. *Important pre installation Steps*
 - a. *Installing .NET core*
 - b. *Installing visual studio*
11. *Installation.*
 - a. *Configuring kestrel*
 - b. *Configuring NGINX for MACos*
 - c. *Configuring IIS for Windows Systems*
 - i. *Enable IIS*
 - ii. *.Net core hosting bundle*
 1. *Checking what is installed*
 2. *Why are we choosing to download the .NET Core Windows Hosting Pack?*
 3. *Install .NET core hosting bundle*
 - iii. *Restarting IIS*
 - iv. *Demonstrate IIS to host asp.net core application*
 - v. *Deploying website using visual studio*

1) *Glossary of Terms*

2) *What is ASP.Net Cores Kestrel*

<https://stackify.com/what-is-kestrel-web-server/>

Kestrel is used to host ASP.NET applications on any platform. It was launched by microsoft with ASP.NET core.

Why is this important?

The diagram illustrates the ASP.NET Core application architecture. It shows a sequence of components: Internet, Kestrel, HttpContext, and Application code. The Internet is represented by a cloud icon. Kestrel and Application code are represented by blue boxes with a small icon. HttpContext is represented by a double-headed arrow between Kestrel and Application code. The flow is as follows: Internet → HTTP → Kestrel → HttpContext → Application code.

The process of starting up an application will be consistent with cross-platforms, meaning startup does not vary with varying web server configurations.

Is the same process throughout any platform.

	FS	WinFS
Platform Support	Windows	Windows/Linux/OSX
SQL File	No	Yes
HTTP Access Logs	No	No
Port Sharing (Multiple apps)	No	No
SSL Certificate	No	Internal**
Windows Authentication	No	No
Management Console	No	No
Process Activation (start 4 app)	No	No
Application Initialization (start 4 app)	No	No
Configuration API	No	No
Request Filtering & Limits	No	No
IP & Domain Restrictions	No	No
HTTP Redirect Rules	No	No
WebSite Protocol	No	HTTP/HTTPS
Response Output Caching	No	No
Compression	Optional	Optional
FTP Server	No	No

Cons	Pros
<ul style="list-style-type: none"> - One of the problems with ASP.NET was the performance. Therefore, Kestrel and ASP.NET core is many times faster - Kestrel solved the problem of getting ASP.NET to become cross-platform. So now we can write an ASP.NET core application and deploy it to Windows or linux either one. - Kestrel is fast but lacks a lot of functionality, So it relies on a full-fledged web server to deal with things like security, management, etc. More on that later below. 	<ul style="list-style-type: none"> - One of the problems with ASP.NET was the performance. Therefore, Kestrel and ASP.NET core is many times faster - Supported Cross platform

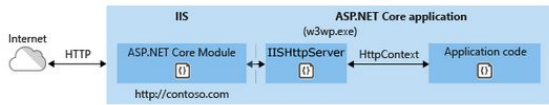
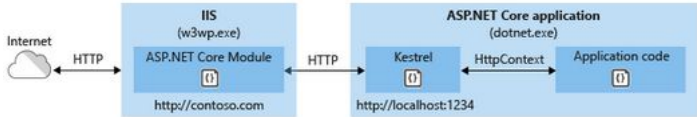
We can see that a web server offers more functionality than kestrel. This is the main reason for our decision of the hosting model.

3) Choosing a hosting model is ASP.NET core

Once our web application is successfully developed what is next? HOSTING. We have to host our application to the server so that other people can access it.

When it comes to deployment to IIS, ASP.NET Core offers two hosting models namely InProcess and OutOfProcess.

