12 Secrets Senior developers use to increase performance of Web APIs

That middles and juniors don't know

Anton Martyniuk
antondevtips.com

01 Prefer asynchronous calls with async / await

- Async/await allows your app to handle more requests simultaneously by freeing threads during I/O operations.
- Use it for database calls, file access, and external API requests.

```
public class PaymentController : ControllerBase
{
    private readonly IPaymentService _paymentService;

    [HttpPost("process")]
    public async Task<IActionResult> ProcessPayment(PaymentRequest request,
        CancellationToken cancellationToken)
    {
        var user = await _paymentService.GetUserAsync(request.UserId,
            cancellationToken);

        var validation = await _paymentService.ValidatePaymentMethodAsync(
            request.CardToken, cancellationToken);

        if (!validation.IsValid)
            return BadRequest("Invalid payment method");

        var charge = await _paymentService.ChargeAsync(request.Amount,
            request.CardToken, cancellationToken);

        return Ok(new { TransactionId = charge.Id, Status = charge.Status });
    }
}
```

Anton Martyniuk
antondevtips.com

02

Improve Database Queries

- Add proper database indexes, optimize joins, and in EF Core use AsNoTracking() for read-only queries.

```
// Read-only query with AsNoTracking
public async Task<List<User>> GetUsersAsync()
{
    return await _dbContext.Users
        .AsNoTracking()
        .Where(u => u.IsActive)
        .ToListAsync();
}
```

Anton Martyniuk
antondevtips.com

03

Reduce Data Transfer with projections, Pagination and Filtering

- Use projections to map only required fields, and always add pagination for lists.
- Combine filtering and sorting to select only the needed data

```
[HttpGet("users")]
public async Task<IActionResult> GetUsersAsync(int pageNumber = 1, int pageSize = 10)
{
    var query = _dbContext.Users.AsNoTracking().Where(u => u.IsActive);

    var users = await query
        .Skip((pageNumber - 1) * pageSize)
        .Take(pageSize)
        .Select(u => new UserDto
        {
            Id = u.Id,
            Name = u.Name
        })
        .ToListAsync();

    return Ok(users);
}
```

Anton Martyniuk
antondevtips.com

04 Minimize JSON serialization Overhead

- Use System.Text.Json — it's faster and built-in.
- Configure it to ignore nulls and apply camelCase naming.

```
// Use System.Text.Json for faster serialization
// and configure options to optimize performance.
builder.Services
    .AddControllers()
    .AddJsonOptions(options ⇒
    {
        options.JsonSerializerOptions.PropertyNamingPolicy = JsonNamingPolicy.CamelCase;
        options.JsonSerializerOptions.IgnoreNullValues = true;
    });
```

Anton Martyniuk
antondevtips.com

05 Add Caching

Caching can reduce database load and response times dramatically.
Use:
↳ OutputCache for simple endpoint-level caching
↳ HybridCache in .NET 9+ for multi-level (memory + distributed) caching
↳ Redis for shared or distributed caching

```
builder.Services.AddOutputCache();

var app = builder.Build();
app.UseOutputCache();

[HttpGet("products")]
[OutputCache(Duration = 60)] // Cache response for 60 seconds
public async Task<IActionResult> GetProductsAsync()
{
    var products = await _productService.GetProductsAsync();
    return Ok(products);
}
```

Anton Martyniuk
antondevtips.com

05

Add Caching

Caching can reduce database load and response times dramatically.
Use:
↳ OutputCache for simple endpoint-level caching
↳ HybridCache in .NET 9+ for multi-level (memory + distributed) caching
↳ Redis for shared or distributed caching

```
builder.Services.AddHybridCache();

[HttpGet("orders/{id}")]
public async Task<IActionResult> GetOrderAsync(int id,
    IHybridCache cache)
{
    string cacheKey = $"Order_{id}";
    var order = await cache.GetOrCreateAsync(cacheKey, async entry =>
    {
        entry.AbsoluteExpirationRelativeToNow = TimeSpan.FromMinutes(10);
        using var context = new AppDbContext();
        return await context.Orders.FindAsync(id);
    });

    if (order == null)
    {
        return NotFound();
    }

    return Ok(order);
}
```

