**ASP.NET Core in process hosting**

* In process hosting model in ASP.NET Core
* What is Kestrel server

When an ASP.NET core application is executed, the .NET runtime looks for Main()method which is the entry point for the application. The Main() method then calls CreateDefaultBuilder() static method of the WebHost class.    
  
  
This CreateDefaultBuilder() method performs several tasks like

* Setting up the web server
* Loading the host and application configuration from various configuration sources and
* Configuring logging

let's understand what the CreateDefaultBuilder() method does to configure and set up the web server. An ASP.NET core application can be hosted InProcess or OutOfProcess.  
  
**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>**  
  
When we create a new ASP.NET Core project using one of the available project templates, the project defaults to the in-process hosting model for all IIS and IIS Express scenarios.  
  
In case of InProcess hosting, CreateDefaultBuilder() method calls UseIIS() method and host the app inside of the IIS worker process (w3wp.exe or iisexpress.exe).

* From a **performance standpoint**, InProcess hosting delivers significantly higher request throughput than OutOfProcess hosting
* In the case of IIS, the process name that executes the app is **w3wp**and in the case of IIS Express it is **iisexpress**
* To get the process name executing the app, use System.Diagnostics.Process.GetCurrentProcess().ProcessName
* When we are run the project from Visual Studio it uses IISExpress by default.
* IIS Express is a lightweight, self-contained version of IIS, optimized for application development. We do not use it for production. In production we use IIS.

**With out of process hosting**

* There are 2 web servers - 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.
* With **InProcess hosting**, there is only **one web server**i.e the IIS that hosts the asp.net core application.
* So, **we do not have the performance penalty**of proxying requests between internal and external web server.

**What is Kestrel**  
Kestrel is a cross-platform web server for ASP.NET Core. It is supported on all platforms and versions that .NET Core supports. It is included by default as internal server in ASP.NET Core. Kestrel can be used, by itself as an edge server i.e Internet-facing web server that can directly process the incoming HTTP requests from the client. In Kestrel, the process used to host the app is **dotnet.exe (or) Appname.**  
  
When we run a .NET Core application using the .NET Core CLI (Command-Line Interface), the application uses **Kestrel as the web server**.   
  
**The .NET Core CLI is a cross-platform tool** for developing .NET core applications. Using the CLI we can

* Create a new project, configuration file, or solution based on the specified template
* Restore the dependencies and tools required for a .net core project
* Build a project and all of its dependencies
* Run a project etc...

There are a broad range of things that we can do with the .NET Core CLI. **To run our asp.net core application using the .NET Core CLI.**

* Fire up Windows Command Prompt
* Change the directory to the folder that contains your asp.net core project and execute dotnet run command
* C:\Projects\EmployeeManagement\EmployeeManagement>**dotnet run**

After the .NET Core CLI builds and runs the project, it shows the URL using which we can access the application. In my case the application is available at http://localhost:5000   
  
In case of Kestrel, the process used to host and execute the app is **dotnet.exe**. So when we navigate to **http://localhost:5000**, we will see the process name **dotnet**displayed.