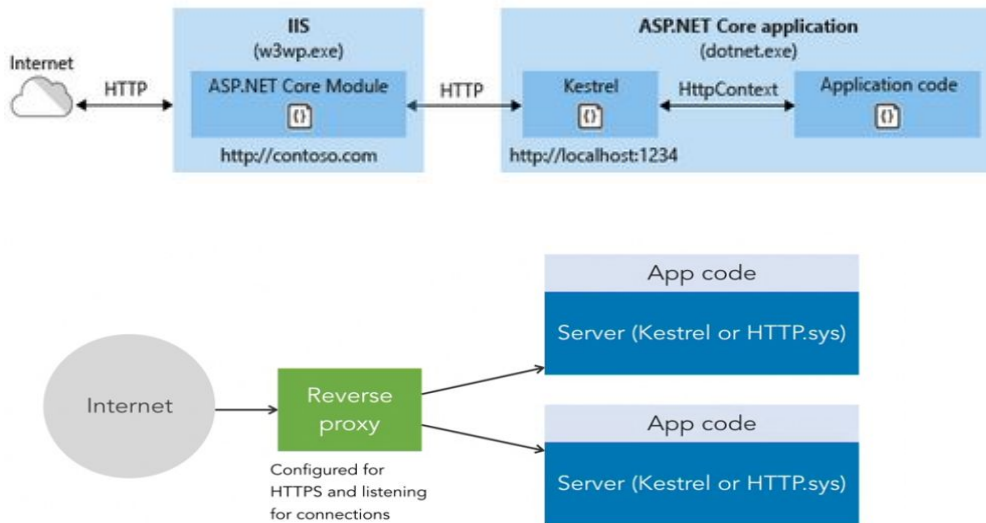
The Differences between the two:

Hosting models:	In process Hosting	Out-of process hosting
Figure:	<p>The following diagram illustrates the relationship between IIS, the ASP.NET Core Module, and an app hosted in-process:</p> 	
Definition:	<ul style="list-style-type: none"> - IIS HTTP Server (IISHttpServer) is used instead of Kestrel server 	<ul style="list-style-type: none"> - Kestrel server is used instead of IIS HTTP Server (IISHttpServer).
Pros	<ul style="list-style-type: none"> - For ASP.NET core 2.2 and later - Offers Better Performance - Less resource intensive as it avoids the extra network hop between IIS and Kestrel and maintaining an additional process on the machine that needs to be monitored. - When hosting in-process, ASP.NET Core module uses an in-process server implementation for IIS, called IIS HTTP Server (IISHttpServer) and avoids the additional cost of reverse-proxying requests over to a separate dotnet process. 	<ul style="list-style-type: none"> - If you want to be 100% compatible between different deployments of the same application, whether it's on Windows or Linux, since Kestrel is the primary mechanism used to handle HTTP requests on all platforms. - Limits the surface area or directly facing the application to the internet.
Cons	<ul style="list-style-type: none"> - Does not use kestrel - Not cross-platform 	<ul style="list-style-type: none"> - Additional cost of reverse-proxying requests over to a separate dotnet process.

Which one will we use?

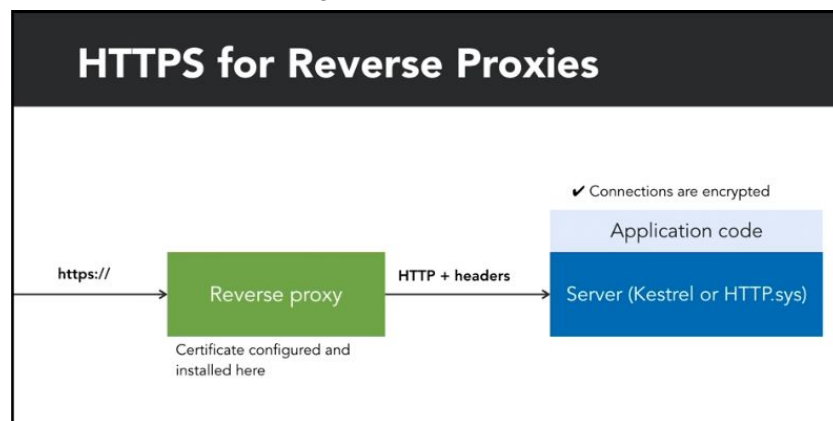
We will be using the “out-of-process” hosting model. Since we want to use different deployments between Windows and MACOs of the same application and want it to be 100% compatible. We want to also add added security and scalability that comes along with putting a reverse proxy facing the internet.

4) What is a proxy server?



a) The benefits.

- i) It limits your exposed surface area of the ASP.NET core application.
- ii) A proxy server provides an additional layer of configuration and defense.
- iii) No need to add a certificate to kestrel to enable HTTPS. A reverse proxy is a single place to configure HTTPS even if you have multiple application codes behind the proxy.



- iv) Makes it easy to scale later, It makes load balancing simplified and send traffic to other servers and offers a secure communication as well(example: HTTPS in figure above)

- v) The ASP.netCoreModule running through IIS also provides process management to ensure that our application gets loaded on first access, ensures that it stays up and running and if there is a crash then restarts happens.