Anton Martyniuk
antondevtips.com

06 Enable Rsponse Compression

- Response compression (Gzip or Brotli) can reduce payload size by 70–90%.
- This means smaller downloads and faster responses.

```
builder.Services.AddResponseCompression(options ⇒
{
    options.EnableForHttps = true;
    options.Providers.Add<BrotliCompressionProvider>();
    options.Providers.Add<GzipCompressionProvider>();
});

builder.Services.Configure<BrotliCompressionProviderOptions>(options ⇒
{
    options.Level = System.IO.Compression.CompressionLevel.Fastest;
});

builder.Services.Configure<GzipCompressionProviderOptions>(options ⇒
{
    options.Level = System.IO.Compression.CompressionLevel.Fastest;
});

var app = builder.Build();
app.UseResponseCompression();
```

Anton Martyniuk
antondevtips.com

07 Optimize Middleware Pipeline

- Each extra middleware adds latency, so keep only what's necessary.
- Place compression before static files, authentication before authorization.

```
var app = builder.Build();

// Correct middleware ordering
app.UseResponseCompression(); // Early in the pipeline
app.UseStaticFiles();         // Serve static files

app.UseRouting();

app.UseAuthentication(); // Authentication before authorization
app.UseAuthorization();

app.UseEndpoints(endpoints =>
{
    endpoints.MapControllers();
});
```

Anton Martyniuk
antondevtips.com

08 Enable HTTP/2 and HTTP/3 Protocols

- HTTP/2 & HTTP/3 offer multiplexing, header compression, and faster handshakes.
- These protocols offer lower latency + better throughput

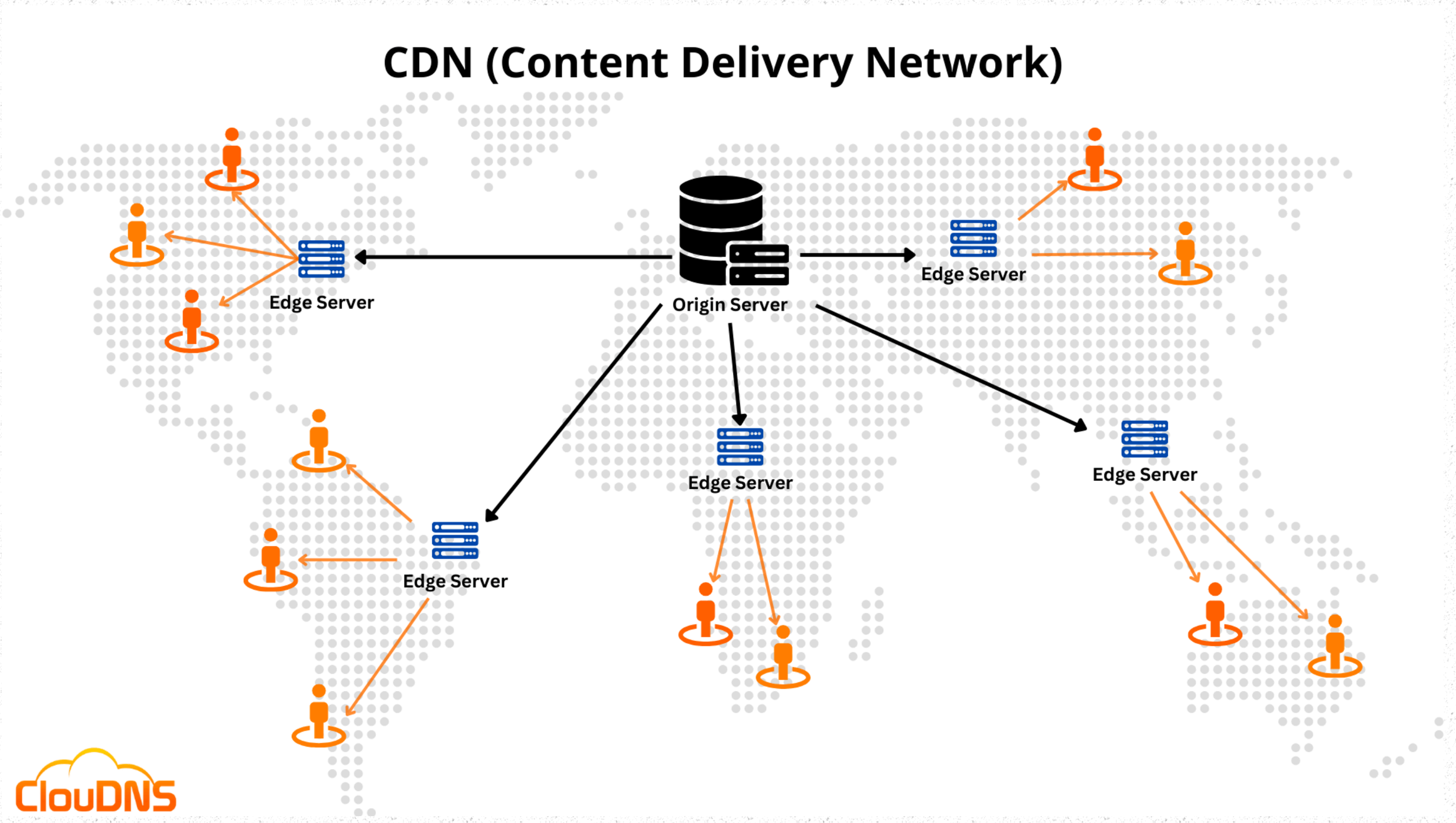
```
// In Program.cs
builder.WebHost.ConfigureKestrel(options ⇒
{
    options.ListenAnyIP(5000); // HTTP/1.1
    options.ListenAnyIP(5001, listenOptions ⇒
    {
        listenOptions.UseHttps();
        listenOptions.Protocols = HttpProtocols.Http1AndHttp2AndHttp3;
    });
});

var app = builder.Build();
// Rest of the configuration
```

