Building scalable cloud native apps with .NET 8

Adam Sitnik

Cloud-native .NET

- Previous .NET investments:
  - YARP
  - Project Tye
  - Health Checks
  - HttpClientFactory
  - gRPC
  - SDK Container Builds
  - NativeAOT
- .NET 8:
  - R9
  - Aspire

R9 – internal SDK for high-scale services

- Which parts of R9 should be shared in the open?
- Move general-purpose concepts to dotnet/runtime
- Move ASP.NET-related concepts into dotnet/aspnetcore
- Move other things to dotnet/extensions

System.TimeProvider: time abstraction!

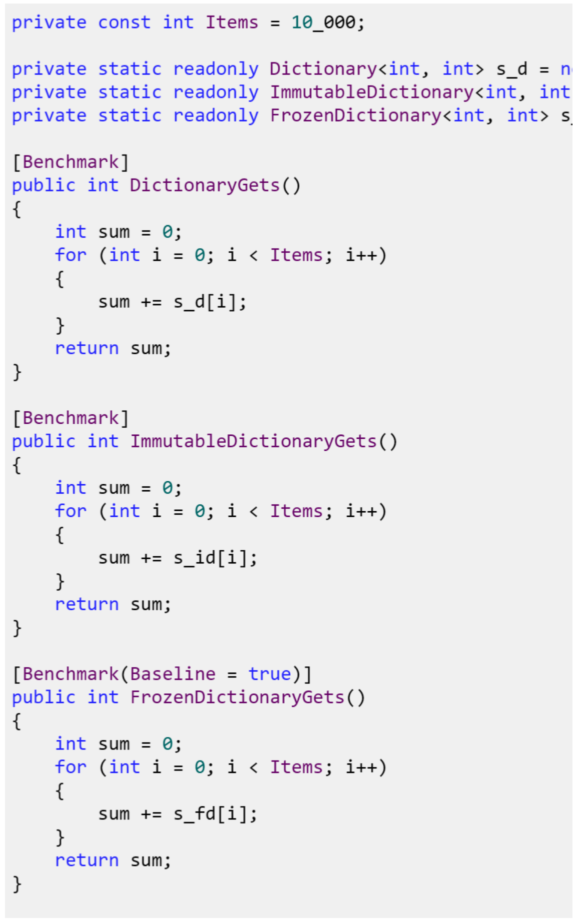
- API to provide the current system time #36617

```
public abstract class TimeProvider
{
    public static TimeProvider System { get; }
    protected TimeProvider();
    public virtual DateTimeOffset GetUtcNow();
    public DateTimeOffset GetLocalNow();
    public virtual TimeZoneInfo LocalTimeZone { get; }
    public virtual long TimestampFrequency { get; }
    public virtual long GetTimestamp();
    public TimeSpan GetElapsedTime(long startingTimestamp);
    public TimeSpan GetElapsedTime(long startingTimestamp, long endingTimestamp);
    public virtual ITimer CreateTimer(TimerCallback callback, object? state, TimeSpan dueTime, TimeSpan period);
}
```

Frozen Collections

- Provide optimized read-only collections #67209
- Two new types:
  - FrozenDictionary<TKey, TValue>
  - FrozenSet<TKey, TValue>
- Part of System.Collections.Immutable NuGet package
- Expensive to create but provide very fast read access.
- Why existing immutable collections were not improved instead?
- More: https://devblogs.microsoft.com/dotnet/performance-improvements-in-net-8/#frozen-collections

FozenDictionary<int, int> benchmarks



```
private const int Items = 10_000;

private static readonly Dictionary<int, int> s_d = n
private static readonly ImmutableDictionary<int, int
private static readonly FrozenDictionary<int, int> s_

[Benchmark]
public int DictionaryGets()
{
    int sum = 0;
    for (int i = 0; i < Items; i++)
    {
        sum += s_d[i];
    }
    return sum;
}

[Benchmark]
public int ImmutableDictionaryGets()
{
    int sum = 0;
    for (int i = 0; i < Items; i++)
    {
        sum += s_id[i];
    }
    return sum;
}

[Benchmark(Baseline = true)]
public int FrozenDictionaryGets()
{
    int sum = 0;
    for (int i = 0; i < Items; i++)
    {
        sum += s_fd[i];
    }
    return sum;
}
```