5) IIS

<https://www.pcwold.com/what-is-iis>

a) *What is IIS*

Name: Internet information services

Def: this is windows web server available on most versions of Microsoft's windows operating system. This windows service allows you to host and manage your website.

The IIS hosts websites, web applications and services needed by the developers.

- b) *Why are we using IIS specifically?*
- c) *What are the benefits of using them?*

6) NGINX

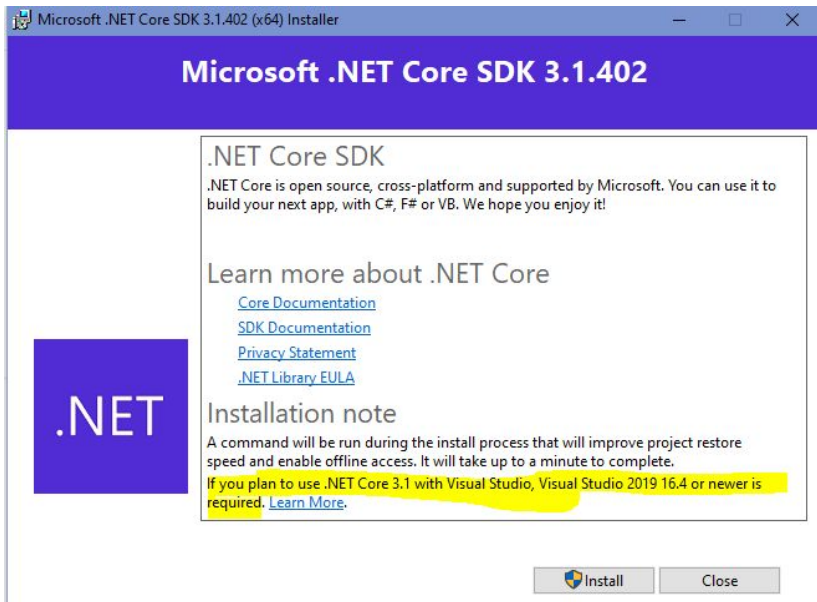
- a) *What is NGINX*
- b) *Why are we using NGINX specifically?*
- c) *What are the benefits of using them?*

7) *Which version?*

8) *Overreaching policy. Have a policy of updating, and look at "when does this technology stop getting support"*

9) *Important Post installation Steps*

- If the Hosting Bundle is installed before IIS(in the next step), the bundle installation must be repaired. Run the Hosting Bundle installer again after installing IIS
- In addition: If the Hosting Bundle is installed after installing the 64-bit (x64) version of .NET Core, SDKs might appear to be missing. So before installing bundle recommended to install the .NET core SDK from: <https://dotnet.microsoft.com/download>



-
- **SDK 3.1.402 has Visual Studio support : Visual Studio 2019 (v16.7) and Visual Studio 2019 for Mac (v8.7.6). SINCE This version of Visual Studio 2019 supports the latest version of Dot Net Core SDK, which is 3.0. Note that you cannot use Visual Studio 2017 to develop Dot Net Core 3 apps.**
- **In the normal cases, you don't have to download the SDK separately since you have installed the latest version of Visual Studio 2019, however, if you open Visual Studio 2019 and do not the see option of Dot Net Core 3 for whatever reason, then you can download the Dot Net Core SDK from the**
- <https://dotnet.microsoft.com/download>
- <https://medium.com/@aram161287/deploy-asp-net-core-web-api-on-iis-f75e755a6402>

10) Installation.

a) Configuring kestrel

b) Configuring NGINX for MACo

<https://rehansaeed.com/nginx-asp-net-core-depth/>

- 1) Install SDK for macOS: <https://dotnet.microsoft.com/download>
- 2) Install NGINX
 - a) mac open command line, the input brew install nginx install nginx.
 - b) installing NGINX using apt-get install nginx
 - c)
- 3) To verify: enter nginx -v see if the installation was successful
- 4) Configuring nginx proxy.
 - a) You've got NGINX running, all you need now is a nginx.conf file to forward requests from the internet to your ASP.NET Core app running using the Kestrel web server.

c) *Configuring IIS for Windows Systems*

i) *Enable IIS*

1. Goto Start-> search control panel
2. "Programs and features"
3. "Turn windows features on or off"

OR

1. Goto Start -> search "apps and features"
2. On the right side click under related setting "Programs and features"
3. Then on the left hand side of the windows click "Turn on windows and features on or off"

Then after you followed one of the two steps above:

