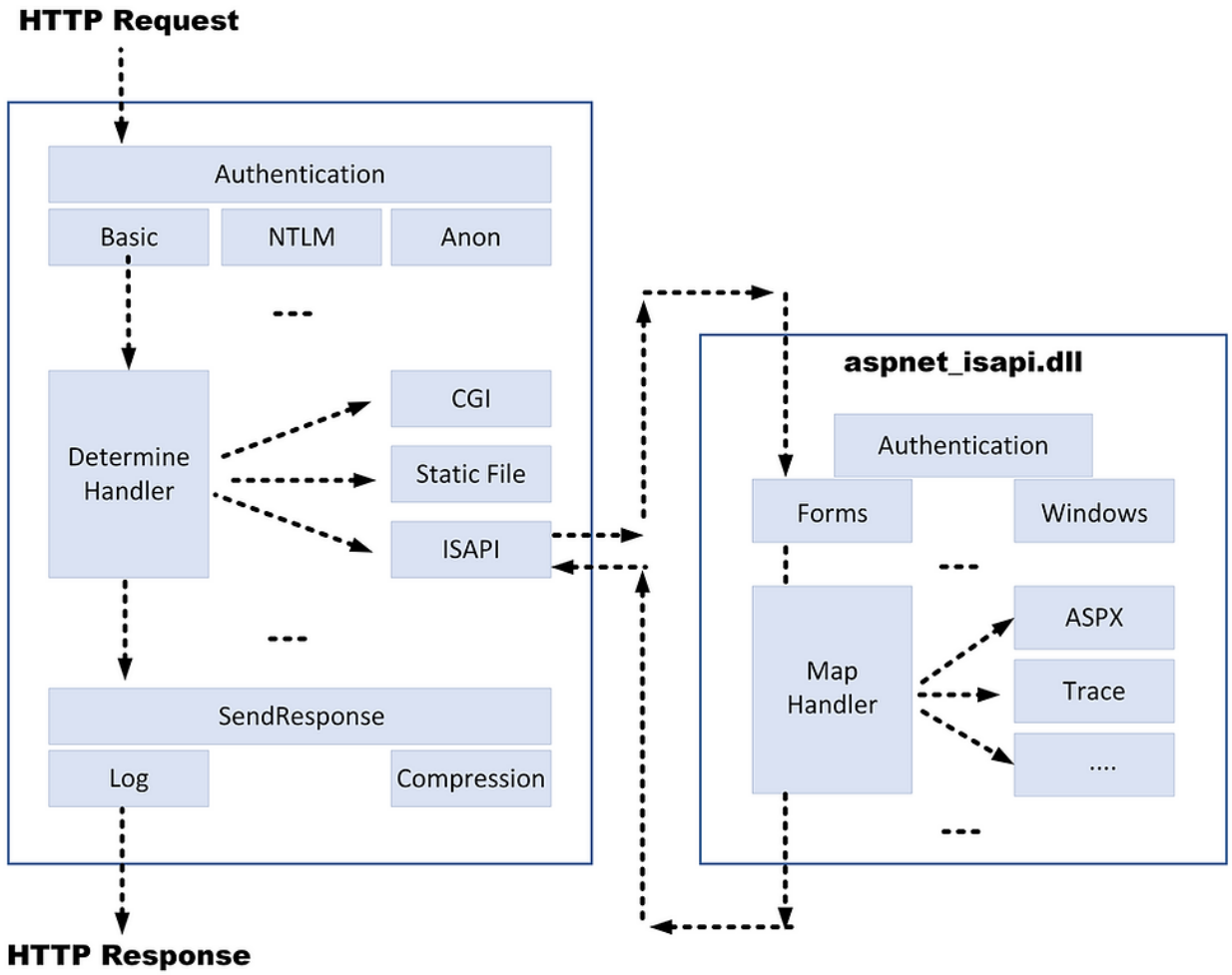
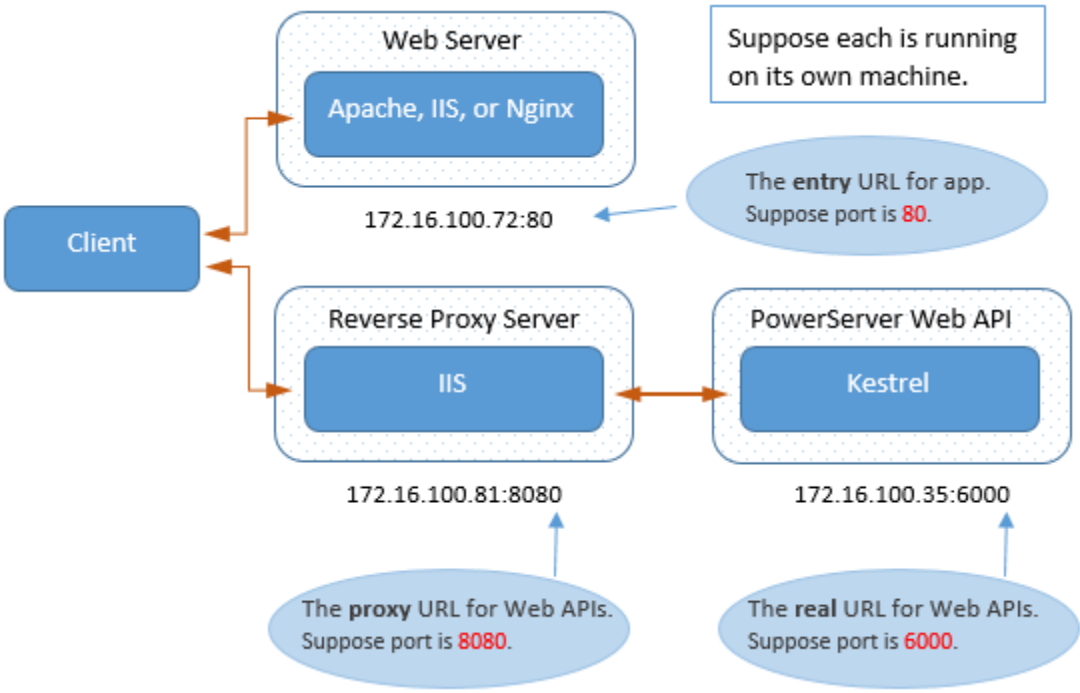
