What`s new in ASP.NET Core 3?

Benjamin Abt – Principal Consultant

May 2019

devoteam | Alegri

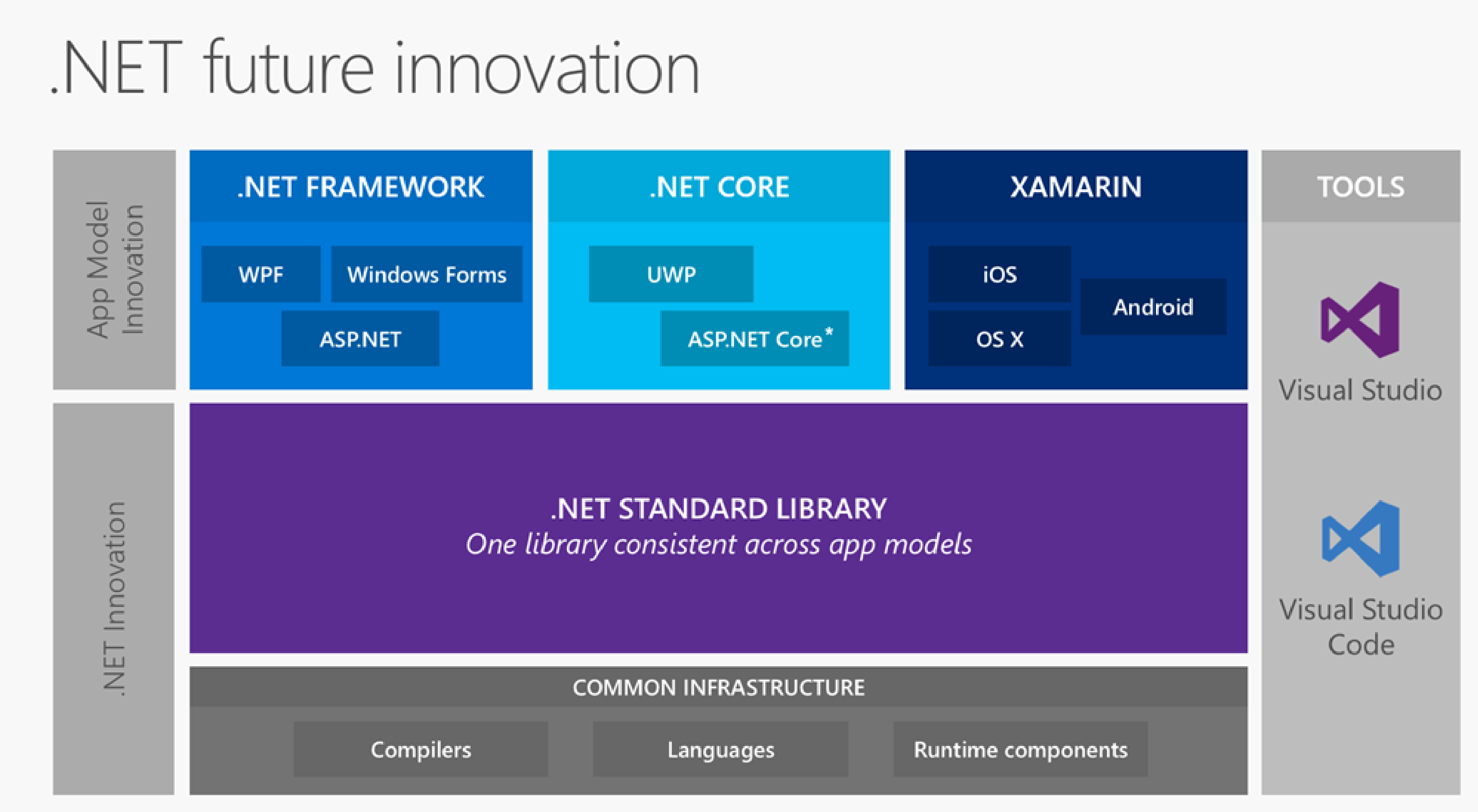
Innovative technology consulting for business

ASP.NET Core 3.0 comes with .NET 3 – incl. C# 8

- **Nullable reference types**
- **Indices, Range** and hat **(^)** operator
- New **`using`** declarations (implicit Dispose)
- New **`switch`** expressions
- Async Streams (`IAsyncEnumerable<T>`)
  - `async` **`foreach`**