1. Look for "internet information services" and "IIS management tool"
2. Enable "Internet Information Services" and expand to
3. make sure "IIS management tool" is enabled as well.
4. Enable "World Wide Web Services" as well.
5. click "ok"
6. After windows completes the requested changes -> Search in the start menu for "IIS manager". (Successfully installed!)

ii) *.NET core hosting bundle*

(1) Check what is installed:

<https://weblog.west-wind.com/posts/2018/Jun/05/Which-NET-Core-Runtime-Download-do-you-need#download-sizes>

<https://docs.microsoft.com/en-us/dotnet/core/tools/dotnet>

- 1) Open the command prompt and type: **"dotnet --info"** :Prints out detailed information about a .NET Core installation and the machine environment, such as the current operating system, and commits SHA of the .NET Core version.
 - a) If it does not work or you get an error, means .NET Core is not installed at all (will later show the steps to fix that)
 - b) What does output show:

- i) Under “.NET Core SDK” shows the active version of SDK
 - ii) Under “Host (useful for support)” shows Runtime version used by the dotnet.exe instance
 - iii) Under “.NET core runtimes installed” shows all the runtimes installed with their file locations.
- c) You can have multiple runtimes and SDKs installed, each project can use a different one and can be specified in your runtime specifier.

(2) Why are we choosing to download the .NET Core Windows Hosting Pack?

- SDK and Runtimes by themselves are usually not the right tool for deployed applications because they don't include everything you need and for this reason - at least on Windows there's a special Hosting Pack download that contains everything you need to run an ASP.NET Core application on Windows.
- What it contains:
 - Everything you need to run an ASP.NET core application on WINDOWS
 - Does not include dotnet SDK tools (IMPORTANT see 9 Post installation steps)
 - This package installs both 32 and 64 bit runtimes, the ASP.NET Core Runtimes as well the IIS hosting components on Windows.

(3) Install .NET core hosting bundle

Steps:

- 1) Goto -> <https://dotnet.microsoft.com/download/dotnet-core> ->
- 2) choose the .NET Core 3.1 (recommended) version
- 3) In the Run apps - Runtime column, ASP.NET Core runtime 3.1.8”
- 4) Download the installer using the Hosting Bundle link “Hosting bundle”
- 5) Agree to the license terms and click install

How to verify:

C:\Program Files\dotnet\shared\Microsoft

iii) Restart IIS

Two options:

- 1) Restart machine
- 2) Restart IIS alone-> open command prompt as administrator -> ” run “iisreset”

```
C:\WINDOWS\system32>iisreset  
  
Attempting stop...  
Internet services successfully stopped  
Attempting start...  
Internet services successfully restarted  
  
C:\WINDOWS\system32>
```

vi. Demonstrate IIS to host asp.net core application

- 1) On the hosting system, create a folder to contain the app's published folders and files.
 - First I created a folder called websites and a subdirectory called HelloCoreWorld as such: C:\Websites\HelloCoreWorld. This is the physical path where the files and binaries are going to live for the website created.
- 2) In windows 10 taskbar search "IIS manager"
- 3) In IIS Manager, open the server's node in the Connections panel. Right-click the Sites folder. Select Add Website from the contextual menu.
- 4) Provide a Site name and set the Physical path to the app's deployment folder. Provide the Binding configuration and create the website by selecting, like so:

The screenshot shows the 'Add Website' dialog box with the following settings:

- Site name:** HelloCoreWorld
- Application pool:** HelloCoreWorld
- Content Directory:**
 - Physical path:** C:\Websites\HelloCoreWorld
- Binding:**
 - Type:** http
 - IP address:** All Unassigned
 - Port:** 80
 - Host name:** localhost
- Start Website immediately:** ☒

An arrow points to the 'OK' button at the bottom right.

- 5) I used Host name: localhost” since this is going to be local testing
- 6) Now we need to modify the setting for the “Application Pool” for .net core:
- 7) Left click on “Application pools” left click on the Application pool just created in our case it is “HelloCorreWorld” and click “Basic setting”
- 8) For the “.NET CLR visions” click “No managed code” .
 - a) ASP.NET Core runs in a separate process and manages the runtime. ASP.NET Core doesn’t rely on loading the desktop CLR (.NET CLR). This says the .NET code will be run and managed by Kestrel.
 - b) For the Dot Net Core Apps to work under IIS, we will have to create a new application pool with the no managed code option. The IIS Application pool will not have any effect on the runtime of the Dot Net Core Apps, it only works as a reverse proxy

Verify:

- 1) Goto “Sites” under “Connections” tab
- 2) click “HelloCoreWORLD”
- 3) On the right hand side panel under “Browse Website” click “ Browse localhost on *:80(http).
- 4) Result: will show an error since currently there are no files in that directory which is okay. This shows that IIS is running and the setup is complete. (will fix this error in a bit)

HTTP Error 403.14 - Forbidden

The Web server is configured to not list the contents of this directory.