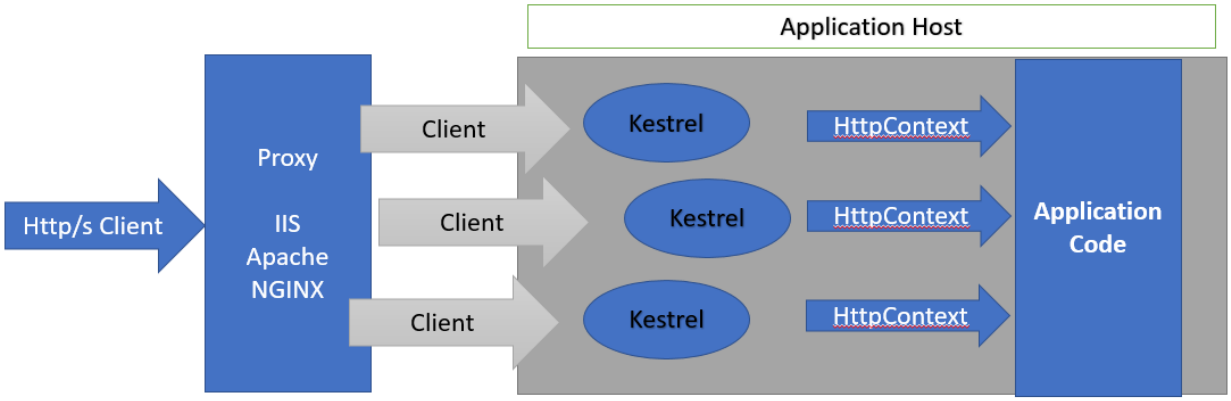
⭐ **Kestrel vs IIS — Complete Comparison**







