⭐ 1. What is Kestrel?

Kestrel is the default cross-platform web server used by ASP.NET Core applications.

✓ Runs on **Windows, Linux, macOS**
✓ Extremely **fast** (built on top of **libuv** initially, now on **Sockets**)
✓ Designed for **high performance** and **asynchronous I/O**
✓ Always self-hosted (no dependency on IIS)

When you run an ASP.NET Core app (dotnet run) → Kestrel is the server listening & serving requests.

⭐ 2. Why Kestrel? (Key Features)

✅ Cross-Platform

Runs anywhere .NET runs (cloud, containers, Linux VMs, macOS, Windows).

✅ High Performance

It beats IIS and Apache in raw speed.
Used by **Azure App Service**, **Docker**, **Microservices**, **Cloud-native** apps.

✅ Lightweight

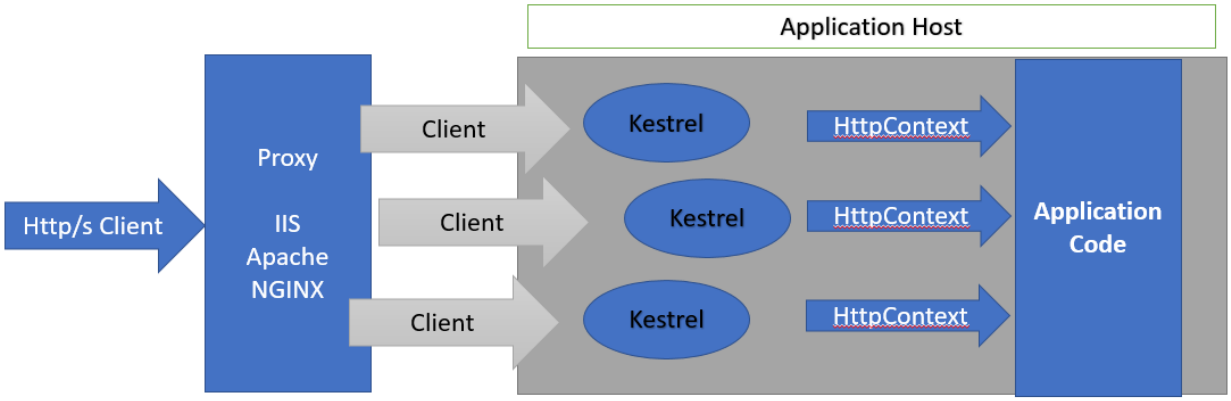
Only includes essential request-processing features — makes it fast.

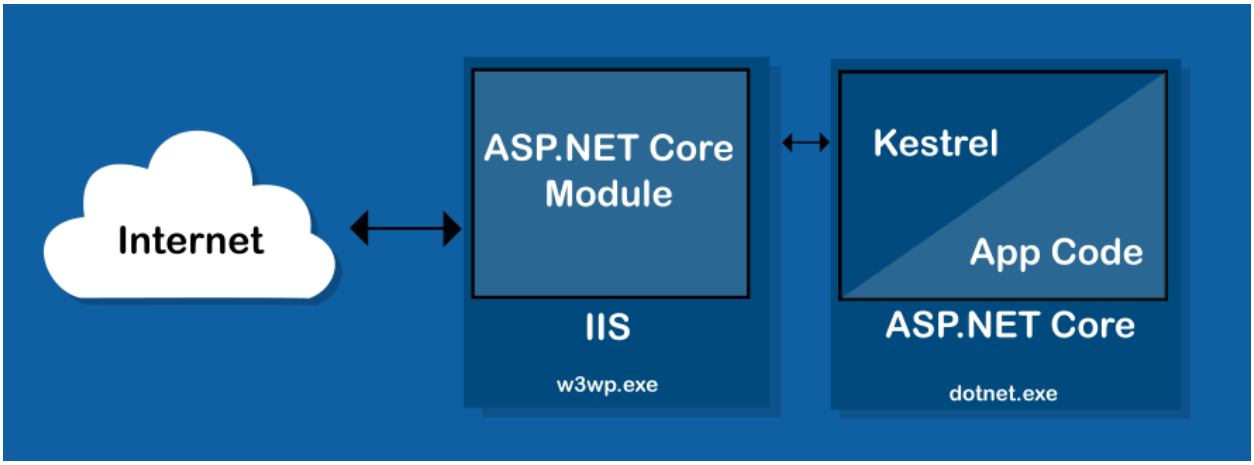
✅ Built-in Web Server

No need to install IIS for development.