Method	Mean	Ratio
ImmutableDictionaryGets	360.55 us	**13.89**
DictionaryGets	39.43 us	**1.52**
FrozenDictionaryGets	25.95 us	**1.00**

FrozenSet<string> benchmarks

```
private static readonly HashSet<string> s_s = new(StringComparer.OrdinalIgnoreCase)
{
    "Olivia", "Emma", "Charlotte", "Amelia", "Sophia", "Isabella", "Ava", "Mia", "Evelyn", "Luna"
};
private static readonly FrozenSet<string> s_fs = s_s.ToFrozenSet(StringComparer.OrdinalIgnoreCase);

[Benchmark(Baseline = true)]
public bool HashSet_IsMostPopular() => s_s.Contains("Alexandria");

[Benchmark]
public bool FrozenSet_IsMostPopular() => s_fs.Contains("Alexandria");
```

Method	Mean	Ratio
HashSet_IsMostPopular	9.824 ns	1.00
FrozenSet_IsMostPopular	1.518 ns	**0.15**

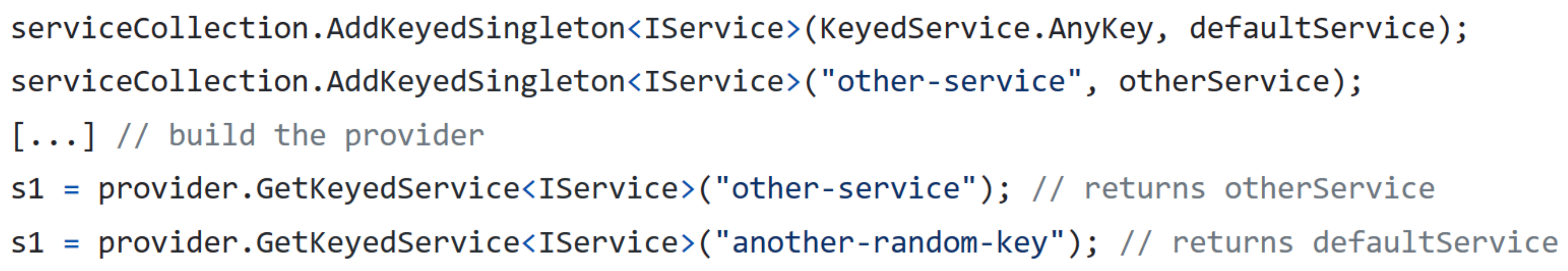
ReadOnly empty collection singletons

```diff
namespace System.Collections.Generic
{
    public interface IReadOnlyCollection<out T>
    {
+       public static IReadOnlyCollection<T> Empty { get; }
    }

    public interface IReadOnlyList<out T> : IReadOnlyCollection<T>
    {
+       public static new IReadOnlyList<T> Empty { get; }
    }

    public interface IReadOnlyDictionary<TKey, TValue> : IReadOnlyCollection
    {
+       public static new IReadOnlyDictionary<TKey, TValue> Empty { get; }
    }

    public interface IReadOnlySet<T> : IReadOnlyCollection<T>
    {
+       public static new IReadOnlySet<T> Empty { get; }
    }
}
```

```diff
namespace System.Collections.ObjectModel
{
    public class ReadOnlyCollection<T>
    {
+       public static ReadOnlyCollection<T> Empty { get; }
    }

    public class ReadOnlyDictionary<TKey, TValue>
    {
+       public static ReadOnlyDictionary<TKey, TValue> Empty { get; }
    }

    public class ReadOnlyObservableCollection<T>
    {
+       public static ReadOnlyObservableCollection<T> Empty { get; }
    }
}
```