📌 1. Quick Summary (Simple Explanation)

Feature	Kestrel	IIS
Type	Cross-platform web server	Windows-only web server
Performance	Very fast, lightweight	Slower than Kestrel but stable
OS Support	Windows, Linux, macOS	Only Windows

Feature	Kestrel	IIS
Recommended For	Development, Linux hosting, Docker, APIs, Microservices	Enterprise hosting on Windows
Reverse Proxy	Recommended behind IIS/Nginx for security	Built-in
Static Files	Must configure manually	Built-in static file hosting
Process Management	Basic	Advanced (recycle, warm-up, logs)

⭐ 2. What is Kestrel?

- The **default web server** for ASP.NET Core
- Cross-platform
- Very fast
- Suitable for microservices and cloud-native architectures
- Usually paired with Nginx/IIS for security, load balancing, SSL

⭐ 3. What is IIS?

- Full-featured **Windows Web Server**
- Handles:
    - Authentication
    - Logging
    - SSL termination
    - Static files
    - Application pool recycling
- Works as a **Reverse Proxy** for ASP.NET Core apps
- Production-ready for enterprise workloads

⭐ 4. Detailed Comparison Table

Kestrel vs IIS — Feature-wise

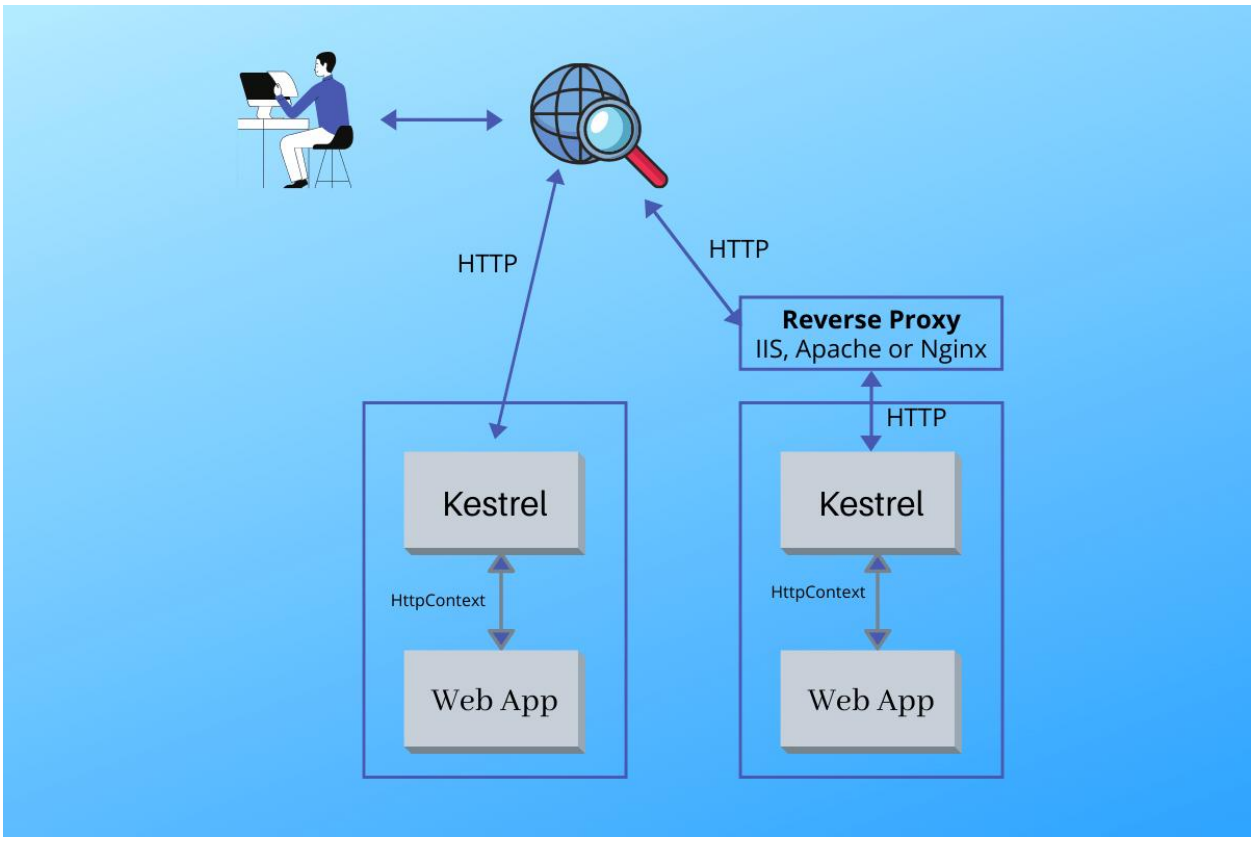
Capability	Kestrel	IIS
Platform	Cross-platform	Windows Only
Performance	Very High	Medium
Security	Limited (needs proxy for hardening)	Very high (URL filtering, request filtering, Windows Auth)
Static File Handling	Requires middleware	Built-in efficient static file server
Process Management	Basic	App pools, worker processes
SSL Handling	Supported but limited	Mature SSL certificate management
Request Filtering	Basic limits	Advanced (size limits, IP restrictions, request filtering rules)
Hosting Model	Self-hosted	Reverse proxy for ASP.NET Core
Logging	Basic	Advanced (IIS logs, Failed Request Tracing)
Port Sharing	No	Yes
Use Case	Cloud containers, Linux, microservices	Enterprise apps, Windows servers