Most likely causes:

- A default document is not configured for the requested URL, and directory browsing is not enabled on the server.

Things you can try:

- If you do not want to enable directory browsing, ensure that a default document is configured and that the file exists.
- Enable directory browsing using IIS Manager.
 1. Open IIS Manager.
 2. In the Features view, double-click Directory Browsing.
 3. On the Directory Browsing page, in the Actions pane, click Enable.
- Verify that the configuration/system.webServer/directoryBrowse@enabled attribute is set to true in the site or application configuration file.

Detailed Error Information:

Module	DirectoryListingModule	Requested URL	http://localhost:80/
Notification	ExecuteRequestHandler	Physical Path	C:\Websites\HelloCoreWorld
Handler	StaticFile	Logon Method	Anonymous
Error Code	0x00000000	Logon User	Anonymous

More Information:

This error occurs when a document is not specified in the URL, no default document is specified for the Web site or application, and directory listing is not enabled for the Web site or application. This setting may be disabled on purpose to secure the contents of the server.

[View more information »](#)

vii. Deploying website using visual studio

<https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/iis/?view=aspnetcore-2.1&tabs=aspnetcore2x#data-protection>

Steps:

- 1) Deploy the app to the IIS Physical path folder that was established in the Creating the IIS site section.
- 2) Web deploy with Visual studio community 2019:
- 3) Go to File
- 4) Select new project
- 5) Select Template: “ASP.NET core Web Application”