more: https://docs.microsoft.com/de-de/dotnet/csharp/whats-new/csharp-8

One .NET – Unified Platform



.NET future innovation

App Model Innovation

.NET FRAMEWORK

WPF

Windows Forms

ASP.NET

.NET CORE

UWP

ASP.NET Core*

XAMARIN

iOS

Android

OS X

TOOLS

Visual Studio

Visual Studio Code

.NET Innovation

.NET STANDARD LIBRARY

One library consistent across app models

COMMON INFRASTRUCTURE

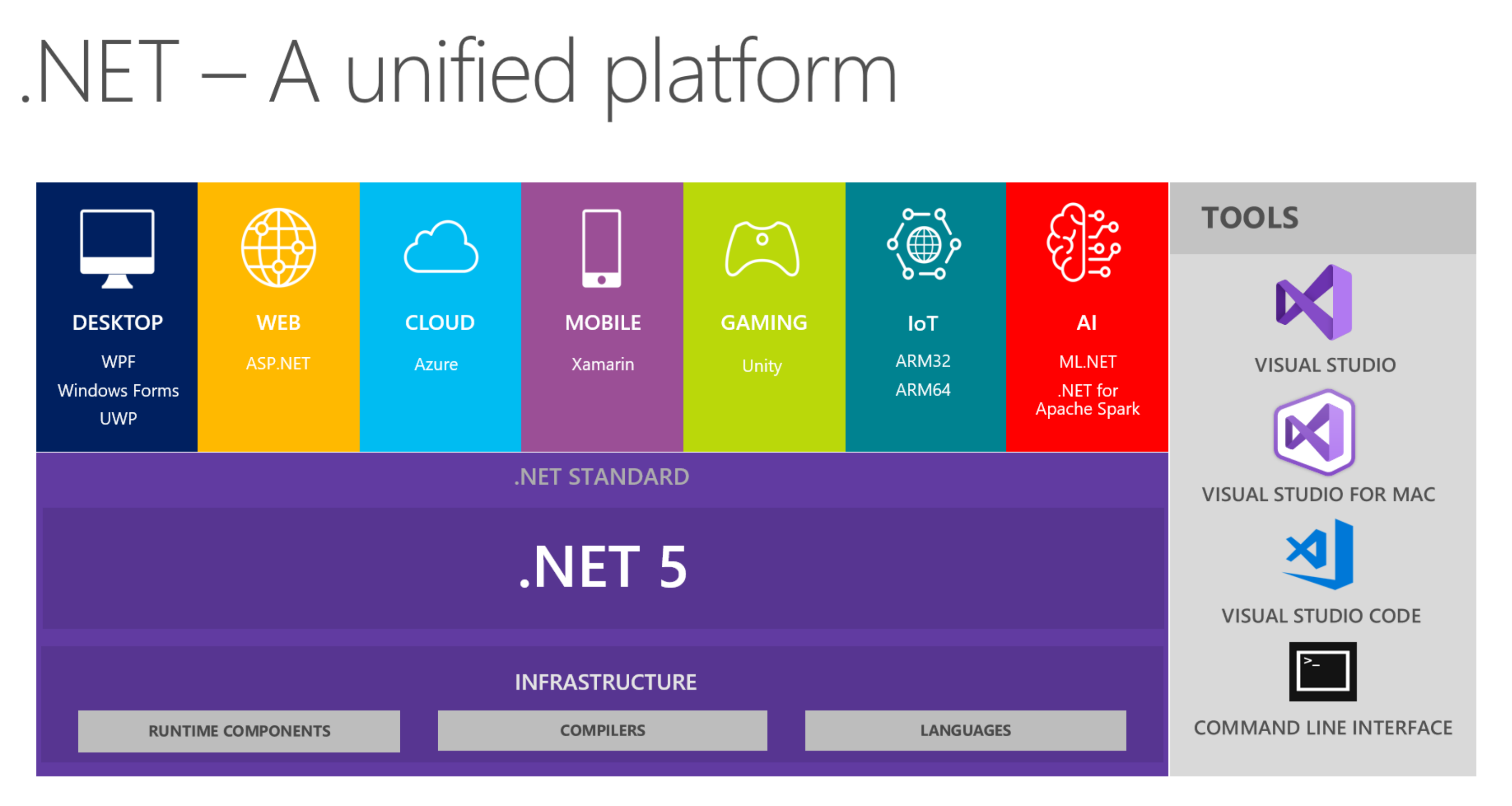
Compilers

Languages

Runtime components

Source: https://mspoweruser.com/microsoft-details-net-language-strategy/

One .NET – Unified Platform



.NET – A unified platform

DESKTOP
WPF
Windows Forms
UWP

WEB
ASP.NET

CLOUD
Azure

MOBILE
Xamarin

GAMING
Unity

IoT
ARM32
ARM64

AI
ML.NET
.NET for Apache Spark

.NET STANDARD

.NET 5

INFRASTRUCTURE

RUNTIME COMPONENTS | COMPILERS | LANGUAGES

TOOLS

VISUAL STUDIO

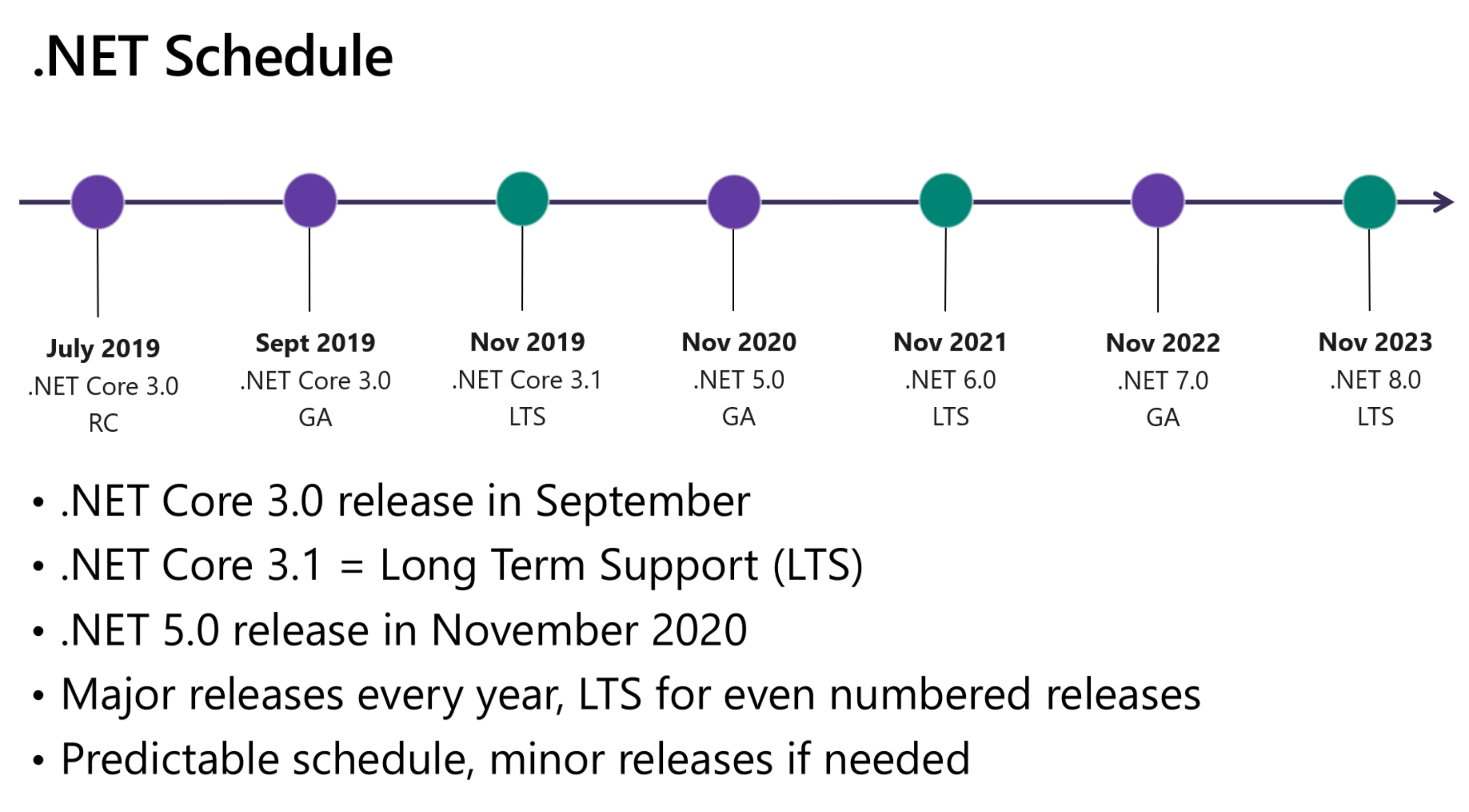
VISUAL STUDIO FOR MAC

VISUAL STUDIO CODE

COMMAND LINE INTERFACE

Source: https://devblogs.microsoft.com/dotnet/introducing-net-5/

One .NET – Schedule



.NET Schedule

July 2019
.NET Core 3.0
RC

Sept 2019
.NET Core 3.0
GA

Nov 2019
.NET Core 3.1
LTS

Nov 2020
.NET 5.0
GA

Nov 2021
.NET 6.0
LTS

Nov 2022
.NET 7.0
GA

Nov 2023
.NET 8.0
LTS

- .NET Core 3.0 release in September
- .NET Core 3.1 = Long Term Support (LTS)
- .NET 5.0 release in November 2020
- Major releases every year, LTS for even numbered releases
- Predictable schedule, minor releases if needed

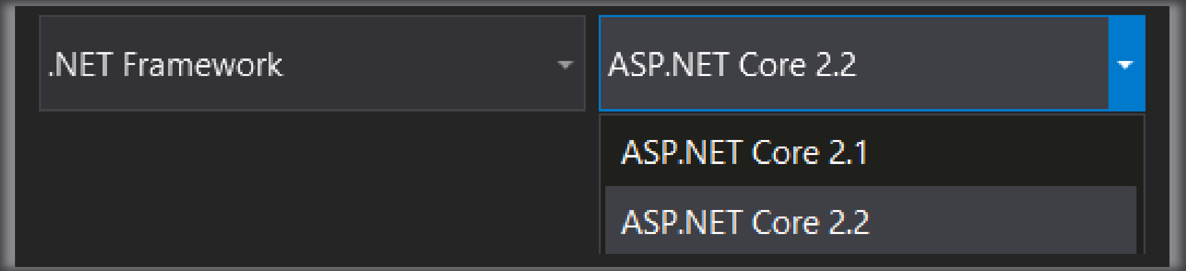
Source: https://devblogs.microsoft.com/dotnet/introducing-net-5/

ASP.NET Core 3.0

- .NET Core 3.0
- gRPC
- New Routing Story
- Performance improvements
- SignalR

- No more .NET Framework!

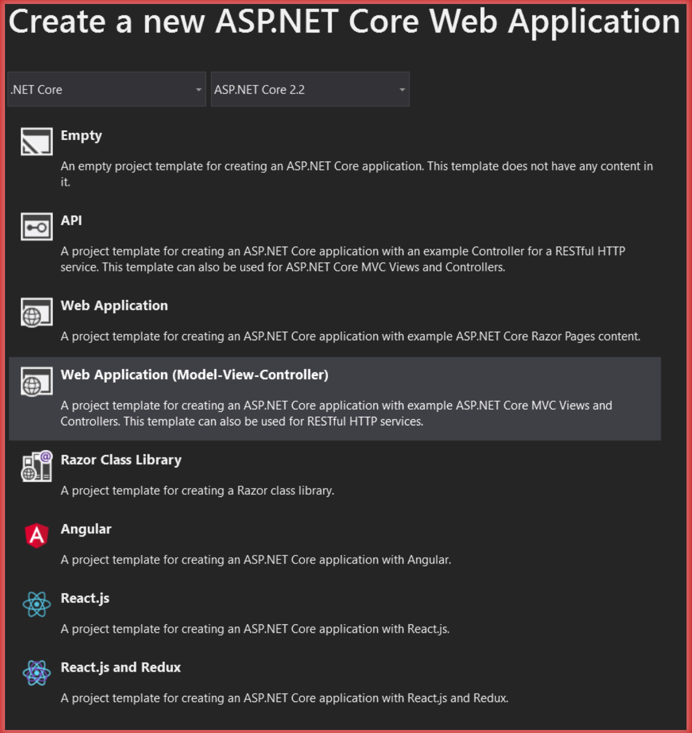
No more .NET Framework



.NET Framework

ASP.NET Core 2.2

New Scaffolding templates in Visual Studio 2019



Create a new ASP.NET Core Web Application

.NET Core | ASP.NET Core 2.2

Empty

An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

API

A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Web Application

A project template for creating an ASP.NET Core application with example ASP.NET Core Razor Pages content.

Web Application (Model-View-Controller)

A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

Razor Class Library

A project template for creating a Razor class library.

Angular

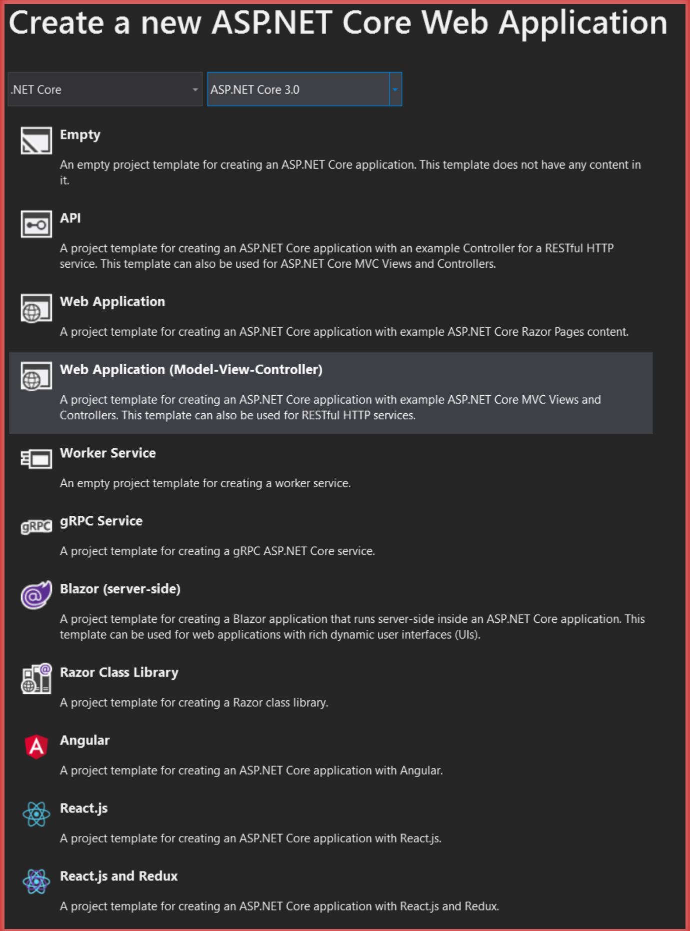
A project template for creating an ASP.NET Core application with Angular.

React.js

A project template for creating an ASP.NET Core application with React.js.

React.js and Redux

A project template for creating an ASP.NET Core application with React.js and Redux.



Create a new ASP.NET Core Web Application

.NET Core | ASP.NET Core 3.0

Empty

An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

API

A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Web Application

A project template for creating an ASP.NET Core application with example ASP.NET Core Razor Pages content.

Web Application (Model-View-Controller)

A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

Worker Service

An empty project template for creating a worker service.

gRPC Service

A project template for creating a gRPC ASP.NET Core service.

Blazor (server-side)

A project template for creating a Blazor application that runs server-side inside an ASP.NET Core application. This template can be used for web applications with rich dynamic user interfaces (UIs).

Razor Class Library

A project template for creating a Razor class library.

Angular

A project template for creating an ASP.NET Core application with Angular.

React.js

A project template for creating an ASP.NET Core application with React.js.