Keyed DI 1/2

- Add Keyed Services Support to Dependency Injection #64427



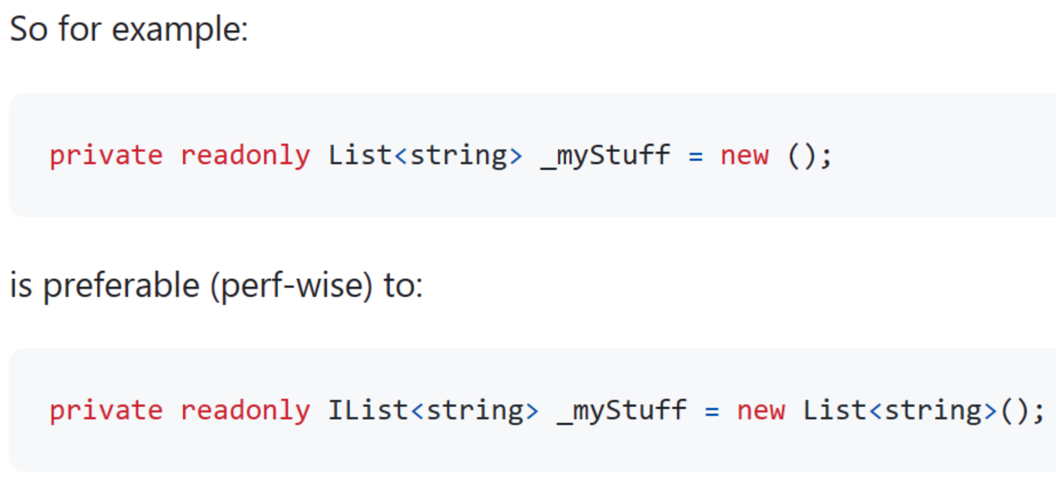
```
serviceCollection.AddKeyedSingleton<IService>(KeyedService.AnyKey, defaultService);
serviceCollection.AddKeyedSingleton<IService>("other-service", otherService);
[...] // build the provider
s1 = provider.GetKeyedService<IService>("other-service"); // returns otherService
s1 = provider.GetKeyedService<IService>("another-random-key"); // returns defaultService
```

Keyed DI 2/2

```
namespace Microsoft.Extensions.DependencyInjection;
[AttributeUsageAttribute(AttributeTargets.Parameter)]
public class FromKeyedServicesAttribute : Attribute
{
    public FromKeyedServicesAttribute(object key) { }
    public object Key { get; }
}

class OtherService
{
    public OtherService(
        [FromKeyedServices("service1")] IService service1,
        [FromKeyedServices("service2")] IService service2)
    {
        Service1 = service1;
        Service2 = service2;
    }
}
```

Analyzers

- Recommend use of concrete types to maximize devirtualization potential #51193



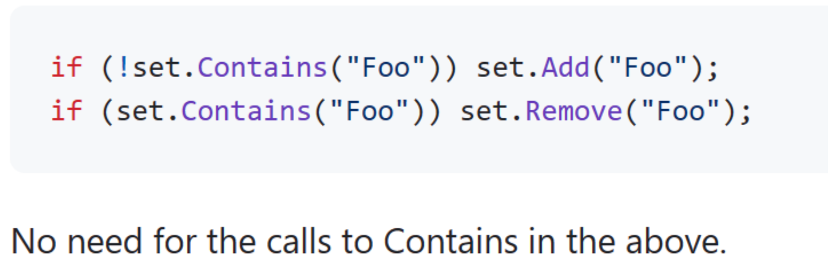
So for example:

```
private readonly List<string> _myStuff = new ();
```

is preferable (perf-wise) to:

```
private readonly IList<string> _myStuff = new List<string>();
```

- Report inefficient use of sets. #85490



```
if (!set.Contains("Foo")) set.Add("Foo");
if (set.Contains("Foo")) set.Remove("Foo");
```