⭐ 3. How Kestrel Works (High-Level Architecture)

📘 Simple Diagram





Flow:

1. Browser sends request →
2. **Kestrel** listens on a TCP port
3. Kestrel passes request into
   ASP.NET Core Middleware Pipeline
4. Controllers/Minimal APIs generate response
5. Kestrel sends response back to client

⭐ **4. Kestrel vs IIS / Nginx**

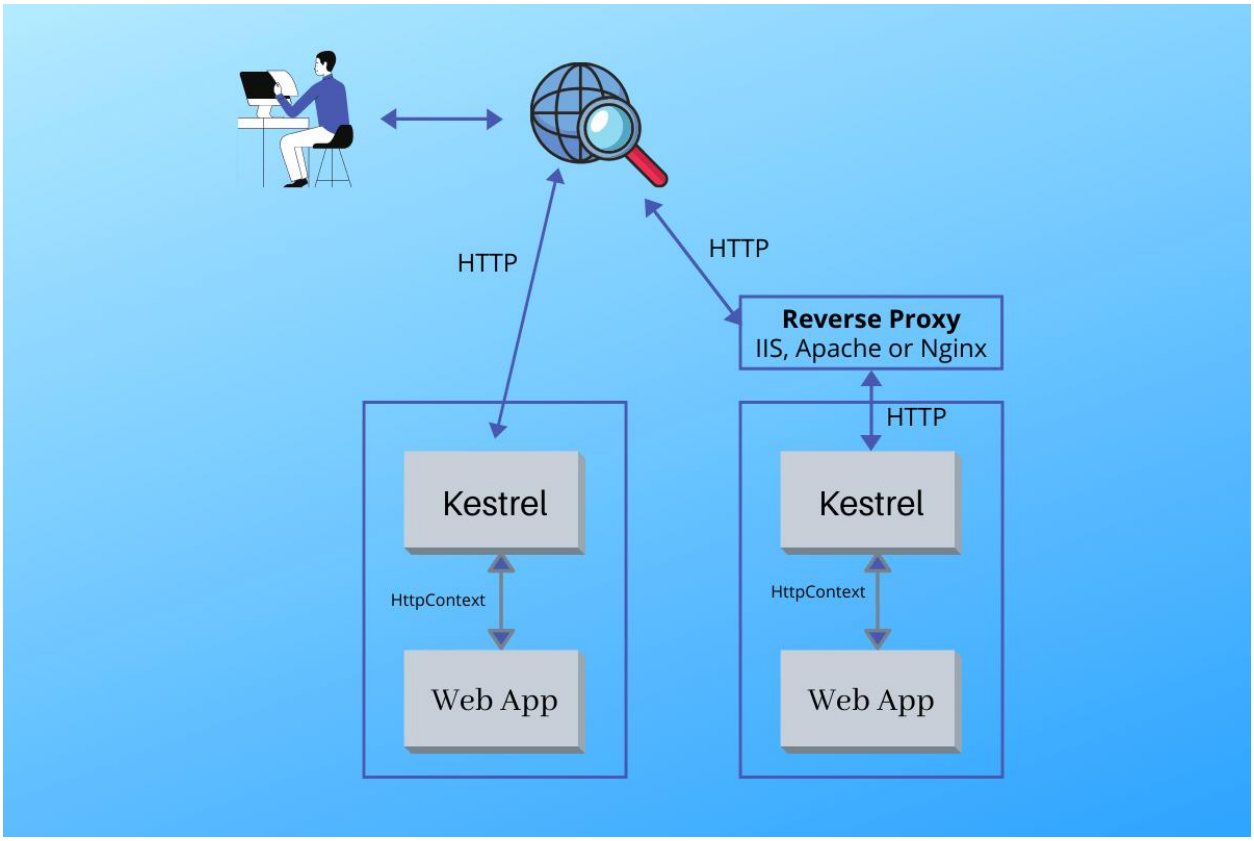
⚠️ **By default (Development)**

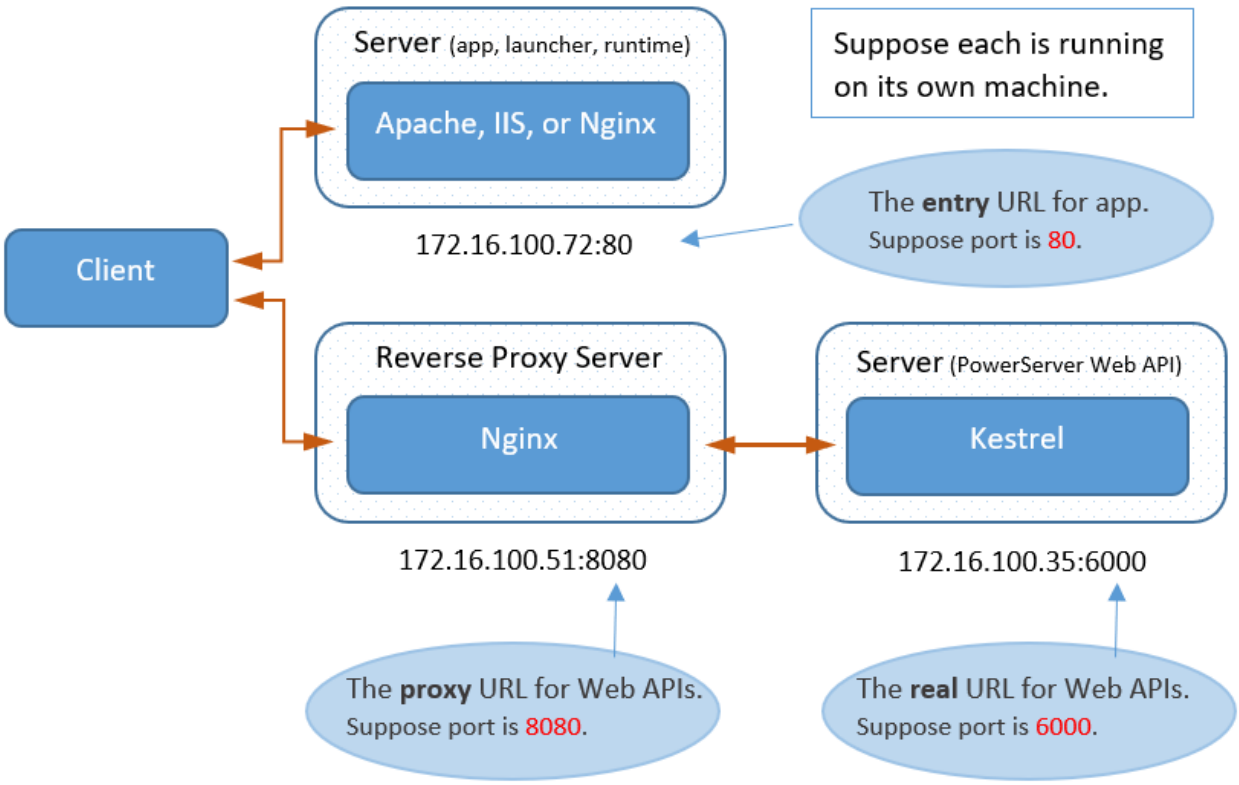
You directly use **Kestrel only**.

✔ **In Production (Recommended)**

You use:

IIS / Apache / Nginx (Reverse Proxy) → Kestrel → Application





Why Reverse Proxy?