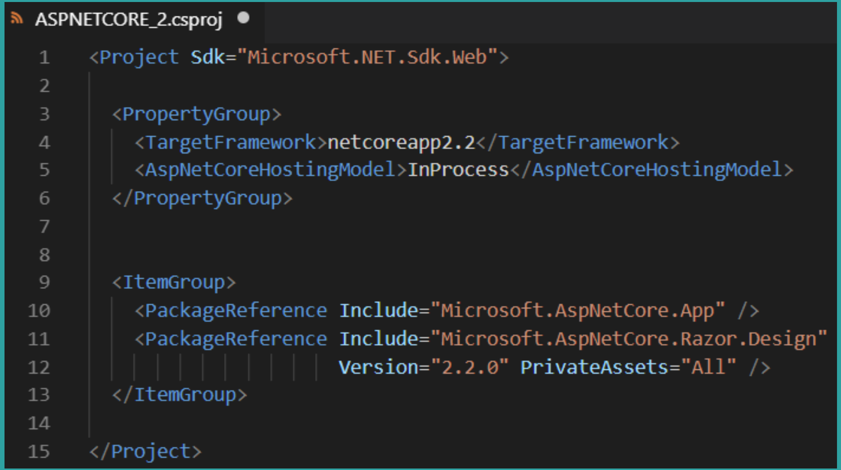
React.js and Redux

A project template for creating an ASP.NET Core application with React.js and Redux.

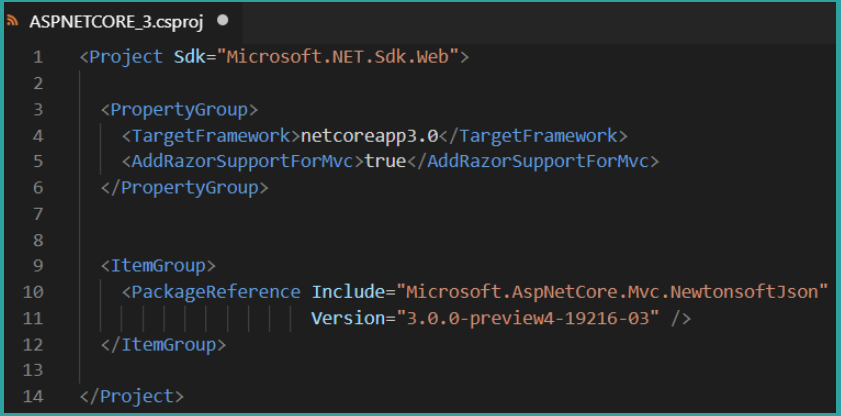
Demo

Version 2x vs 3x

First Party Assemblies only



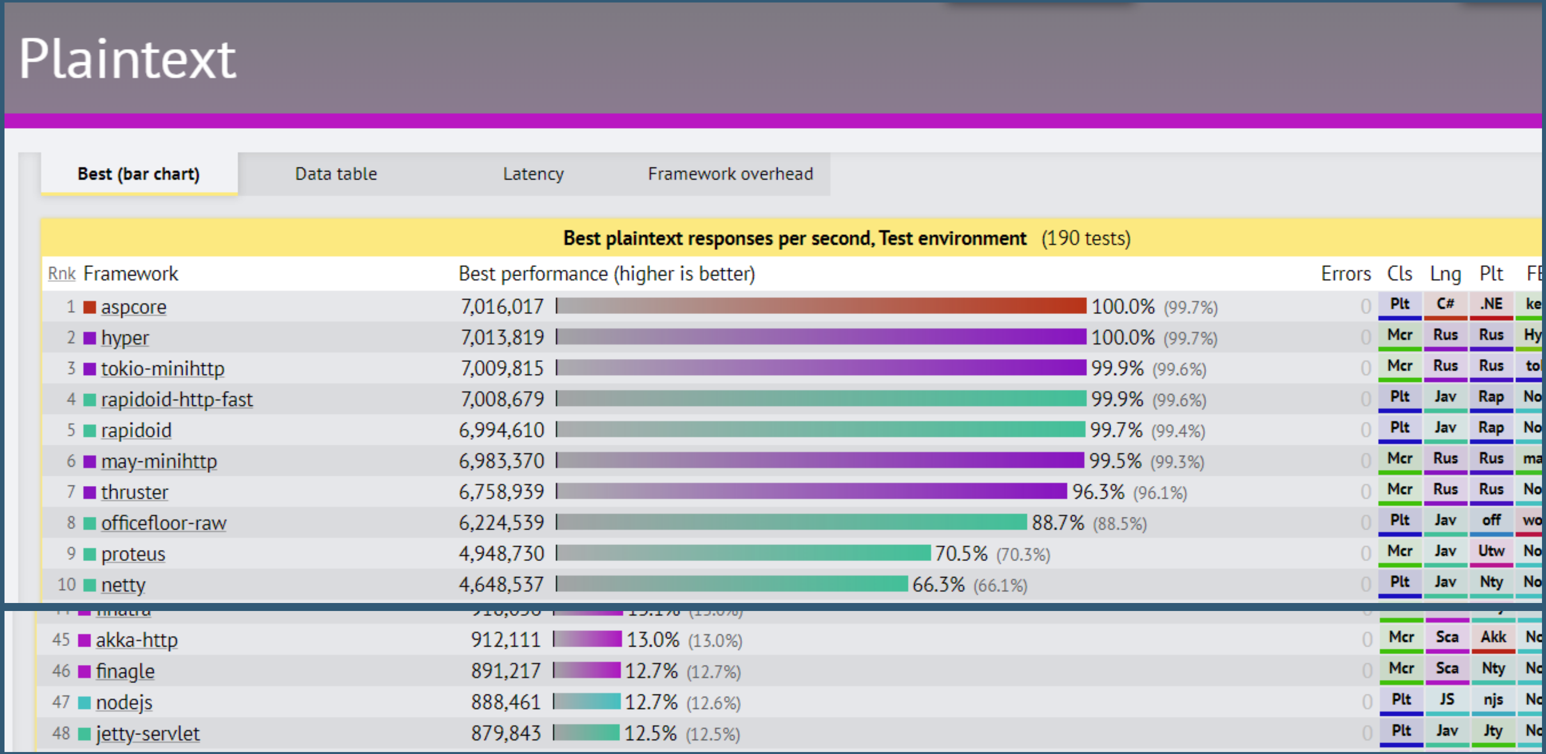
```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>netcoreapp2.2</TargetFramework>
    <AspNetCoreHostingModel>InProcess</AspNetCoreHostingModel>
  </PropertyGroup>


  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.App" />
    <PackageReference Include="Microsoft.AspNetCore.Razor.Design"
                      Version="2.2.0" PrivateAssets="All" />
  </ItemGroup>

</Project>
```



```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>netcoreapp3.0</TargetFramework>
    <AddRazorSupportForMvc>true</AddRazorSupportForMvc>
  </PropertyGroup>


  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.Mvc.NewtonsoftJson"
                      Version="3.0.0-preview4-19216-03" />
  </ItemGroup>

</Project>
```

https://www.nuget.org/packages/Microsoft.AspNetCore.App

Performance. Performance. Performance.



Plaintext

Best (bar chart) | Data table | Latency | Framework overhead

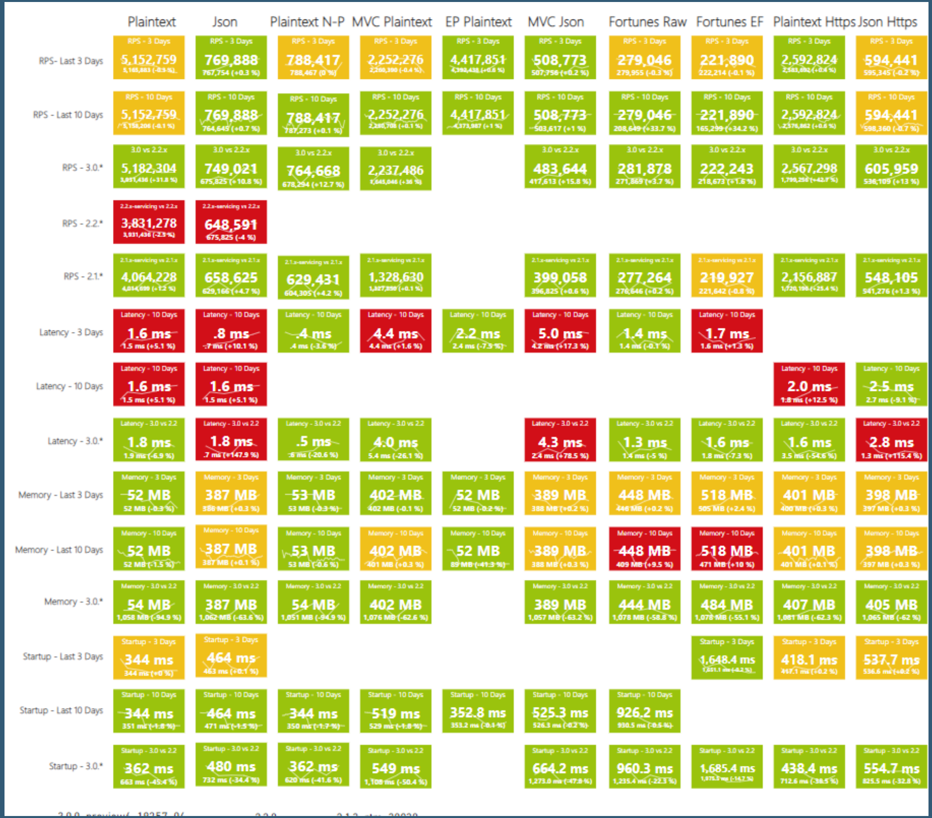
Best plaintext responses per second, Test environment (190 tests)

Rnk	Framework	Best performance (higher is better)		Errors	Cls	Lng	Plt	FE
1	🟥 aspcore	7,016,017	100.0% (99.7%)	0	Plt	C#	.NE	ke
2	🟪 hyper	7,013,819	100.0% (99.7%)	0	Mcr	Rus	Rus	Hy
3	🟪 tokio-minihttp	7,009,815	99.9% (99.6%)	0	Mcr	Rus	Rus	to
4	🟩 rapidoid-http-fast	7,008,679	99.9% (99.6%)	0	Plt	Jav	Rap	No
5	🟩 rapidoid	6,994,610	99.7% (99.4%)	0	Plt	Jav	Rap	No
6	🟪 may-minihttp	6,983,370	99.5% (99.3%)	0	Mcr	Rus	Rus	ma
7	🟪 thruster	6,758,939	96.3% (96.1%)	0	Mcr	Rus	Rus	No
8	🟩 officefloor-raw	6,224,539	88.7% (88.5%)	0	Plt	Jav	off	wo
9	🟩 proteus	4,948,730	70.5% (70.3%)	0	Mcr	Jav	Utw	No
10	🟩 netty	4,648,537	66.3% (66.1%)	0	Plt	Jav	Nty	No
45	🟪 akka-http	912,111	13.0% (13.0%)	0	Mcr	Sca	Akk	No
46	🟪 finagle	891,217	12.7% (12.7%)	0	Mcr	Sca	Nty	No
47	🟦 nodejs	888,461	12.7% (12.6%)	0	Plt	JS	njs	No
48	🟩 jetty-servlet	879,843	12.5% (12.5%)	0	Plt	Jav	Jty	No

https://www.techempower.com/benchmarks/

devoteam | Alegri

Performance. Performance. Performance.