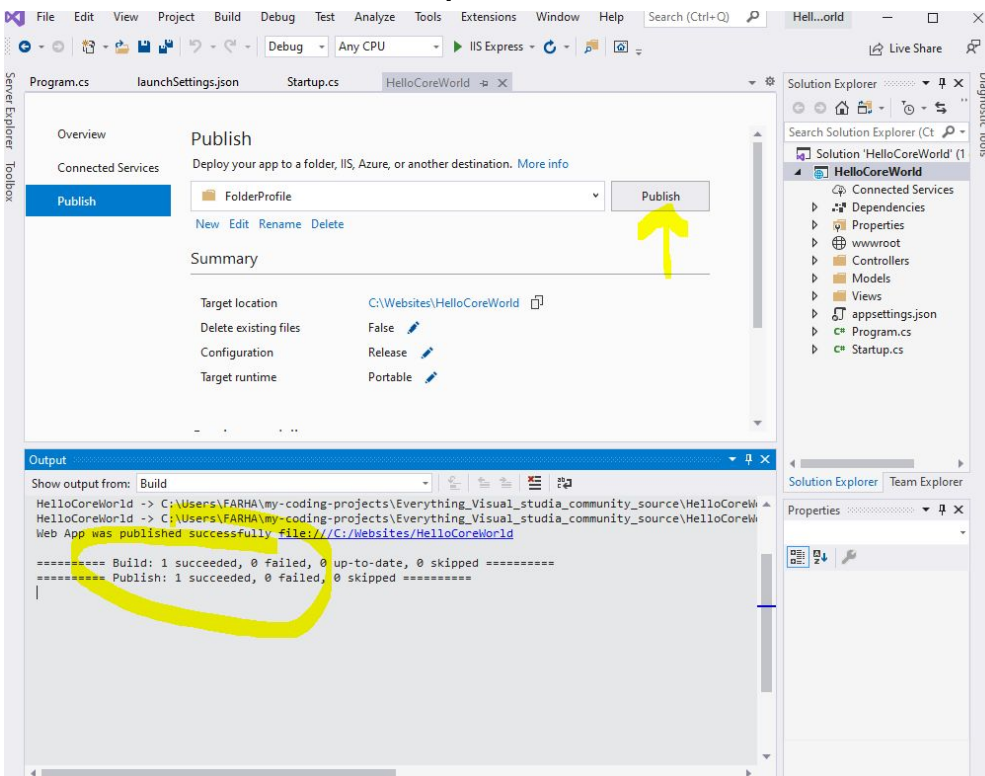


ASP.NET Core Web Application

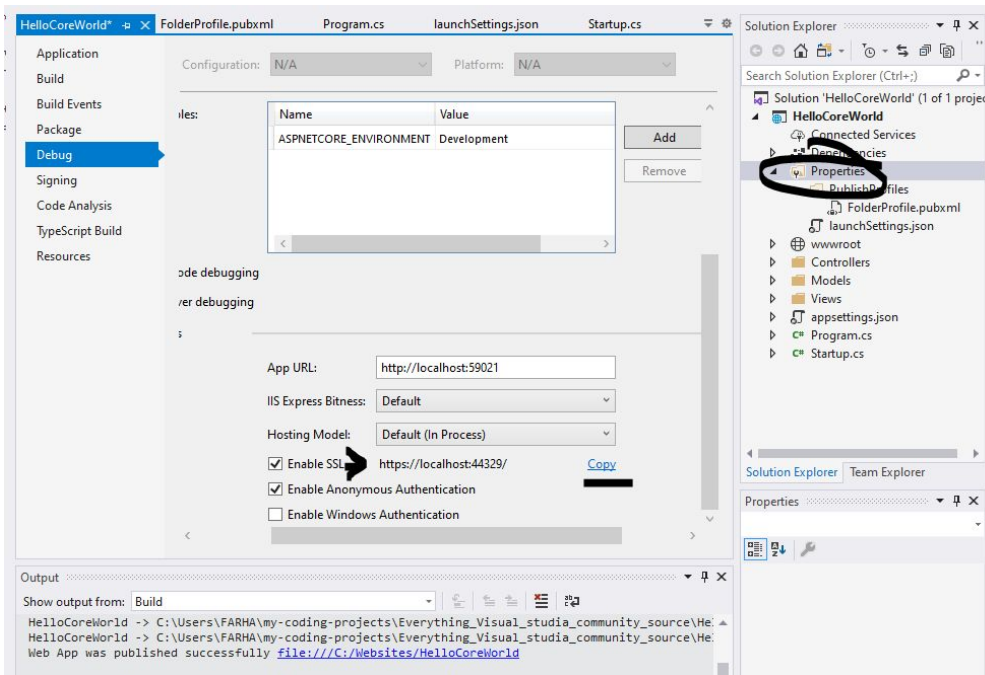
Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.

C# Linux macOS Windows Cloud Service Web

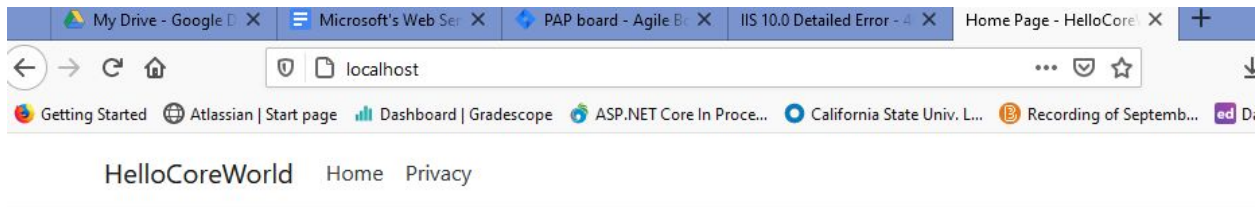
- 6) Click next
- 7) Project name : “HelloCoreWorld”
- 8) Location is the same directory as the one used in the previous section “C:\Websites\HelloCoreWorld. “
- 9) Click “create application”
- 10) Click “Web application (Model-View-Controller)”
- 11) Make sure “.NET core” and “ASP.NET core 3.1” are selected
- 12) On the left side, The Solution Explorer column
- 13) Left click on “HelloCoreworld” project and select “publish”
- 14) Select “Delete existing files”
- 15) Under “File publish options” check off “Delete all existing files prior to publish”
- 16) Press ok the click “Publish” output should show:



17) Then In the solutions column Goto Properties and copy “App URL”



18) Then search:



That is how you successfully publish your application to IIS using visual studio.

References:

What is a proxy server?

- A proxy server receives HTTP requests from the internet and forwards them to kestrel.

Our goal is to make our web application available over the internet. To make this web server public we must host it on a web server. The web server is responsible for processing the HTTP requests from users to our website via the URI. IIS is a microsoft windows component that works as a web server. In addition NGINX is a web server that works with MACos systems.