- Security (filter attacks before reaching Kestrel)
- TLS/SSL termination
- Port forwarding
- Load balancing
- Logging
- Process management

⭐ 5. Hosting Models

1️⃣ Kestrel Only

Used for:

- Development
- Containers

- Microservices
- Cloud-native APIs
- Internal services

2️⃣ **Kestrel + Reverse Proxy (IIS / Nginx)**

Used for:

- Public-facing websites
- Enterprise apps
- Apps requiring SSL certificates on IIS/Nginx

⭐ **6. How Kestrel Starts (Program.cs)**

✓ **For ASP.NET Core 8+ (minimal hosting model)**

```
var builder = WebApplication.CreateBuilder(args);


builder.WebHost.ConfigureKestrel(options =>
{
   options.ListenAnyIP(5000); // HTTP
   options.ListenAnyIP(5001, listen => listen.UseHttps()); // HTTPS
});


var app = builder.Build();
app.MapGet("/", () => "Kestrel running!");
app.Run();
```

✓ **Default Behavior**

If no port is configured:

- HTTP → **http://localhost:5000**
- HTTPS → **https://localhost:5001**

⭐ 7. Configuring Kestrel in appsettings.json

```
"Kestrel": {
  "Endpoints": {
    "Http": {
      "Url": "http://0.0.0.0:8080"
    },
    "Https": {
      "Url": "https://0.0.0.0:8443",
      "Certificate": {
        "Path": "cert.pfx",
        "Password": "mypassword"
      }
    }
  }
}
```

Enables:

- Multiple endpoints
- Custom ports
- Certificates
- Limits

⭐ 8. Kestrel Server Limits (Very Important for Interviews)

Kestrel allows configuring:

🔹 Request Body Size

```
options.Limits.MaxRequestBodySize = 10 * 1024; // 10 KB
```

🔹 **Concurrent connections**

options.Limits.MaxConcurrentConnections = 100;

🔹 **Keep-alive timeout**

options.Limits.KeepAliveTimeout = TimeSpan.FromMinutes(2);

🔹 **Max request line length**

→ Helps protect from attacks

⭐ **9. Where is Kestrel Used?**

✓ **Azure App Service (Linux & Windows)**

✓ **Docker Containers**

✓ **Kubernetes Pods**

✓ **Microservices Architecture**

✓ **Self-hosted APIs**

✓ **Edge services (reverse proxy with YARP)**

⭐ **10. Real-World Use Case (Instructor Example)**

🛒 **E-Commerce API Using Kestrel**

- ProductService API hosted on Linux → Kestrel
- CartService API running in Docker → Kestrel
- API Gateway (Ocelot/YARP) routes the calls
- Nginx acts as reverse proxy for SSL

Perfect for **scalable microservices**.

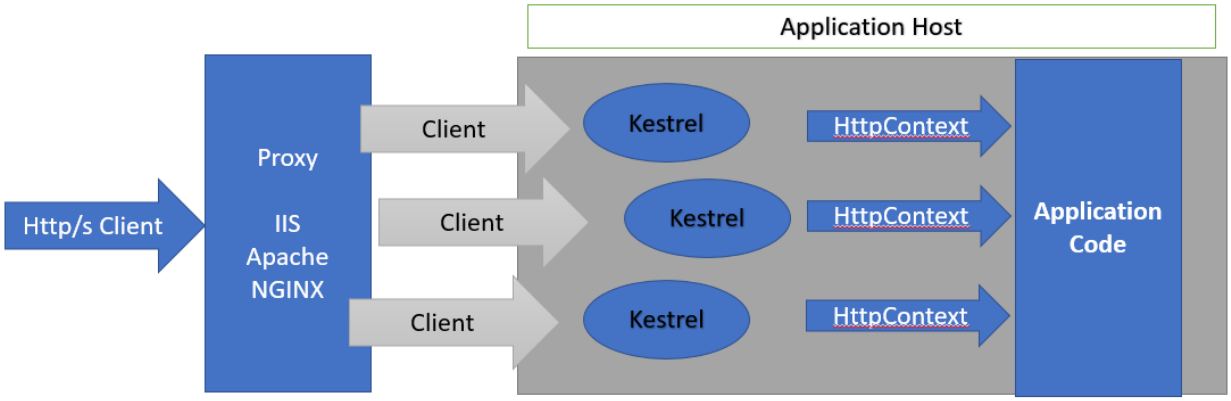
⭐ **11. Kestrel Interview Questions (For Students)**