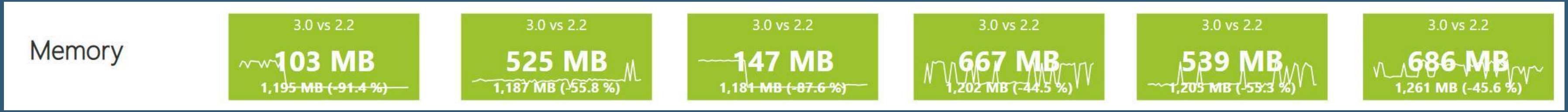
	Plaintext	Json	Plaintext N-P	MVC Plaintext	EP Plaintext	MVC Json	Fortunes Raw	Fortunes EF	Plaintext Https	Json Https
RPS- Last 3 Days	RPS - 3 Days 5,152,759 5,165,883 (-0.3 %)	RPS - 3 Days 769,888 767,754 (+0.3 %)	RPS - 3 Days 788,417 788,467 (0 %)	RPS - 3 Days 2,252,276 2,260,390 (-0.4 %)	RPS - 3 Days 4,417,851 4,392,438 (+0.6 %)	RPS - 3 Days 508,773 507,756 (+0.2 %)	RPS - 3 Days 279,046 279,955 (-0.3 %)	RPS - 3 Days 221,890 222,214 (-0.1 %)	RPS - 3 Days 2,592,824 2,583,692 (+0.4 %)	RPS - 3 Days 594,441 595,345 (-0.2 %)
RPS - Last 10 Days	RPS - 10 Days 5,152,759 5,158,206 (-0.1 %)	RPS - 10 Days 769,888 764,645 (+0.7 %)	RPS - 10 Days 788,417 787,273 (+0.1 %)	RPS - 10 Days 2,252,276 2,250,708 (+0.1 %)	RPS - 10 Days 4,417,851 4,373,987 (+1 %)	RPS - 10 Days 508,773 503,617 (+1 %)	RPS - 10 Days 279,046 208,649 (+33.7 %)	RPS - 10 Days 221,890 165,299 (+34.2 %)	RPS - 10 Days 2,592,824 2,576,862 (+0.6 %)	RPS - 10 Days 594,441 598,360 (-0.7 %)
RPS - 3.0.*	3.0 vs 2.2.x 5,182,304 3,931,436 (+31.8 %)	3.0 vs 2.2.x 749,021 675,825 (+10.8 %)	3.0 vs 2.2.x 764,668 678,294 (+12.7 %)	3.0 vs 2.2.x 2,237,486 1,645,046 (+36 %)		3.0 vs 2.2.x 483,644 417,613 (+15.8 %)	3.0 vs 2.2.x 281,878 271,869 (+3.7 %)	3.0 vs 2.2.x 222,243 218,673 (+1.6 %)	3.0 vs 2.2.x 2,567,298 1,799,256 (+42.7 %)	3.0 vs 2.2.x 605,959 536,109 (+13 %)
RPS - 2.2.*	2.2.x-servicing vs 2.2.x 3,831,278 3,931,436 (-2.5 %)	2.2.x-servicing vs 2.2.x 648,591 675,825 (-4 %)								
RPS - 2.1.*	2.1.x-servicing vs 2.1.x 4,064,228 4,014,699 (+1.2 %)	2.1.x-servicing vs 2.1.x 658,625 629,166 (+4.7 %)	2.1.x-servicing vs 2.1.x 629,431 604,305 (+4.2 %)	2.1.x-servicing vs 2.1.x 1,328,630 1,327,890 (+0.1 %)		2.1.x-servicing vs 2.1.x 399,058 396,825 (+0.6 %)	2.1.x-servicing vs 2.1.x 277,264 276,646 (+0.2 %)	2.1.x-servicing vs 2.1.x 219,927 221,642 (-0.8 %)	2.1.x-servicing vs 2.1.x 2,156,887 1,720,196 (+25.4 %)	2.1.x-servicing vs 2.1.x 548,105 541,276 (+1.3 %)
Latency - 3 Days	Latency - 10 Days 1.6 ms 1.5 ms (+5.1 %)	Latency - 10 Days .8 ms .7 ms (+10.1 %)	Latency - 10 Days .4 ms .4 ms (-3.6 %)	Latency - 10 Days 4.4 ms 4.4 ms (+1.6 %)	Latency - 10 Days 2.2 ms 2.4 ms (-7.5 %)	Latency - 10 Days 5.0 ms 4.2 ms (+17.3 %)	Latency - 10 Days 1.4 ms 1.4 ms (-0.1 %)	Latency - 10 Days 1.7 ms 1.6 ms (+1.3 %)		
Latency - 10 Days	Latency - 10 Days 1.6 ms 1.5 ms (+5.1 %)	Latency - 10 Days 1.6 ms 1.5 ms (+5.1 %)							Latency - 10 Days 2.0 ms 1.8 ms (+12.5 %)	Latency - 10 Days 2.5 ms 2.7 ms (-9.1 %)
Latency - 3.0.*	Latency - 3.0 vs 2.2 1.8 ms 1.9 ms (-6.9 %)	Latency - 3.0 vs 2.2 1.8 ms .7 ms (+147.9 %)	Latency - 3.0 vs 2.2 .5 ms .6 ms (-20.6 %)	Latency - 3.0 vs 2.2 4.0 ms 5.4 ms (-26.1 %)		Latency - 3.0 vs 2.2 4.3 ms 2.4 ms (+78.5 %)	Latency - 3.0 vs 2.2 1.3 ms 1.4 ms (-5 %)	Latency - 3.0 vs 2.2 1.6 ms 1.8 ms (-7.3 %)	Latency - 3.0 vs 2.2 1.6 ms 3.5 ms (-54.6 %)	Latency - 3.0 vs 2.2 2.8 ms 1.3 ms (+115.4 %)
Memory - Last 3 Days	Memory - 3 Days 52 MB 52 MB (-0.3 %)	Memory - 3 Days 387 MB 386 MB (+0.3 %)	Memory - 3 Days 53 MB 53 MB (-0.3 %)	Memory - 3 Days 402 MB 402 MB (-0.1 %)	Memory - 3 Days 52 MB 52 MB (-0.2 %)	Memory - 3 Days 389 MB 388 MB (+0.2 %)	Memory - 3 Days 448 MB 446 MB (+0.2 %)	Memory - 3 Days 518 MB 505 MB (+2.4 %)	Memory - 3 Days 401 MB 400 MB (+0.3 %)	Memory - 3 Days 398 MB 397 MB (+0.3 %)
Memory - Last 10 Days	Memory - 10 Days 52 MB 52 MB (-1.5 %)	Memory - 10 Days 387 MB 387 MB (+0.1 %)	Memory - 10 Days 53 MB 53 MB (-0.6 %)	Memory - 10 Days 402 MB 401 MB (+0.3 %)	Memory - 10 Days 52 MB 89 MB (-41.3 %)	Memory - 10 Days 389 MB 388 MB (+0.3 %)	Memory - 10 Days 448 MB 409 MB (+9.5 %)	Memory - 10 Days 518 MB 471 MB (+10 %)	Memory - 10 Days 401 MB 401 MB (+0.1 %)	Memory - 10 Days 398 MB 397 MB (+0.3 %)
Memory - 3.0.*	Memory - 3.0 vs 2.2 54 MB 1,058 MB (-94.9 %)	Memory - 3.0 vs 2.2 387 MB 1,062 MB (-63.6 %)	Memory - 3.0 vs 2.2 54 MB 1,051 MB (-94.9 %)	Memory - 3.0 vs 2.2 402 MB 1,076 MB (-62.6 %)		Memory - 3.0 vs 2.2 389 MB 1,057 MB (-63.2 %)	Memory - 3.0 vs 2.2 444 MB 1,078 MB (-58.8 %)	Memory - 3.0 vs 2.2 484 MB 1,078 MB (-55.1 %)	Memory - 3.0 vs 2.2 407 MB 1,081 MB (-62.3 %)	Memory - 3.0 vs 2.2 405 MB 1,065 MB (-62 %)
Startup - Last 3 Days	Startup - 3 Days 344 ms 344 ms (+0 %)	Startup - 3 Days 464 ms 463 ms (+0.1 %)						Startup - 3 Days 1,648.4 ms 1,651.1 ms (-0.2 %)	Startup - 3 Days 418.1 ms 417.1 ms (+0.2 %)	Startup - 3 Days 537.7 ms 536.6 ms (+0.2 %)
Startup - Last 10 Days	Startup - 10 Days 344 ms 351 ms (-1.8 %)	Startup - 10 Days 464 ms 471 ms (-1.5 %)	Startup - 10 Days 344 ms 350 ms (-1.7 %)	Startup - 10 Days 519 ms 529 ms (-1.8 %)	Startup - 10 Days 352.8 ms 353.2 ms (-0.1 %)	Startup - 10 Days 525.3 ms 526.3 ms (-0.2 %)	Startup - 10 Days 926.2 ms 930.5 ms (-0.5 %)			
Startup - 3.0.*	Startup - 3.0 vs 2.2 362 ms 663 ms (-45.4 %)	Startup - 3.0 vs 2.2 480 ms 732 ms (-34.4 %)	Startup - 3.0 vs 2.2 362 ms 620 ms (-41.6 %)	Startup - 3.0 vs 2.2 549 ms 1,108 ms (-50.4 %)		Startup - 3.0 vs 2.2 664.2 ms 1,273.0 ms (-47.8 %)	Startup - 3.0 vs 2.2 960.3 ms 1,235.4 ms (-22.3 %)	Startup - 3.0 vs 2.2 1,685.4 ms 1,975.5 ms (-14.7 %)	Startup - 3.0 vs 2.2 438.4 ms 712.6 ms (-38.5 %)	Startup - 3.0 vs 2.2 554.7 ms 825.5 ms (-32.8 %)

https://aka.ms/aspnet/benchmarks

devoteam | Alegri

Performance. Performance. Performance.

