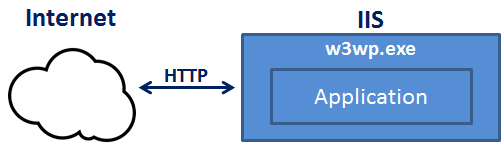
**ASP.NET Core out of process hosting**

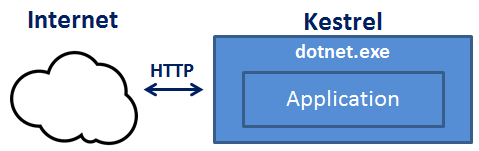
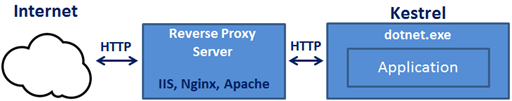
**InProcess Hosting in ASP.NET Core**   
  
To configure InProcess hosting, add <AspNetCoreHostingModel> element to the app's project file with a value of InProcess

<AspNetCoreHostingModel>InProcess</AspNetCoreHostingModel>

With **InProcess hosting**, the application is hosted in the IIS worker process (w3wp.exeor iisexpress.exe). With InProcess hosting, there is only one web server and that is the IIS server that hosts our application.   
  
   
  
  
**Out of Process Hosting in ASP.NET Core**   
  
There are 2 ways to configure **Out of Process hosting**   
  
**Option 1 :** Add <AspNetCoreHostingModel> element to the app's project file with a value of OutOfProcess

<AspNetCoreHostingModel>OutOfProcess</AspNetCoreHostingModel>

**Option 2 :** The default is OutOfProcess hosting. So if we remove the <AspNetCoreHostingModel> element from the project file, OutOfProcess hosting will be used by default.   
  
With **out of process hosting**

* There are 2 web servers - An an internal web server and an external web server.
* The internal web server is Kestrel and the external web server can be IIS, Nginx or Apache.
* Depending on how you are running the asp.net core application, the external web server may or may not be used.    
    
  **Kestrel is a cross-platform web server** that is embedded in your ASP.NET Core application. With Out of Process Hosting model, Kestrel can be used in one of the following 2 ways.   
    
  **Kestrel can be used as the internet facing web server** that process the incoming HTTP requests directly. In this model we are not using an external web server. Only Kestrel is used and it is this server that faces the internet, to directly handle the incoming HTTP requests. When we run the asp.net core application using the .NET core CLI, Kestrel is the only web server that is used to handle and process the incoming HTTP request.   
    
     
    
  **Kestrel can also be used in combination with a reverse proxy server**, such as IIS, Nginx, or Apache.   
    
     
    
  **If Kestrel can be used by itself as a web server, why do we need a reverse proxy server.**  
  With **Out of Process Hosting**, using a reverse proxy server is a good choice as it provides an additional layer of configuration and security. It might integrate better with the existing infrastructure. It can also be used for load balancing.    
    
  So, with a reverse proxy server in place, it receives incoming HTTP requests from the network and forwards them to the Kestrel server for processing. Upon processing the request, the Kestrel server sends the response to the reverse proxy server which then ultimately sends the response to the requested client over the network.   
    
  We will discuss **Deploying ASP.NET Core application to IIS**and using IIS as a reverse proxy server in our upcoming videos. When we run an asp.net core application directly from Visual Studio it uses by default IIS Express. Since we have configured our application to use Out of Process hosting, IIS Express in this case acts a reverse proxy server.   
    
  **IIS Express** takes the incoming HTTP request and forwards it to Kestrel for processing. Kestrel process the request and sends the response to IIS Express. IIS Express, in turn sends that response to the browser.   
    
  With **Out of Process Hosting**, whether you use a reverse proxy server or not, it is the Kestrel server that hosts the application and process the request. The reverse proxy server if used, takes the incoming HTTP request and forwards it to the Kestrel server. Similarly it takes the response from the Kestrel server and sends it to the client. So the name of the process that hosts the application is dotnet.exe.   
    
  **Use the following code to get the process name**

System.Diagnostics.Process.GetCurrentProcess().ProcessName

When we run the asp.net core project using the .NET Core CLI, by default it ignores the hosting setting we specified in the csproj file. So the AspNetCoreHostingModel element value in the **csproj file**is ignored.    
  
Irrespective of the value you specified (InProcess or OutOfProcess), it always uses OutOfProcess hosting and Kestrel is the web server that hosts the application and handle the http requests.   
  
**One common question :**Can we run an asp.net core application without using the built in kestrel web server.  
The answer is YES. If we use the InProcess hosting model, the application is hosted inside of the IIS worker process (w3wp.exe or iisexpress.exe). Kestrel is not used with InProcess hosting model.