EVOLUCIÓN DE .NET FRAMEWORK 4.5 → 4.8.1 Y TRANSICIÓN A .NET CORE / .NET MODERNO

Incluye explicaciones de Roslyn, JIT, PGO dinámico, Native AOT, GC avanzado, LOH compacting y RefreshMemoryLimit.

---

1. EVOLUCIÓN .NET FRAMEWORK 4.5 → 4.8.1

.NET Framework 4.5 (2012)

- async/await (C# 5).

- Mejoras en ThreadPool.

- HttpClient moderno.

- GC background mejorado.

.NET Framework 4.5.1 / 4.5.2

- Mejoras en edición continua y rendimiento ASP.NET.

- Mejoras de DPI y excepciones.

.NET Framework 4.6

- Introducción de RyuJIT 64-bit.

- Soporte TLS 1.2.

- Soporte C# 6 (Roslyn).

.NET Framework 4.6.1 / 4.6.2

- Mejoras en criptografía, X509, Always Encrypted.

- Mejoras en compresión.

.NET Framework 4.7 / 4.7.1 / 4.7.2

- DPI alto mejorado.

- Compatibilidad con .NET Standard 2.0.

- GC mejorado.

.NET Framework 4.8 / 4.8.1

- Mejoras en JIT RyuJIT.

- LOH compacting mejorado.

- Accesibilidad y ARM64.

---

2. TRANSICIÓN A .NET CORE Y .NET MODERNO

.NET Core 1.0–2.0

- Reescritura total del runtime (CoreCLR).

- Multiplataforma.

- Kestrel como servidor web.

- Inicio del uso extensivo de RyuJIT.

.NET Core 3.0–3.1

- C# 8.

- Mejora masiva en System.Text.Json.

- gRPC, Windows Desktop.

- Tiered Compilation.

.NET 5–6–7–8

- Unificación de runtimes.

- Aceleración de rendimiento.

- Dynamic PGO.

- Native AOT.

- Mejor GC: LOH compacting, RefreshMemoryLimit.

- Minimal APIs, Hot Reload.

- Blazor moderno.

---

3. GLOSARIO DE INTERNALS IMPORTANTES

¿Qué es Roslyn?

- Es el compilador moderno de C# y VB.NET.

- Totalmente escrito en .NET.

- Permite análisis del código, code generation, linters y tooling avanzado.

- Desacopló la evolución del lenguaje de la versión del framework.

¿Qué es el JIT?

- Just-In-Time Compiler.

- Convierte IL (Intermediate Language) en código máquina en tiempo de ejecución.

- Versiones:

* Legacy JIT

* RyuJIT (moderno, rápido, soporta optimizaciones avanzadas)

- Beneficios: optimización según arquitectura del CPU.

¿Qué es Tiered JIT y Dynamic PGO?

- Tiered JIT: primero compila rápido, luego optimiza partes calientes.

- PGO Dinámico: el runtime recolecta perfiles de uso y recompila código con optimizaciones basadas en ejecución real.

- Resultado: rendimiento +20–50% en escenarios reales.

¿Qué es Native AOT?

- Compilación Ahead-of-Time.

- Publica ejecutables nativos sin necesidad del JIT.

- Beneficios: arranque casi instantáneo, menor memoria, ideal para microservicios y contenedores.

¿Qué es el GC avanzado en .NET moderno?

- GC generacional mejorado.

- Mejor compactación, especialmente en Large Object Heap.

- Reducción de pausas.

- Algoritmos más eficientes para cargas paralelas.

¿Qué es el LOH Compacting?

- LOH = Large Object Heap.

- Objetos > 85 KB.

- Antes no se compactaban (fragmentación alta).

- .NET moderno puede compactar LOH y reducir la fragmentación.

¿Qué es GC.RefreshMemoryLimit()?

- Nueva API para contenedores en .NET 8.

- Ajusta dinámicamente el límite de memoria del proceso.

- Evita OOM después de cambios de cuota en Docker/Kubernetes.

---

4. RESUMEN DE LOS CAMBIOS FUNDAMENTALES ENTRE .NET FRAMEWORK Y .NET CORE/MODERNO

---

1. Rendimiento:

- RyuJIT + PGO Dinámico + AOT.

- Mucho menor uso de memoria.

- Mejor throughput.

2. Multiplataforma:

- Linux, Mac, Windows.

- Framework quedó limitado a Windows.

3. GC avanzado:

- Mejor compactación.

- LOH compacting.

- Ajuste dinámico de memoria para contenedores.

4. Sistema de compilación:

- Roslyn permite evolución rápida del lenguaje.

- C# 7–12 solo soportado en .NET Core/5/6/7/8.

5. Contenedores y Cloud-Native:

- Tamaño reducido.

- Ejecutables single-file.

- AOT.

6. API moderna:

- System.Text.Json rápido.

- Minimal APIs.

- Blazor.

- HttpClientFactory, Polly.