Glimpse



Memory

3.0 vs 2.2
103 MB
1,195 MB (-91.4 %)

3.0 vs 2.2
525 MB
1,187 MB (-55.8 %)

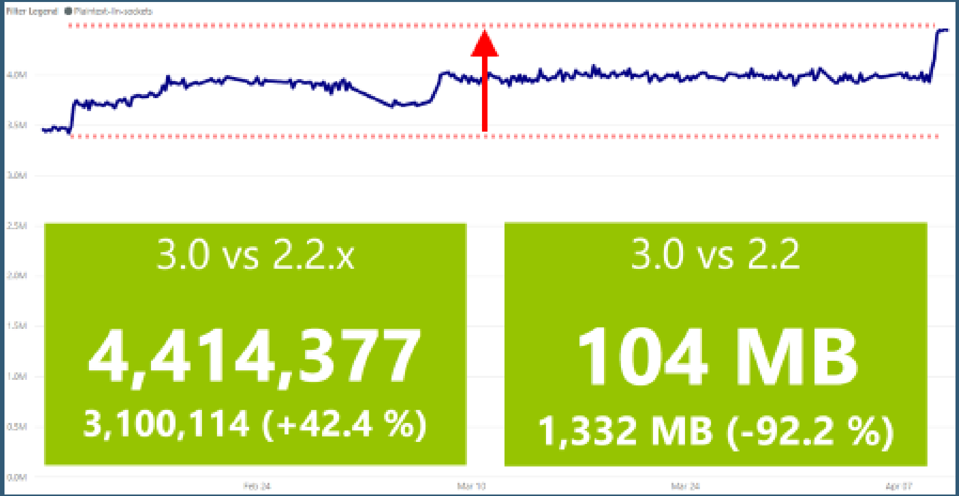
3.0 vs 2.2
147 MB
1,181 MB (-87.6 %)

3.0 vs 2.2
667 MB
1,202 MB (-44.5 %)

3.0 vs 2.2
539 MB
1,205 MB (-55.3 %)

3.0 vs 2.2
686 MB
1,261 MB (-45.6 %)

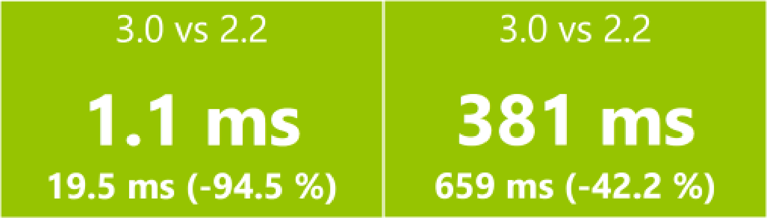
+42% throughput -92% memory



3.0 vs 2.2.x
4,414,377
3,100,114 (+42.4 %)

3.0 vs 2.2
104 MB
1,332 MB (-92.2 %)

Cold start



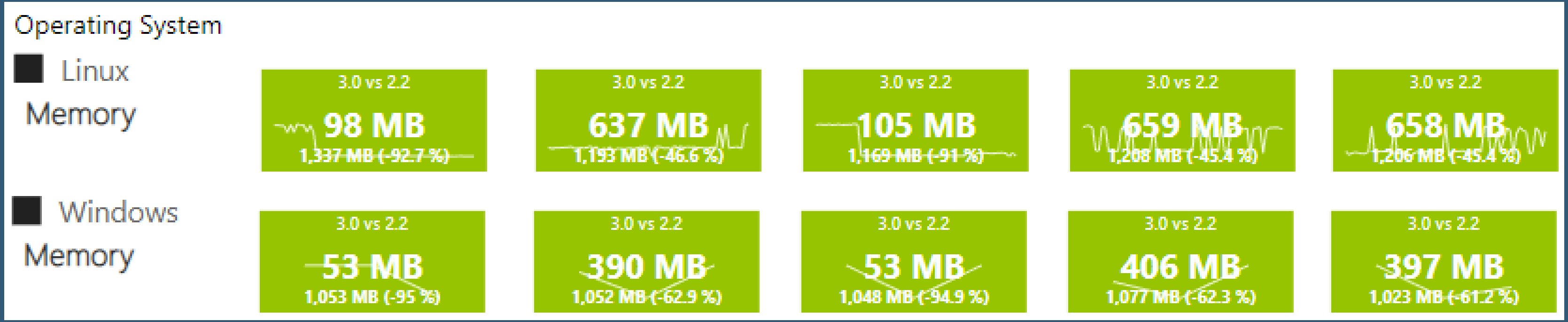
3.0 vs 2.2
1.1 ms
19.5 ms (-94.5 %)

3.0 vs 2.2
381 ms
659 ms (-42.2 %)

https://aka.ms/aspnet/benchmarks

devoteam | Alegri

Performance. Performance. Performance.



Operating System

■ Linux

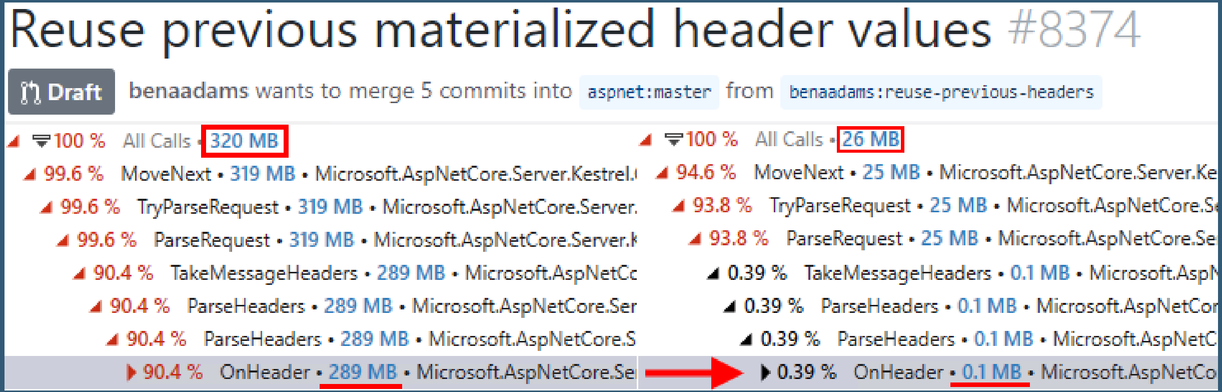
Memory

3.0 vs 2.2	3.0 vs 2.2	3.0 vs 2.2	3.0 vs 2.2	3.0 vs 2.2
98 MB	637 MB	105 MB	659 MB	658 MB
1,337 MB (-92.7 %)	1,193 MB (-46.6 %)	1,169 MB (-91 %)	1,208 MB (-45.4 %)	1,206 MB (-45.4 %)

■ Windows

Memory

3.0 vs 2.2	3.0 vs 2.2	3.0 vs 2.2	3.0 vs 2.2	3.0 vs 2.2
53 MB	390 MB	53 MB	406 MB	397 MB
1,053 MB (-95 %)	1,052 MB (-62.9 %)	1,048 MB (-94.9 %)	1,077 MB (-62.3 %)	1,023 MB (-61.2 %)



Reuse previous materialized header values #8374

Draft benaadams wants to merge 5 commits into `aspnet:master` from `benaadams:reuse-previous-headers`

```
◢ 100 %  All Calls • 320 MB
 ◢ 99.6 %  MoveNext • 319 MB • Microsoft.AspNetCore.Server.Kestrel.
  ◢ 99.6 %  TryParseRequest • 319 MB • Microsoft.AspNetCore.Server.
   ◢ 99.6 %  ParseRequest • 319 MB • Microsoft.AspNetCore.Server.K
    ◢ 90.4 %  TakeMessageHeaders • 289 MB • Microsoft.AspNetCc
     ◢ 90.4 %  ParseHeaders • 289 MB • Microsoft.AspNetCore.Ser
      ◢ 90.4 %  ParseHeaders • 289 MB • Microsoft.AspNetCore.S
       ▶ 90.4 %  OnHeader • 289 MB • Microsoft.AspNetCore.Se
```

