**Dot Net Core**

* .Net Core is a modular and high performance implementation of .Net for Creating web applications and services that run on windows Linux and Mac

**Characteristics of .Net Core**

1. **Cross Platform** - Runs on Windows Linux and Mac OS
2. **Open-Source**
3. **Compatible**: .Net core is compatible with .Net Framework, Xamarin and Mono
4. **Flexible Deployment** – Can be included in your app or installed side by side user or machine wide.
5. **Performance**: One of its biggest benefits of Asp.net core is – performance. When you compile your code, it automatically optimizes it in order to increase the performance.
6. **Support for Cloud-Based Development**: It is always better to develop cloud-based applications. For large enterprises, Asp.net core provides the development of various type of web applications, Mobile back-end, Internet of things apps and much more!

**Use .NET Core for your server application when:**

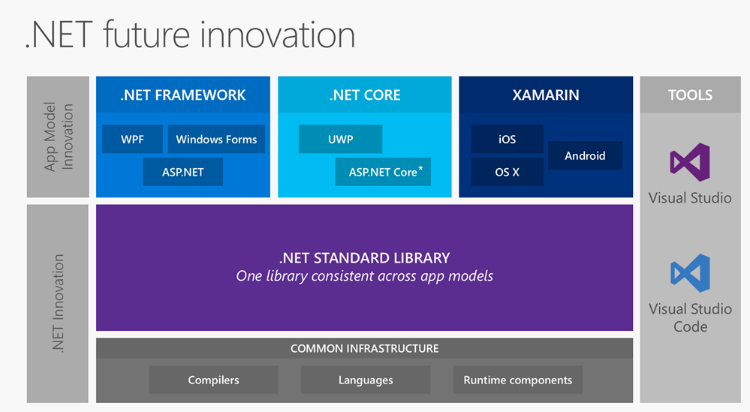
* You have cross-platform needs.
* You are targeting microservices.
* You are using Docker containers.
* You need high-performance and scalable systems.
* You need side-by-side .NET versions per application.

**A need to use .NET technologies not available for .NET Core**

Some .NET Framework technologies aren't available in .NET Core. Some of them might be available in later .NET Core releases. Others don’t apply to the new application patterns targeted by .NET Core and may never be available. The following list shows the most common technologies not found in .NET Core:

1. **ASP.NET Web Forms applications:** ASP.NET Web Forms are only available in the .NET Framework. ASP.NET Core cannot be used for ASP.NET Web Forms. There are no plans to bring ASP.NET Web Forms to .NET Core.
2. **ASP.NET Web Pages applications:** ASP.NET Web Pages aren't included in ASP.NET Core.
3. **WCF services implementation**. Even when there’s a WCF-Client library to consume WCF services from .NET Core, WCF server implementation is currently only available in the .NET Framework. This scenario is not part of the current plan for .NET Core but it’s being considered for the future.
4. **Workflow-related services**: Windows Workflow Foundation (WF), Workflow Services (WCF + WF in a single service) and WCF Data Services (formerly known as "ADO.NET Data Services") are only available in the .NET Framework. There are no plans to bring WF/WCF+WF/WCF Data Services to .NET Core.
5. **Language support**: Visual Basic and F# are currently supported in .NET Core, but not for all project types.

.**NET Core 2.2 Framework Architecture**



**ASP.Net Core**

ASP.NET Core is an open-source, cross-platform framework for building modern, cloud-based web apps on Windows, macOS, or Linux.

* Its significant redesign of ASP.Net
* Open Source and Cross Platform framework for building modern cloud based internet connected applications such as web apps, IOT apps and mobile backends.
* ASP.Net Core apps can run on .Net Core or full .Net framework.
* Consists of modular components with minimal overhead.
* Develop and Run your ASP.Net Core apps cross platform on windows ,Mac and Linux
* Its not based on System.Web.dll rather based on well factored Nuget Packages This allows us to optimize your app to include just the Nuget packages you need

**Why ASP.Net Core**

* Unified for building WebUI and WebAPI
* Architected for testability.
* Razor Pages makes coding page-focused scenarios easier and more productive.
* Integration of Modern Client side frameworks
* Cloud ready environment based configuration system
* Built in DI
* New lightweight and modular Http Request Pipeline
* Ability to host on IIS, Nginx, Apache, Docker, or self-host in your own process.
* Built on .Net core that supports side by side app versioning
* Ships entirely as nuget package
* Open Source

**The following table compares ASP.NET Core to ASP.NET 4.x.**

| **ASP.NET Core** | **ASP.NET 4.x** |
| --- | --- |
| Build for Windows, macOS, or Linux | Build for Windows |
| [Razor Pages](https://docs.microsoft.com/en-us/aspnet/core/razor-pages/index?view=aspnetcore-2.2) is the recommended approach to create a Web UI as of ASP.NET Core 2.x. | Use [Web Forms](https://docs.microsoft.com/en-us/aspnet/web-forms), [SignalR](https://docs.microsoft.com/en-us/aspnet/signalr), [MVC](https://docs.microsoft.com/en-us/aspnet/mvc), [Web API](https://docs.microsoft.com/en-us/aspnet/web-api/), [WebHooks](https://docs.microsoft.com/en-us/aspnet/webhooks/), or [Web Pages](https://docs.microsoft.com/en-us/aspnet/web-pages) |
| Multiple versions per machine | One version per machine |
| Develop with Visual Studio, [Visual Studio for Mac](https://www.visualstudio.com/vs/visual-studio-mac/), or [Visual Studio Code](https://code.visualstudio.com/) using C# or F# | Develop with Visual Studio using C#, VB, or F# |
| Higher performance than ASP.NET 4.x | Good performance |
| [Choose .NET Framework or .NET Core runtime](https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server) | Use .NET Framework runtime |

**Kestrel web server**

Kestrel is a lightweight HTTP server that handles ASP.NET Core applications request pipeline.

With ASP.NET core, there is a Kestrel web server within the ASP.NET Core Application which handles the request. So, the IIS will hand off the request to the Kestrel web server and kestrel processes the request. Now, how is this an open source? Yes, it is open source. Instead of IIS, for Mac users, they can have Nginx/Apache as a server to handle the requests and send it to kestrel.