Deploying Blazor applications to Production

1.Blazor .NET 10 supports **three execution models**, each with different deployment characteristics:

Model	**Runtime Location**
Blazor Server / InteractiveServer	Server
InteractiveWebAssembly	Browser + Server
Blazor WebAssembly Standalone	Browser only

---

2. Continuous Delivery (CD) Options for Blazor

Continuous Delivery ensures that **every commit can be deployed safely**.

Common CD Pipeline Stages

1. **Source Control**
    - GitHub / Azure DevOps / GitLab
2. **Build**
    - dotnet restore
    - dotnet build
3. **Test**
    - Unit tests (xUnit, bUnit)
4. **Publish**
    - dotnet publish -c Release
5. **Deploy**
    - IIS / Azure / Linux / Docker

Popular CD Tools

Tool	When to Use
GitHub Actions	Open-source or GitHub-hosted code
Azure DevOps Pipelines	Enterprise CI/CD
GitLab CI	GitLab-hosted repos
Jenkins	On-premise automation

Example: Publish Command Used in Pipelines

dotnet publish -c Release -o ./publish

This output folder is what gets deployed.

---

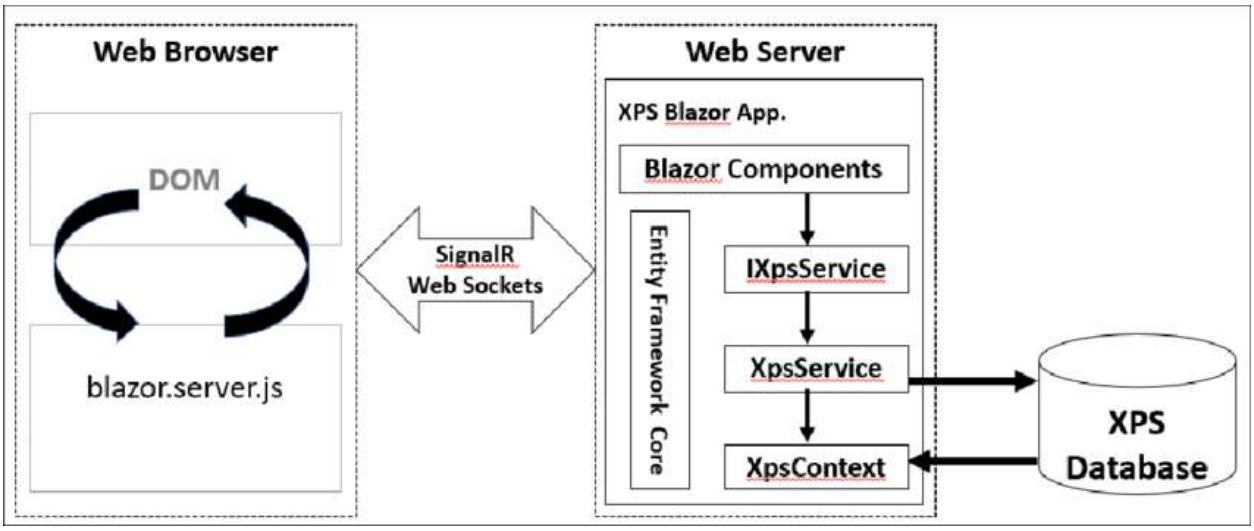
3. Hosting Options for Blazor Apps

High-Level Hosting Choices

Hosting Platform	Supported Blazor Models
IIS (Windows)	All
Azure App Service	All
Linux (Nginx + Kestrel)	Server, WASM Hosted
Docker / Kubernetes	All
Static Hosting (CDN)	WASM Standalone

---

4. Hosting Blazor Server / InteractiveServer



How It Works

- UI rendered **on the server**
- Browser communicates via **SignalR**
- Fast initial load
- Requires **persistent connection**