Anton Martyniuk
antondevtips.com

09 Use Content Delivery Networks (CDNs)

- A CDN caches static files closer to your users.
- This reduces latency, speeds up content delivery, and offloads your server.



Anton Martyniuk
antondevtips.com

10 Implemet Rate Limiting and Throttling

- Protect your API from abuse and overload.
- Rate limiting ensures each client gets fair usage.

```
// In Program.cs
builder.Services.AddRateLimiter(options =>
{
    options.GlobalLimiter = PartitionedRateLimiter
        .Create<HttpContext, IPAddress>(context =>
    {
        var ipAddress = context.Connection.RemoteIpAddress;
        return RateLimitPartition.GetFixedWindowLimiter(ipAddress,
            _ => new FixedWindowRateLimiterOptions
        {
            PermitLimit = 100, // Allow 100 requests
            Window = TimeSpan.FromMinutes(1), // Per 1 minute window
            QueueProcessingOrder = QueueProcessingOrder.OldestFirst,
            QueueLimit = 0
        });
    });
});

var app = builder.Build();
app.UseRateLimiter();
```

Anton Martyniuk
antondevtips.com

11 Replace Controllers with Minimal APIs (.NET 9)

- Controllers are powerful but heavy.
- Minimal APIs cut pipeline overhead and can process up to 10% more requests/sec in .NET 9+.

```
var app = builder.Build();

// Minimal API endpoint
app.MapGet("/customers/{id}", async (int id) ⇒
{
    var customer = await _dbContext.Customers.FindAsync(id);

    return customer is not null
        ? Results.Ok(customer)
        : Results.NotFound();
});

app.Run();
```

Anton Martyniuk
antondevtips.com

12 Use GraphQL for Complex Data Needs

- Need multiple data queries in one call? Use GraphQL.
- Clients can fetch exactly what they need — nothing more.
- This reduces over-fetching and under-fetching problems common in REST.

```
public class Query
{
    public IQueryable<Product> GetProducts([Service] AppDbContext context)
        => context.Products;

    public Product GetProductById([Service] AppDbContext context, int id)
        => context.Products.FirstOrDefault(p => p.Id == id);
}
```

```
query {
    getProductById(id: 1) {
        id
        name
        price
    }
}
```

Anton Martyniuk
antondevtips.com

Repost & Follow Anton for more content like this

Want to improve .NET Skills?

Subscribe to my AntonDevTips newsletter