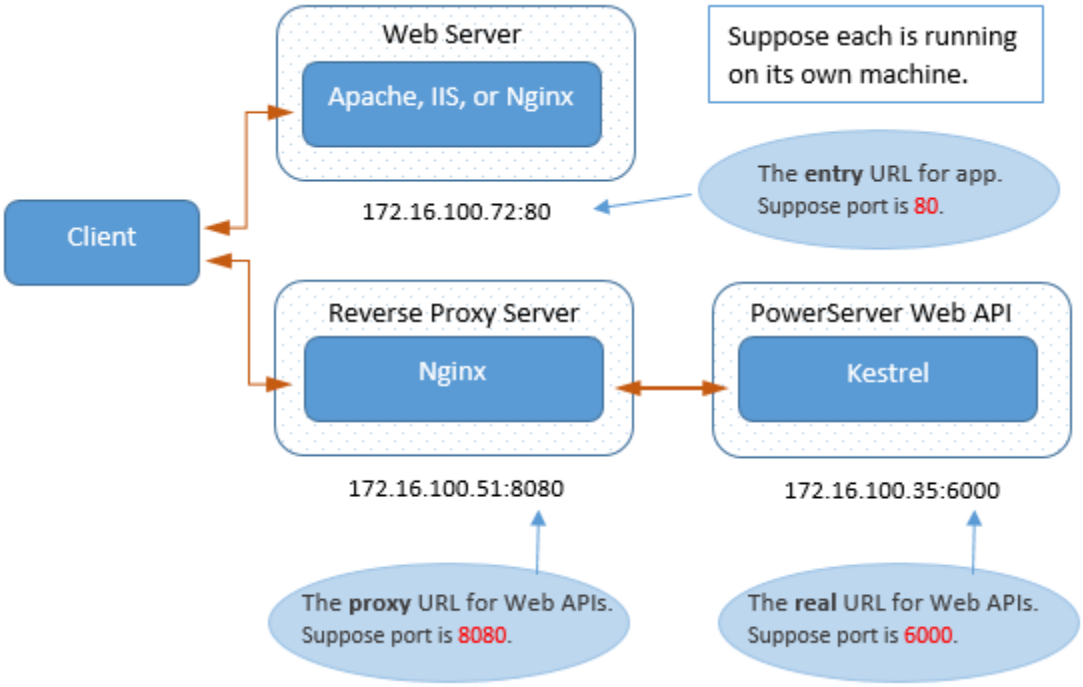
1. What is Kestrel in ASP.NET Core?

2. Why is Kestrel used instead of IIS?
3. What is the difference between Kestrel and IIS?
4. Why do we use IIS/Nginx as a reverse proxy with Kestrel?
5. How do you configure Kestrel ports?
6. How do you host ASP.NET Core in Linux using Kestrel?
7. Explain the Kestrel request processing pipeline.
8. How do you configure request limits in Kestrel?

⭐ 12. One-Line Summary

Kestrel is a high-performance, cross-platform, built-in web server for ASP.NET Core applications, used alone or behind a reverse proxy like IIS/Nginx for production.




Web Server
Apache, IIS, or Nginx
Suppose each is running on its own machine.
The entry URL for app.
Suppose port is 80.
172.16.100.72:80
Client
Reverse Proxy Server
Nginx
PowerServer Web API
Kestrel
172.16.100.51:8080
172.16.100.35:6000
The proxy URL for Web APIs.
Suppose port is 8080.
The real URL for Web APIs.
Suppose port is 6000.