Hi this is Scott, and for this webinar Pluralsight asked me to talk about ASP.NET Core and the impact this new version of asp.net has on the enterprise.

When larger companies with larger development teams ask me about asp.net core, I generally frame the conversation in terms of risk and reward. Yes, we’ll talk about the new architecture and new features, but when working on critical business applications with long lifecycles, you want to pay attention to the risks as well as the rewards and be prepared for what you are getting into.

I feel there are six areas to consider in evaluating ASP.NET Core and it’s impact on your business and operations. First is understanding .NET Core and ASP.NET Core’s relationship with .NET Core, which is the new .NET framework. Second there is the new hosting model for ASP.NET. Third is the new HTTP processing pipeline in ASP.NET which you build completely from scratch using what we call middleware components. Fourth is the new design and architecture for implementing security requirements for user authentication and authorization. Fifth would be your data access considerations, because there is a new version of the entity Framework available in this new world. Lastly, there is ASP.NET Core itself. What advantages and disadvantages does the new framework bring to the application and application developers? These six areas are what I talk about and represent an outline for the rest of the conversation.

# ASP.NET Core and .NET Core

One of the first topics to understand about the new world is we now have two distinct flavors of the .NET framework to choose from. First there is the full .NET framework, this is the mature .NET framework that has been around since the beginning, this is the framework that comes pre-installed with Windows, and the most recent version of the full .NET framework is 4.6.2. Then we have .NET Core, a brand and modular version of .NET that runs on more than just the Windows operating system.

Where ASP.NET Core fits into the picture is that ASP.NET Core is an application framework that can run on either the full .NET framework or on .NET Core. Selecting a .NET framework flavor is one of your first decisions when making a move to ASP.NET Core. Do you want to run on the full framework, or .NET Core, or will you chose to support both?

When you choose to run on the full .NET framework you are running on the framework that also includes Windows Forms, ASP.NET Web Forms, Windows Communication Foundation, and Windows Presentation Foundation. Although ASP.NET Core will be a bit different from the ASP.NET frameworks of the past, your underlying when choosing the full .NET framework your underlying framework is the same as it has been in the past. In order to understand why you might chose .NET Core instead of the full framework we need to dig into the benefits of .NET Core.

## The Rewards

.NET Core is a cross-platform framework, meaning.NET Core will run on Windows, on the Mac, and on various flavors of Linux. .NET Core also works in a Docker container for those who are using or thinking about using Docker software containers.

I want to make clear that ASP.NET Core does not require .NET Core as the underlying framework. and we’ll circle back to this topic later. But, if you *do* choose to use .NET Core as your underlying framework, you will be able to deploy applications on all of these various platforms. Linux is, of course, a big target for server-side applications. Most enterprises are heterogeneous and already have the IT expertise to run business applications on both Windows and Linux servers, so you can re-use this existing expertise. There is also the opportunity to save money as Linux servers typically run a bit cheaper, particularly when using cloud based infrastructure. On Azure, for example, a single 4 core virtual machine running Linux is around $90 cheaper per month than its Windows counterpart.

What’s not immediately obvious when talking about a cross-platform .NET framework is how the necessary tooling also works across platforms. .NET Core has no hard dependency on Windows or Visual Studio. All of the low level tools you need to do work will run from the command line, and there is an entire new world of text editors and IDEs available that we can now use to develop .NET applications. This includes Visual Studio code from Microsoft, Project Rider from Jet Brains, as well as editors like Sublime and Atom. I’ve worked on more than one project over the years where there is a front-end specialist who uses a Mac and this developer hasn’t been able to install Visual Studio and work with an ASP.NET project the same way a Windows developer would work with the project. This type of scenario is considerably easier with .NET Core because a developer on Apple hardware can write, run, and debug ASP.NET code just as well as a developer on Windows.

.NET Core is also an open source project. On GitHub you can find not just the code for the framework itself, but also the code for the unit tests, and the documentation. You can view bugs in the GitHub issues list and see the current status of a bug. I’ve been telling people that one of the underappreciated advantages of Microsoft’s move towards open source is not in having the source code to a framework or library. We’ve always been able to get source code, even if we had to use a de-compiler. There have been a number of times this year where I’ve found digging into the unit test code to be valuable. The unit tests have given me better insight into how a particular feature works compared to using documentation or the source code itself, because the unit tests can help describe a feature from several different perspectives and also tell me what a piece of software is not designed to do.

In the enterprise, there is often a worry that open source projects do not have the same level of support as closed source commercial products. However, Microsoft has announced a support lifecycle of 3 years for each major and minor release of .NET Core. The .NET Core 1.0.0 release was June 27, 2016, meaning the end of support for 1.0.0 is in June of 2019, or even later if there is a Long Term Support (LTS) release in the future.

Something you’ll notice when looking at the source code on GitHub is how modular the new .NET Core has become. This is no longer a monolithic framework where you have everything or you have nothing at all. With .NET Core the fundamental pieces are smaller and integrated with the NuGet package management ecosystem. You can pull in just the pieces of .NET Core that you need to build an application.

## The Risks

Making a new version of the .NET framework that works across platforms required some hard work and sacrifices. What exactly is in this version of .NET Core and who will be able to use it?

First, you should know

1. Introduction
2. .net core
3. Hosting
4. middleware
5. Security and identityserver
6. Data access
7. Asp.net core itself
   1. Merge webapi and mvc
   2. Fiel system based
   3. Tag helpers
   4. Config is new
8. Summary – the big reboot