

.NET v architektuře mikroslužeb

.NET (dříve .NET Core) je v současnosti jednou z nejpreferovanějších platform pro budování **mikroslužeb** (microservices). Díky své modularitě, vysokému výkonu a multiplatformnosti umožňuje efektivní provoz v kontejnerech a cloudových prostředích.

Historie a evoluce

Cesta od monolitického frameworku k agilním mikroslužbám probíhala v několika fázích:

- **.NET Framework (2002-2019):** Původní verze pevně spjatá s Windows. Byla nevhodná pro mikroslužby kvůli velké velikosti, závislosti na OS a vysoké spotřebě paměti.
- **.NET Core 1.0 (2016):** Zásadní zlom. Microsoft představil kompletně přepsaný, open-source a multiplatformní framework. Poprvé bylo možné spouštět .NET na Linuxu v Dockeru.
- **.NET Core 3.1 (2019):** Verze s dlouhodobou podporou (LTS), která potvrdila dominanci v oblasti cloud-native aplikací.
- **Sjednocení (.NET 5, 6, 7, 8 a 9):** Od verze 5 se označení „Core“ vypouští. Moderní .NET sjednocuje vývoj pro web, desktop i mobil a přináší extrémní optimalizace výkonu.

Proč zvolit .NET pro mikroslužby?

V architektuře mikroslužeb vyniká .NET díky těmto vlastnostem:

- **Modularita:** Díky systému [NuGet](#) si každá mikroslužba stahuje pouze ty knihovny, které skutečně potřebuje.
- **Vysoký výkon:** V benchmarkcích (např. TechEmpower) patří ASP.NET Core mezi nejrychlejší webové frameworky na světě.
- **Kontejnerizace:** Obrazy pro Docker jsou optimalizované a velmi malé (zejména při použití **Alpine Linux** nebo **Native AOT**).
- **Side-by-side instalace:** Na jednom serveru může běžet více mikroslužeb, každá s jinou verzí .NET, aniž by se ovlivňovaly.

Aktuální technologie a ekosystém

Při vývoji mikroslužeb v .NET se využívají tyto systémy:

1. Komunikační protokoly

- **REST API (ASP.NET Core):** Standard pro externí komunikaci.
- **gRPC:** Preferovaná volba pro interní komunikaci mezi mikroslužbami díky binární serializaci a nízké latenci.
- **Message Brokers:** Pro asynchronní komunikaci se využívají nástroje jako **RabbitMQ**, **Apache Kafka** nebo **Azure Service Bus**.

2. Správa dat

- **Entity Framework (EF) Core:** Moderní ORM pro práci s SQL i NoSQL databázemi.
- **Dapper:** „Micro-ORM“ používané v mikroslužbách, kde je kladen absolutní důraz na rychlost SQL dotazů.

3. Orchestrace a Deployment

- **Docker:** Základní jednotka pro balení mikroslužeb.
- **Kubernetes (K8s):** Standard pro správu, škálování a dohled nad stovkami běžících kontejnerů.
- **.NET Aspire (Novinka):** Názorový (opinionated) stack od Microsoftu, který zjednodušuje lokální vývoj a propojování distribuovaných aplikací.

Výzvy a systémy dohledu

Mikroslužby přinášejí komplexitu, kterou .NET řeší pomocí:

- **Health Checks:** Integrované rozhraní pro kontrolu stavu služby (využívá Kubernetes pro restarty).
- **OpenTelemetry:** Standard pro trasování požadavků napříč desítkami služeb (Distributed Tracing).
- **Serilog / ELK Stack:** Pro centralizované logování.

— Související pojmy: [.NET Framework](#), [C#](#), [gRPC](#), [Mikroslužby](#)

From:

<http://serviceit.cz/> - IT ENCYKLOPEDIE

Permanent link:

http://serviceit.cz/doku.php?id=it:sw:dot_net_core

Last update: **2026/01/05 12:00**