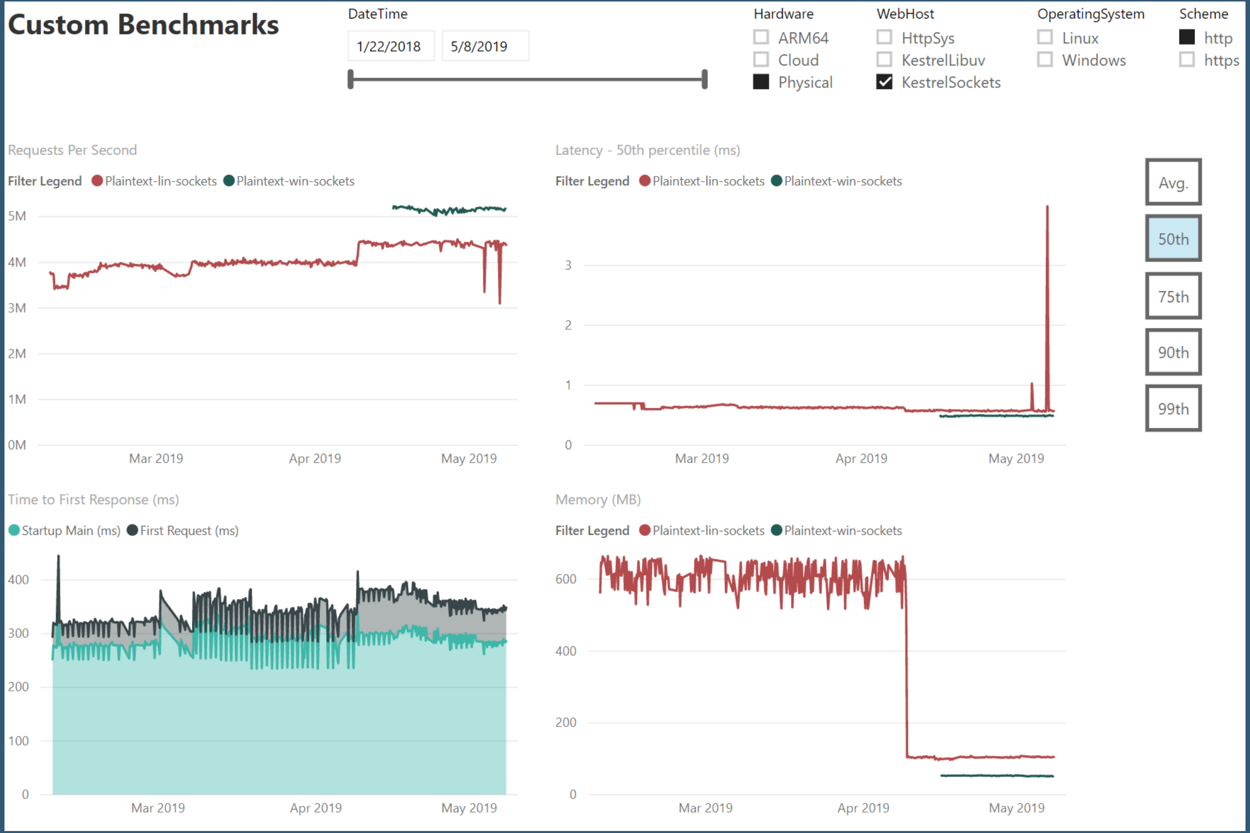
```
◢ 100 %  All Calls • 26 MB
 ◢ 94.6 %  MoveNext • 25 MB • Microsoft.AspNetCore.Server.Ke
  ◢ 93.8 %  TryParseRequest • 25 MB • Microsoft.AspNetCore.Se
   ◢ 93.8 %  ParseRequest • 25 MB • Microsoft.AspNetCore.Se
    ◢ 0.39 %  TakeMessageHeaders • 0.1 MB • Microsoft.AspN
     ◢ 0.39 %  ParseHeaders • 0.1 MB • Microsoft.AspNetCor
      ◢ 0.39 %  ParseHeaders • 0.1 MB • Microsoft.AspNetC
       ▶ 0.39 %  OnHeader • 0.1 MB • Microsoft.AspNetCo
```

https://aka.ms/aspnet/benchmarks

https://twitter.com/ben_a_adams

devoteam | Alegri

Performance. Performance. Performance.



Custom Benchmarks

DateTime

1/22/2018 5/8/2019

Hardware

- □ ARM64
- □ Cloud
- ■ Physical

WebHost

- □ HttpSys
- □ KestrelLibuv
- ☑ KestrelSockets

OperatingSystem

- □ Linux
- □ Windows

Scheme

- ■ http
- □ https

Requests Per Second
Filter Legend ● Plaintext-lin-sockets ● Plaintext-win-sockets
5M
4M
3M
2M
1M
0M
Mar 2019
Apr 2019
May 2019
Latency - 50th percentile (ms)
Filter Legend ● Plaintext-lin-sockets ● Plaintext-win-sockets
3
2
1
0
Mar 2019
Apr 2019
May 2019
Avg.
50th
75th
90th
99th
Time to First Response (ms)
● Startup Main (ms) ● First Request (ms)
400
300
200
100
0
Mar 2019
Apr 2019
May 2019
Memory (MB)
Filter Legend ● Plaintext-lin-sockets ● Plaintext-win-sockets
600
400
200
0
Mar 2019
Apr 2019
May 2019

https://aka.ms/aspnet/benchmarks

The Routing Story

- ASP.NET and ASP.NET Core 1.x/2.x is hard coupled to HTTP
- There are other protocols that have become very important or more important on the Web.

HTTP/HTTP2

AMQP/MQTT/OPC-UA

gRPC

gRPC

- gRPC = (gRPC Remote Procedure Calls)
- By Google, Open Source, Standardized, Cross-Technology, Based on HTTP/2
- Uses Protobuf = Binary
- Persistant Connections (bi-directional pooling)

gRPC

REST (HTTP)	gRPC (HTTP/2)
Json	Protobuf
Request-Response-Model	Streaming(Server/Client/Both)
Serialization	Strongly-Typed, all languages
Human readable	Not Human readable
	Faster, built-in load balancer
All Services	Internal Services (Gateway)

Blazor

- WebAssembly
- Client Side Application
- C# instead of JavaScript
- JavaScript interop!
- .NET Standard!
- Wont be shipped with .NET 3.0 (later)

https://docs.microsoft.com/en-us/aspnet/core/blazor/hosting-models?view=aspnetcore-3.0
https://marketplace.visualstudio.com/items?itemName=aspnet.blazor

Blazor

Blazor (Client Side)

Offline

Low Latency

Full Runtime ☹

(Visual Studio Extension)

Blazor (ASP.NET Core Hosted)

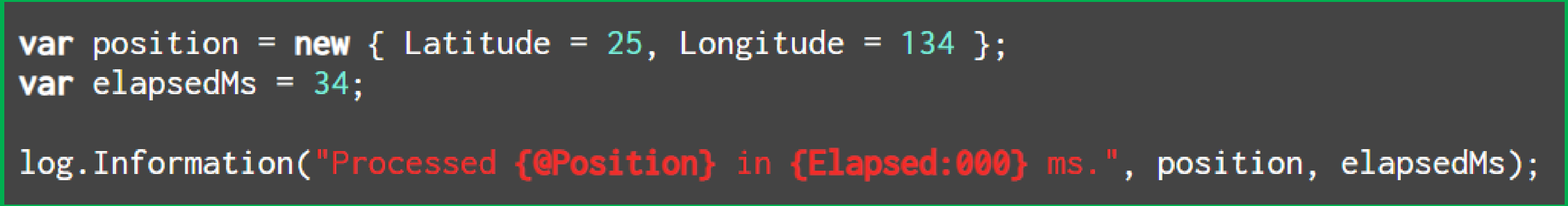
No Offline

Latency

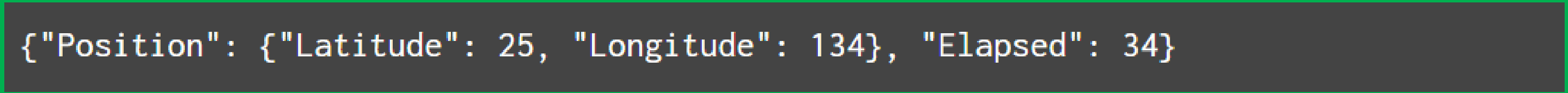
Thin Client

Simpler Architecture

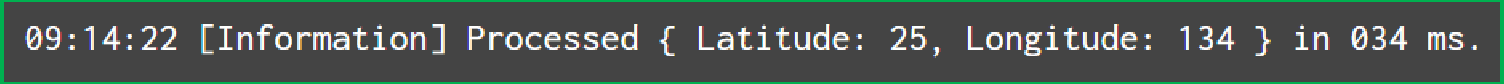
Best Practise – Full structured logging.



```
var position = new { Latitude = 25, Longitude = 134 };
var elapsedMs = 34;

log.Information("Processed {@Position} in {Elapsed:000} ms.", position, elapsedMs);
```



```
{"Position": {"Latitude": 25, "Longitude": 134}, "Elapsed": 34}
```

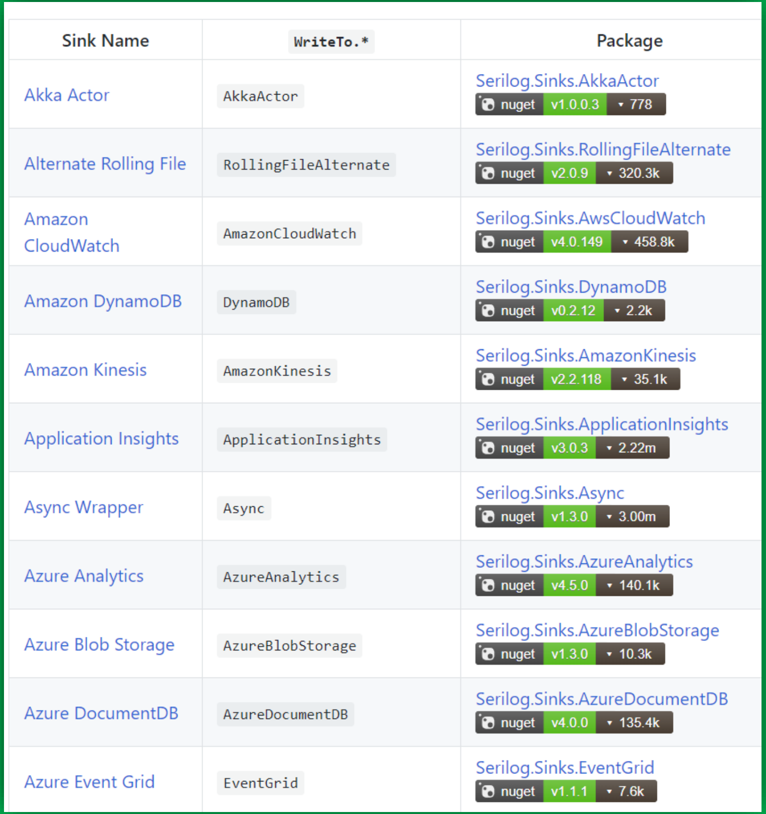


```
09:14:22 [Information] Processed { Latitude: 25, Longitude: 134 } in 034 ms.
```

