What is Kestrel (vs IIS / Express)

111

25

What is the kestrel web server and how does it relate to IIS / IIS Express?

I come from developing apps on IIS Express and hosting them on an IIS web server. With ASP.NET Core I have a dependency on `Microsoft.AspNetCore.Server.Kestrel` and my startup has `.UseServer("Microsoft.AspNetCore.Server.Kestrel")`. But when I run my website, I still get the IIS Express icon in the system tray. Someone asked me if I was using IIS Express or Kestrel and I didn't know what to say!

I don't have any cross-platform requirements as I develop on a PC and host in Azure, so I'm confused if I even `need` Kestrel, but it doesn't seem like there's an alternative - even the simplest samples use Kestrel.

asp.net iis asp.net-core kestrel-http-server

edited Feb 25 '16 at 22:09
vcsjones
108k 23 252 256

asked Feb 25 '16 at 21:58
Sean
7,727 9 56 89

When you have a question about this new technology, start at the GitHub page for the projects in question and look at their Wikis. You'd run across this Servers wiki page for the ASP.NET repo. – mason Feb 25 '16 at 22:02

7 Of course, then you run into stuff like `This document is now out of date. For up-to-date ASP.NET Core documentation go to: http://docs.asp.net`. Oops. – user4942583 Dec 29 '16 at 19:57

stackoverflow.com/questions/42382317/...? – Deepak Mishra Dec 26 '18 at 12:35

3 Answers

92

> What is Kestrel

It's a full blown web server. You can run your ASP.NET Core application using just Kestrel.

> But when I run my website, I still get the IIS Express icon in the system tray

In your ASP.NET application, probably in the `wwwroot` directory, you'll see a web.config that contains this:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
<system.webServer>
```

```
stdoutLogEnabled="false" startupTimeLimit="3600"/>
</system.webServer>
</configuration>
```

This is the HttpPlatformHandler. Essentially, what this does is forward *all* requests to Kestrel. IIS Express (and IIS for that matter) will not run ASP.NET themselves anymore. Instead, they will act as proxies that simply pass requests and responses back and forth from Kestrel. There is still advantages of using IIS, specifically it gives you security configuration, kernel-level caching, etc.

answered Feb 25 '16 at 22:07
vcsjones
108k 23 252 256

2 excellent intro to what exactly is going on under the covers when using ASP.Net core youtu.be/e2qZvabmSvo – user99513 Jan 14 '17 at 17:56

2 This answer is a little bit out-of-date due to the introduction of ASP.NET Core Module (instead of HttpPlatformHandler). I offered an alternative answer with more stories and related products as well. – Lex Li Oct 22 '17 at 20:10

Answer below from @LexLi is awesome, a must-read. – Jim Aho May 11 '18 at 7:28

92

I'd like to offer an alternative answer, with some history, so that you might understand why Kestrel comes, even if you only use Windows and IIS.

At the very beginning of ASP.NET development before year 2000, clearly Microsoft created two pieces to host ASP.NET WebForms apps,

- Cassini, later became ASP.NET Development Server in Visual Studio. It is a fully managed web server written in C# based on `HttpListener` . Of course, since it was for development only, many features were never implemented. As Microsoft made the source code of Cassini available for the public, there are third parties who forked the code base and added more features, which started the Cassini family.
- ASP.NET support on IIS (revision 1). Because IIS was 4.0 and 5.0/5.1 at that time, which has nothing like application pools, ASP.NET even has its own worker process (`aspnet_wp.exe`).

So to develop a web app, you use Cassini, and to deploy you use IIS.

- The introduction of application pools in IIS 6 required some changes on ASP.NET side, so `aspnet_wp.exe` became obsolete and replaced by `aspnet_isapi.dll` . That can be seen as ASP.NET support on IIS revision 2. So ASP.NET apps are being hosted in IIS worker processes `w3wp.exe` .
- The introduction of integrated pipeline in IIS 7 and above required further changes, which replaced `aspnet_isapi.dll` with `webengine4.dll` . That can be seen as ASP.NET support on IIS revision 3. ASP.NET and IIS pipelines are unified.

You can see ASP.NET has become much more complex and tightly integrated with IIS, so Cassini performance metrics in mind (as WebForms focuses quite a lot of productivities and RAD).

Then in November 2014, ASP.NET 5 (later renamed to ASP.NET Core) was announced and became a cross platform technology. Obviously Microsoft needed a new design to support Windows, macOS, and Linux, where all major web servers, nginx/Apache (or other web servers) should be considered besides IIS.

I think many would agree that Microsoft learned quite a lot from NodeJS, and then designed and developed Kestrel (based on `libuv` initially but might move to other technology soon). It is a light-weight web server like Cassini initially, but later more features are being added (like another answer commented, much more features so can be treated as a full web server). Though fully managed (some native dependencies exist), it is no longer a toy web server like Cassini.

Then why cannot you just use Kestrel? Why IIS Express and potentially IIS, nginx, or Apache are still needed? That primarily is a result of today's internet practice. Most web sites use reverse proxies to take requests from your web browsers and then forward to the application servers in the background.

- IIS Express/IIS/nginx/Apache are the reverse proxy servers
- Kestrel/NodeJS/Tomcat and so on are the application servers

Another answer already showed a link to Microsoft documentation, so you can take a look.

Microsoft developed HttpPlatformHandler initially to make IIS a good enough reverse proxy for Java/Python and so on, so planned to use it for ASP.NET Core. Issues started to appear during development, so later Microsoft made ASP.NET Core Module specifically for ASP.NET Core. That's ASP.NET support on IIS revision 4.

Starting from ASP.NET Core 2.2, ASP.NET Core Module for IIS (version 2) can host .NET Core environment inside IIS worker process (`w3wp.exe`), quite similar to ASP.NET 2.x/4.x. This mode is called "IIS in-process hosting". It can be considered as ASP.NET support on IIS revision 5.

Well, quite lengthy, but I hope I put all necessary pieces together and you enjoy reading it.

edited Dec 28 '18 at 18:02

answered Oct 22 '17 at 20:08
Lex Li
43.7k 6 82 106

Nice answer. But you cannot simply say that using kestrel with IIS is 'result of today's internet practice'. There are many rationales of using a reverse proxy. Would have been good to mention a few here. – Nilay Vishwakarma Nov 15 '18 at 13:19

5 "There are many rationales of using a reverse proxy" belongs to its own question and answer. Usually people can find good resources by asking Google, so I didn't append that to this already-long-enough answer. – Lex Li Nov 15 '18 at 14:16

Excellent history lesson leading up to Kestrel's purpose. – Metro Smurf Feb 19 at 14:48

> Kestrel is a cross-platform web server for ASP.NET Core based on libuv, a cross-platform asynchronous I/O library. Kestrel is the web server that is included by default in ASP.NET Core project templates.
>
> You can use Kestrel by itself or with a reverse proxy server, such as IIS, Nginx, or Apache. A reverse proxy server receives HTTP requests from the Internet and forwards them to Kestrel after some preliminary handling.

UPDATE: .net core 2.1, Kestrel uses managed sockets instead if libuv

From asp.net core 2.1 docs at: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel?view=aspnetcore-2.1#transport-configuration

> With the release of ASP.NET Core 2.1, Kestrel's default transport is no longer based on Libuv but instead based on managed sockets.

edited May 20 at 11:32

answered Sep 13 '17 at 9:52
Max
2,465 21 28