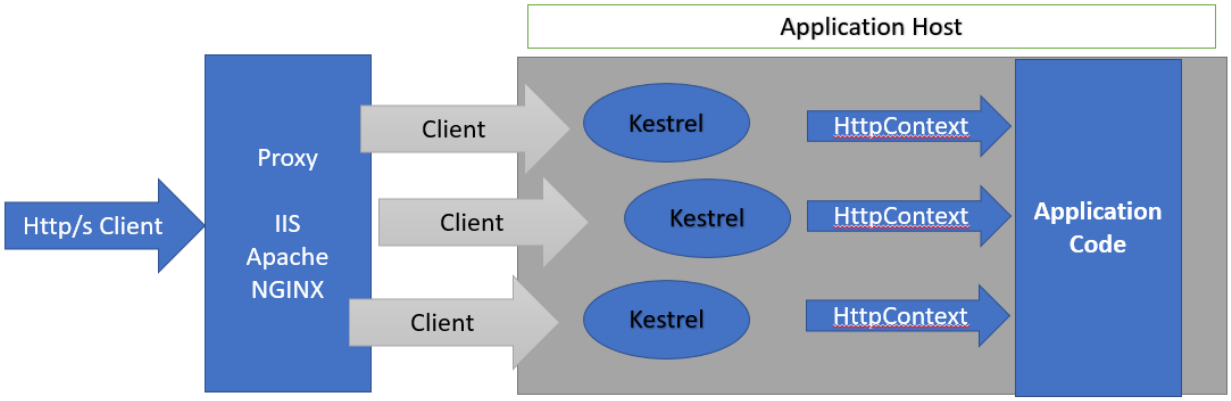
---

⭐ 5. Why Use Both Together? (Most Common Production Setup)

✓ Recommended Production Architecture

IIS (Reverse Proxy) → Kestrel → ASP.NET Core App





Why?

Because each has a different job:

Task	**Handled By**
SSL termination	IIS
Windows Authentication	IIS

Task	Handled By
Logging & monitoring	IIS
Static files	IIS
Request filtering	IIS
Business logic	ASP.NET Core app running on Kestrel

⭐ 6. Practical Example (Production Setup)

You deploy an app on Windows Server:

- IIS listens on **port 80/443**
- Acts as a **reverse proxy**
- Forwards requests to Kestrel on **port 5000**
- Kestrel runs your ASP.NET Core application
- If Kestrel crashes → IIS can restart it automatically

⭐ 7. Which One Should Students Use?

🌐 Development

- Use **Kestrel only**
- When running via Visual Studio or dotnet run

🔐 Production (Windows Server)

- Use **IIS + Kestrel**

🐧 Container / Linux / Kubernetes

- Use **Kestrel + Nginx/Apache**

⭐ 8. Performance Comparison

Scenario	Winner
Raw request throughput	**Kestrel**
Static content (images, HTML, CSS)	**IIS**
CPU optimization	**Kestrel**
Hard security	**IIS**
Reverse proxy performance	**IIS/Nginx**

⭐ 9. Interview Questions (For Students)

1. What is Kestrel?
2. What is IIS?
3. Why do we need IIS if Kestrel is so fast?
4. Can we run ASP.NET Core without IIS?
5. Why is reverse proxy required?
6. What is the recommended production architecture?
7. Why is IIS not cross-platform?
8. How does IIS monitor and restart your app?

⭐ 10. Real-World Example

E-Commerce Website Hosting

- External customers hit → **IIS**
- IIS handles caching, static files, SSL
- Forwards API requests to **Kestrel**
- Kestrel executes Product API, Cart API, Auth API
- Application scales horizontally using Load Balancer

⭐ Final Short Summary

Kestrel is a super-fast, cross-platform web server used by ASP.NET Core.
IIS is a Windows-only, feature-rich web server that acts as a reverse proxy and security layer.
Best practice: IIS/Nginx → Kestrel → Application.