serilog.net

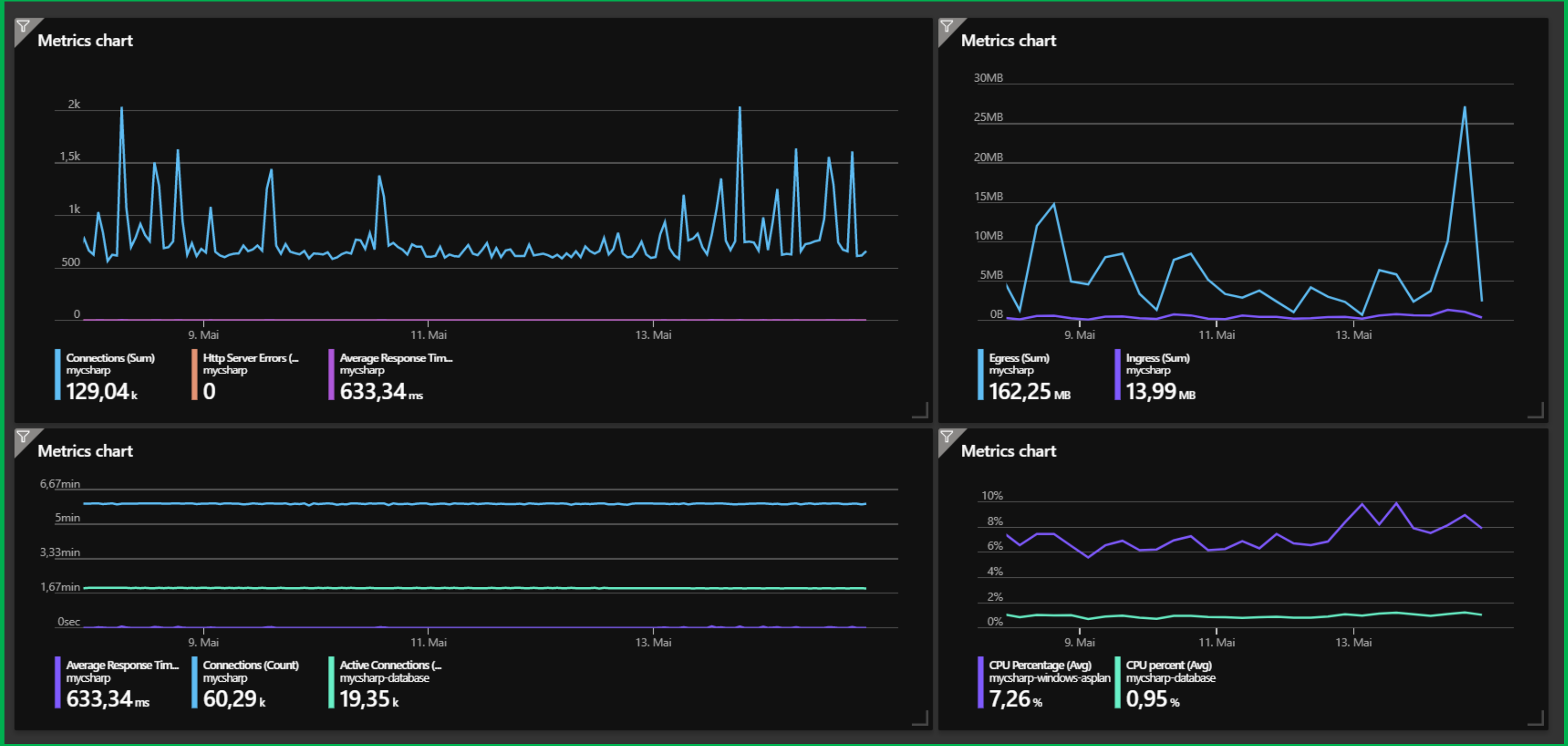
Best Practise – Serilog

- Leading .NET Log Framework
- Uses Microsoft Logging Interface
- Hundreds of Sinks supported
- Very very easy to use
- High benefit



Sink Name	`WriteTo.*`	Package
Akka Actor	`AkkaActor`	Serilog.Sinks.AkkaActor nuget v1.0.0.3 ▾ 778
Alternate Rolling File	`RollingFileAlternate`	Serilog.Sinks.RollingFileAlternate nuget v2.0.9 ▾ 320.3k
Amazon CloudWatch	`AmazonCloudWatch`	Serilog.Sinks.AwsCloudWatch nuget v4.0.149 ▾ 458.8k
Amazon DynamoDB	`DynamoDB`	Serilog.Sinks.DynamoDB nuget v0.2.12 ▾ 2.2k
Amazon Kinesis	`AmazonKinesis`	Serilog.Sinks.AmazonKinesis nuget v2.2.118 ▾ 35.1k
Application Insights	`ApplicationInsights`	Serilog.Sinks.ApplicationInsights nuget v3.0.3 ▾ 2.22m
Async Wrapper	`Async`	Serilog.Sinks.Async nuget v1.3.0 ▾ 3.00m
Azure Analytics	`AzureAnalytics`	Serilog.Sinks.AzureAnalytics nuget v4.5.0 ▾ 140.1k
Azure Blob Storage	`AzureBlobStorage`	Serilog.Sinks.AzureBlobStorage nuget v1.3.0 ▾ 10.3k
Azure DocumentDB	`AzureDocumentDB`	Serilog.Sinks.AzureDocumentDB nuget v4.0.0 ▾ 135.4k
Azure Event Grid	`EventGrid`	Serilog.Sinks.EventGrid nuget v1.1.1 ▾ 7.6k

Best Practise – Serilog Application Insights



Metrics chart

Connections (Sum) mycsharp 129,04 k

Http Server Errors (... mycsharp 0

Average Response Tim... mycsharp 633,34 ms

Metrics chart

Egress (Sum) mycsharp 162,25 MB

Ingress (Sum) mycsharp 13,99 MB

Metrics chart

Average Response Tim... mycsharp 633,34 ms

Connections (Count) mycsharp 60,29 k

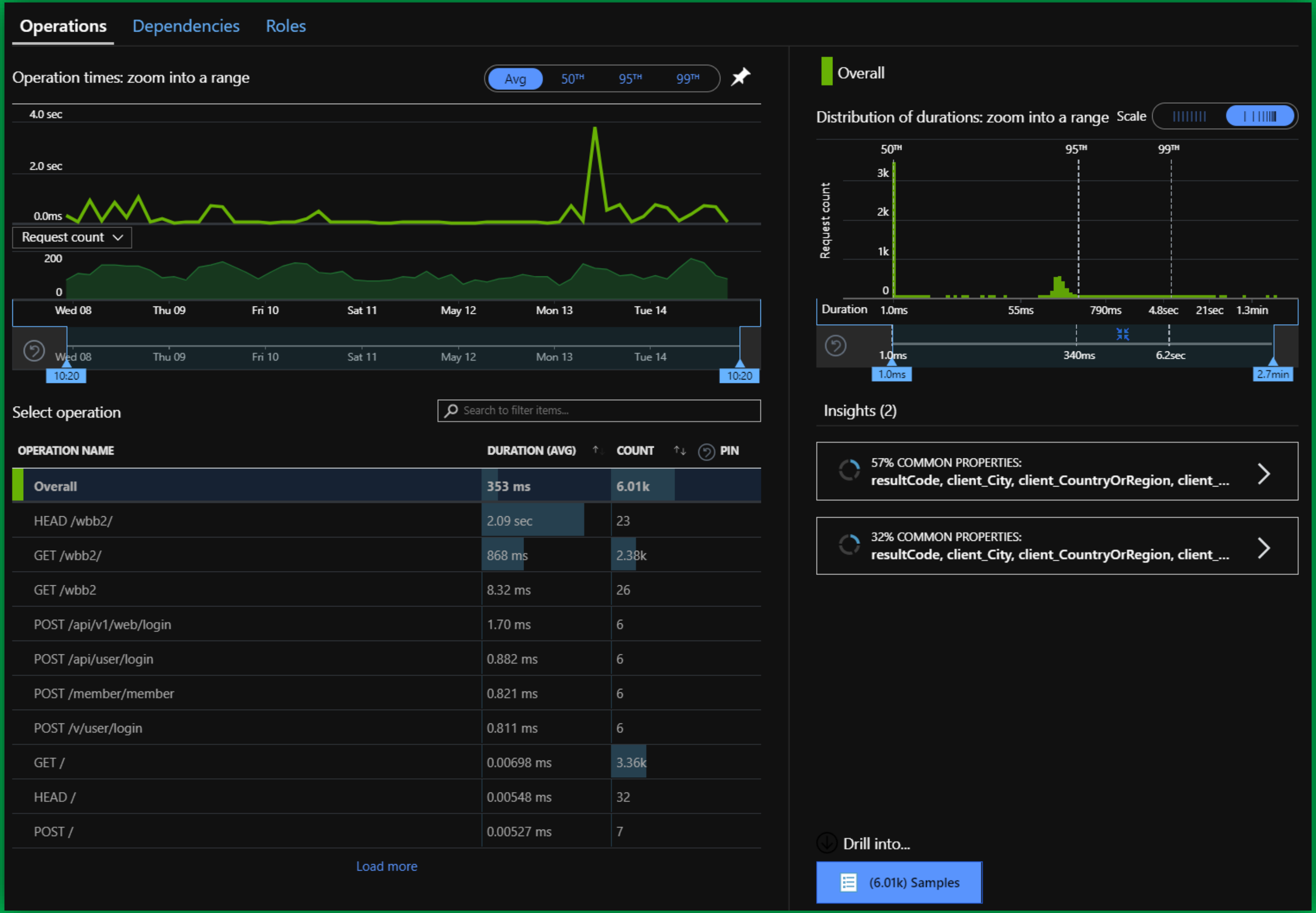
Active Connections (... mycsharp-database 19,35 k

Metrics chart

CPU Percentage (Avg) mycsharp-windows-asplan 7,26 %

CPU percent (Avg) mycsharp-database 0,95 %

Best Practise – Serilog Application Insights



Select operation

OPERATION NAME	DURATION (AVG)	COUNT	PIN
Overall	353 ms	6.01k	
HEAD /wbb2/	2.09 sec	23	
GET /wbb2/	868 ms	2.38k	
GET /wbb2	8.32 ms	26	
POST /api/v1/web/login	1.70 ms	6	
POST /api/user/login	0.882 ms	6	
POST /member/member	0.821 ms	6	
POST /v/user/login	0.811 ms	6	
GET /	0.00698 ms	3.36k	
HEAD /	0.00548 ms	32	
POST /	0.00527 ms	7	

Load more

Insights (2)

57% COMMON PROPERTIES:
resultCode, client_City, client_CountryOrRegion, client_...

32% COMMON PROPERTIES:
resultCode, client_City, client_CountryOrRegion, client_...

Drill into...