Deployment Characteristics

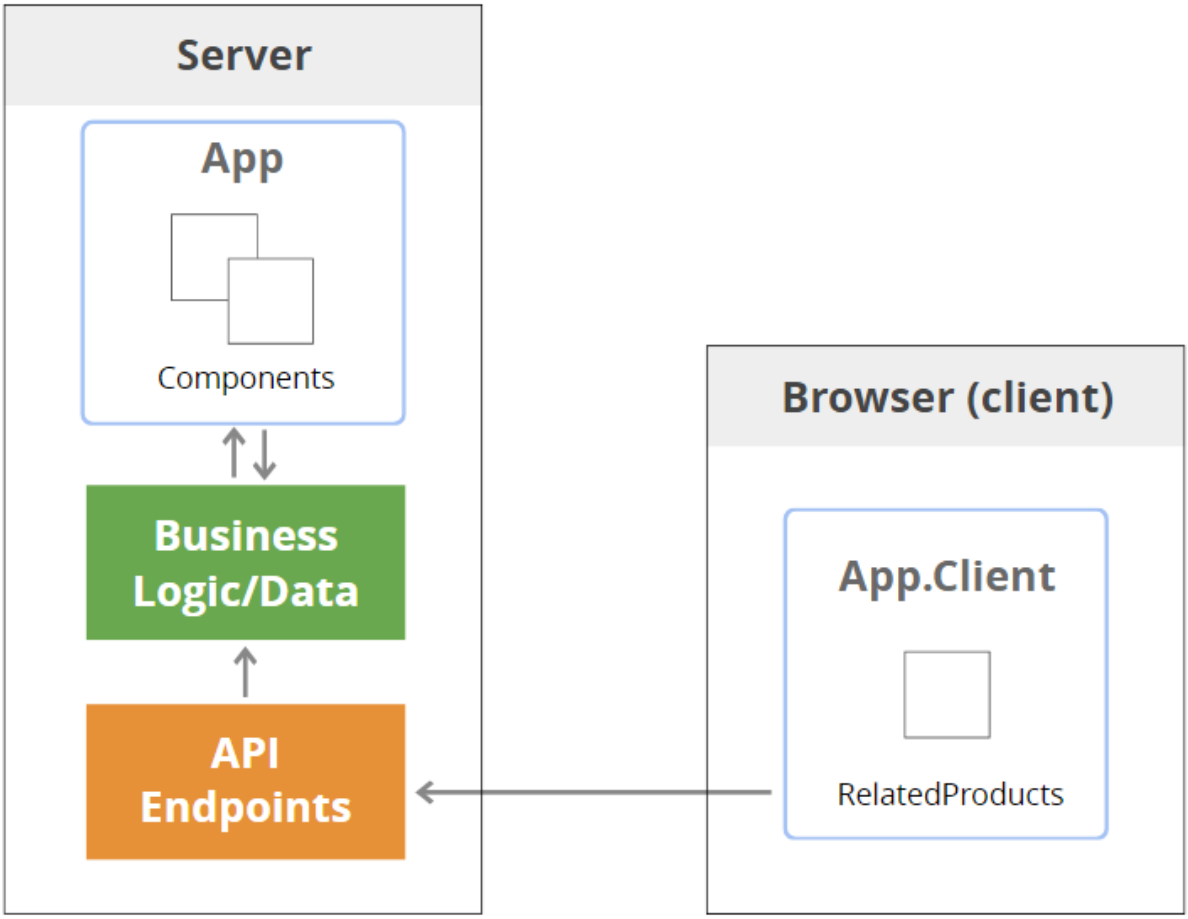
- Deployed like **ASP.NET Core MVC**
- Needs:
    - .NET Runtime on server
    - WebSockets enabled(Communication Protocol)
- Not suitable for **offline usage**

Typical Hosting Targets

- IIS
- Azure App Service
- Linux + Nginx

---

5. Hosting InteractiveWebAssembly (Blazor Web App)



How It Works

- App starts as **server-rendered**
- Later switches to **WebAssembly**
- Best balance of:
    - SEO
    - Performance
    - Interactivity

Deployment Characteristics

- Single ASP.NET Core app

- Includes:
    - Server project
    - WASM assets
- Hosted like a **normal ASP.NET Core app**

Typical Hosting Targets

- IIS
- Azure App Service
- Docker

Production Benefits

- Faster first paint
- Reduced server load after hydration
- Modern default for .NET 10

---

6. Hosting Blazor WebAssembly Standalone

How It Works

- Entire app runs in **browser**
- Server only provides static files
- Calls APIs via HTTP

Deployment Characteristics

- Output is **static files**
    - .html
    - .js
    - .wasm
- No .NET runtime needed on server

Hosting Targets

- IIS

- Azure Static Web Apps
- CDN (CloudFront, Azure CDN)
- Nginx

Production Benefits

- Scales easily
- Offline capable
- Very low hosting cost

Limitations

- Larger initial download
- SEO requires prerendering strategy
- API must be hosted separately

---

7. Hosting on IIS (Windows Server)

IIS Hosting Model

Prerequisites

- Windows Server
- IIS installed
- ASP.NET Core Hosting Bundle from **Microsoft**

Deployment Steps (Typical)

1. Publish the app:

dotnet publish -c Release

2. Copy published files to IIS site folder
3. Create Application Pool:
    - No Managed Code

- Integrated pipeline

4. Configure web.config (auto-generated)

IIS Hosting Support Matrix

Blazor Type	**IIS Support**
Blazor Server	Yes
InteractiveServer	Yes
InteractiveWebAssembly	Yes
WASM Standalone	Yes (static files)

---

8. Environment Configuration in Production

Use **environment-based configuration**:

appsettings.Production.json

builder.Configuration

.AddJsonFile("appsettings.Production.json");

Set environment variable:

ASPNETCORE_ENVIRONMENT=Production

---

9. Choosing the Right Deployment Model

Scenario	**Recommended Model**
Internal enterprise app	Blazor Server
Public SEO-friendly app	InteractiveWebAssembly
SPA / PWA	WASM Standalone
Low-cost hosting	WASM Standalone

Scenario	Recommended Model
Real-time dashboards	Server

---

10. Summary

- **Continuous Delivery** automates build, test, and deployment
- **Hosting choice depends on Blazor execution model**
- **IIS remains a first-class production host**
- **InteractiveWebAssembly is the preferred modern default in .NET 10**