(6.01k) Samples

Best Practise - Output

Controllers (MVC)

Complex(er) Applications

Complex(er) Routing

Multiple Output formats

Decoupled Views

Razor Pages

Simple Application

Simple Routing

HTML Only

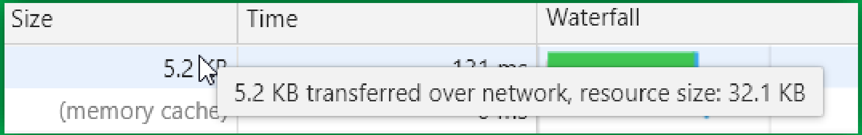
Single Responsibility (Page=Action)

Feels a bit like WebForms

Best Practise - Compression

- Use Compression!
  - Use Brotli, if possible
  - Use Gzip as fallback

- Enable HTTPS Compression!



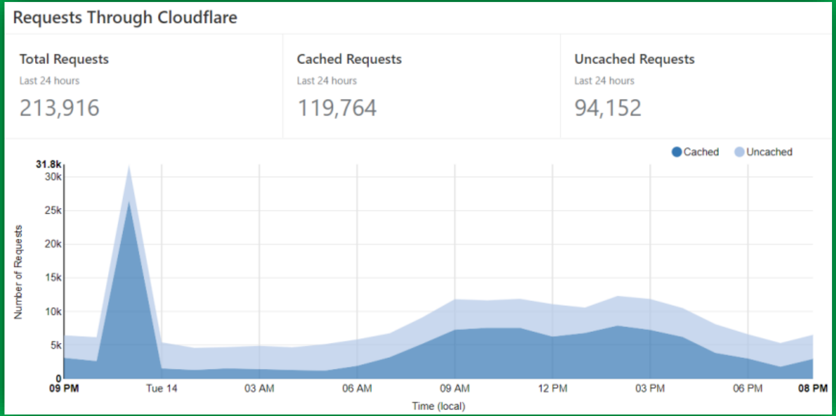
Size	Time	Waterfall
5.2 KB	121 ms	
(memory cache)	0 ms	

5.2 KB transferred over network, resource size: 32.1 KB

Best Practise - Caching

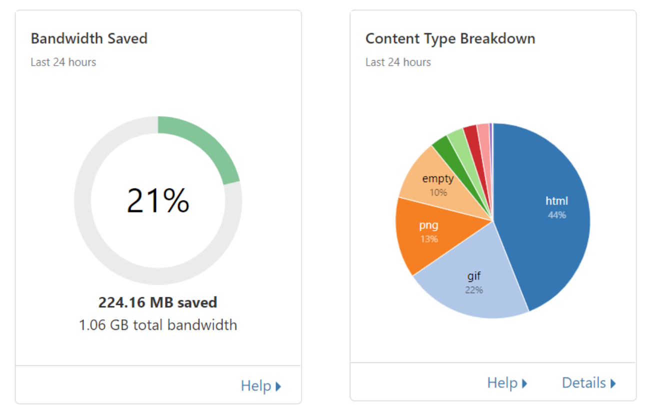
- Cache resources for at least 30 days
- Use version-parameters for dynamic resources
- **DO NOT CACHE SENSETIVE CONTENT**!

Best Practise – Sample dynamic (Resources only)



Requests Through Cloudflare

Total Requests	Cached Requests	Uncached Requests
Last 24 hours	Last 24 hours	Last 24 hours
213,916	119,764	94,152



Bandwidth Saved

Last 24 hours

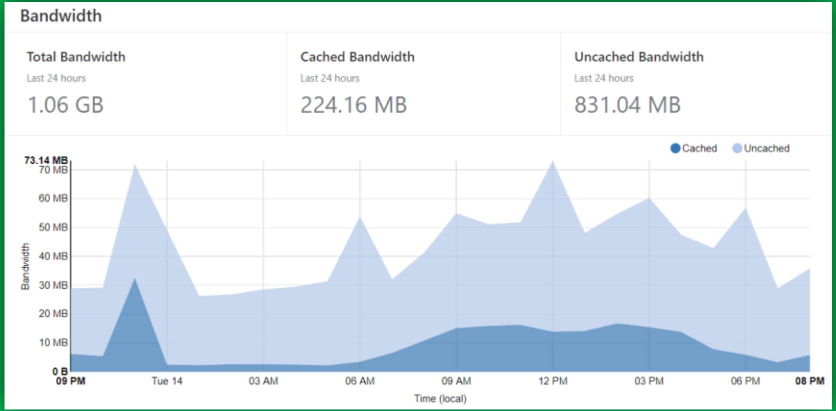
21%

224.16 MB saved

1.06 GB total bandwidth

Content Type Breakdown

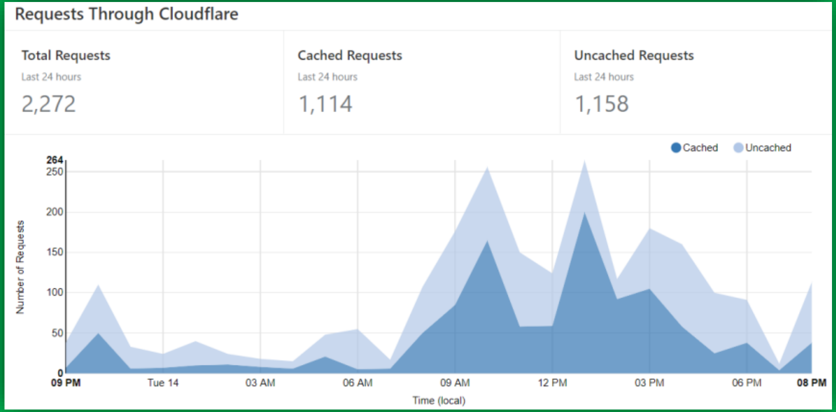
Last 24 hours



Bandwidth

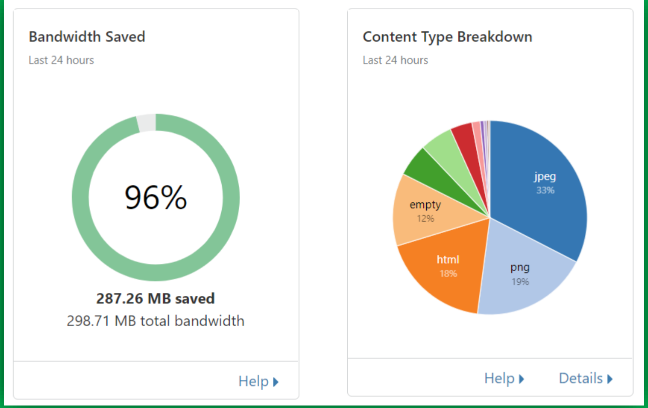
Total Bandwidth	Cached Bandwidth	Uncached Bandwidth
Last 24 hours	Last 24 hours	Last 24 hours
1.06 GB	224.16 MB	831.04 MB

Best Practise – Sample static (Cache everything, external CDN)



Requests Through Cloudflare

Total Requests	Cached Requests	Uncached Requests
Last 24 hours	Last 24 hours	Last 24 hours
2,272	1,114	1,158



Bandwidth Saved

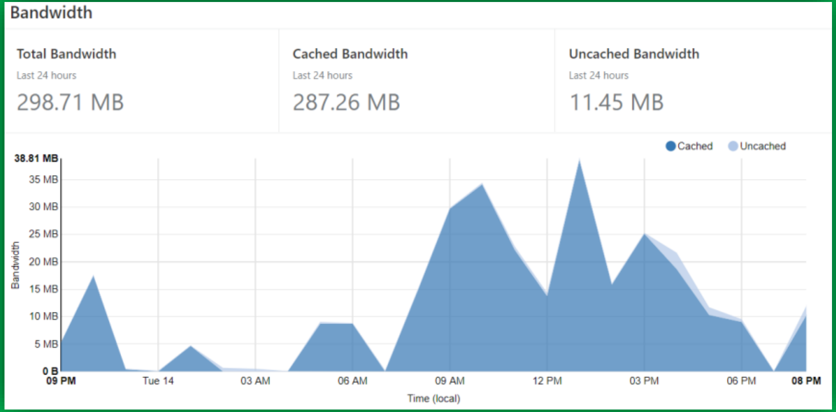
Last 24 hours

287.26 MB saved

298.71 MB total bandwidth

Content Type Breakdown

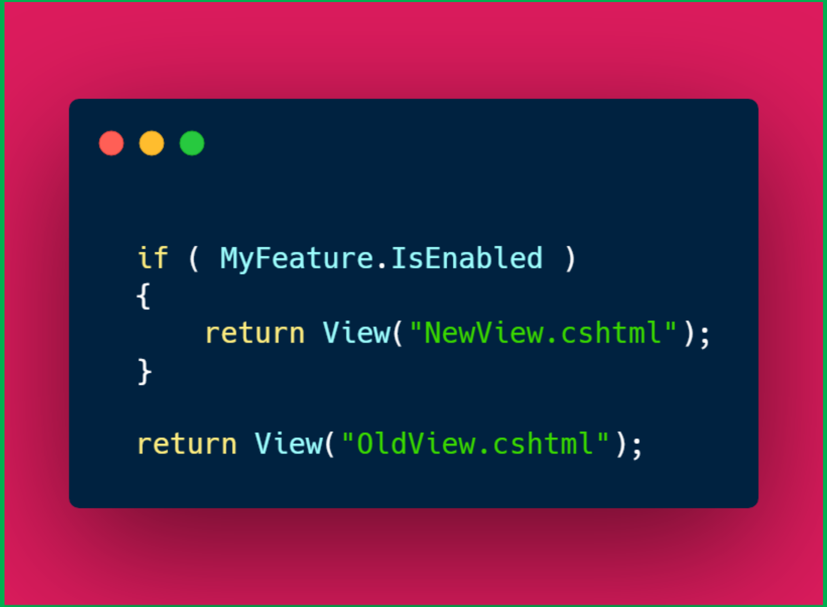
Last 24 hours



Bandwidth

Total Bandwidth	Cached Bandwidth	Uncached Bandwidth
Last 24 hours	Last 24 hours	Last 24 hours
298.71 MB	287.26 MB	11.45 MB

Best Practise – Feature Flags



```
if ( MyFeature.IsEnabled )
{
    return View("NewView.cshtml");
}

return View("OldView.cshtml");
```

Frameworks:

- FeatureSwitcher
- FeatureBits

Services:

- LaunchDarkly