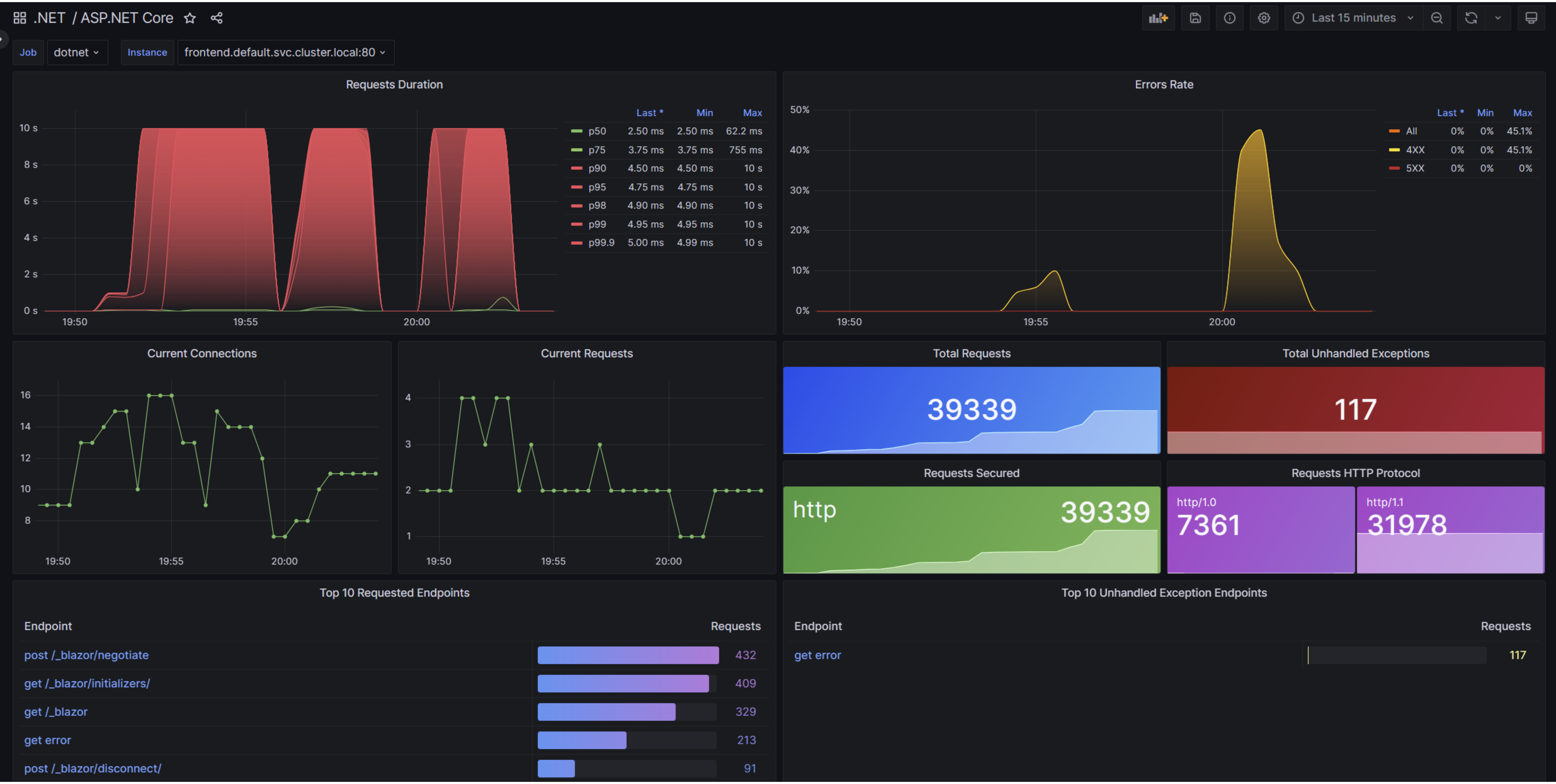
No need for the calls to Contains in the above.

dotnet/runtime: even more

- Full list: Libraries work to support Cloud Native effort for .NET 8 #78518
- Don’t miss:
  - Boost performance of localized, non-static or incremental string formatting #50330
  - Base64.IsValid #76020
  - XXH3 #75948
  - Introduce DI friendly IMeter<T> #77514
  - Add a type to represent IP networks based on CIDR notation #79946
- Check .NET 9 part!

ASP.NET: Metrics!

- Measurements reported over time.
- Used to monitor the health of an app and to generate alerts.
- System.Diagnostics.Metrics vs Event Counters:
  - New kinds: counters, gauges and histograms
  - Multi-dimensional
  - Integration into the wider cloud native eco-system by aligning with OpenTelemetry standards
- Metrics have been added for ASP.NET Core hosting, Kestrel and SignalR.
- And documented in #37875



Requests Duration

	Last *	Min	Max
p50	2.50 ms	2.50 ms	62.2 ms
p75	3.75 ms	3.75 ms	755 ms
p90	4.50 ms	4.50 ms	10 s
p95	4.75 ms	4.75 ms	10 s
p98	4.90 ms	4.90 ms	10 s
p99	4.95 ms	4.95 ms	10 s
p99.9	5.00 ms	4.99 ms	10 s

Errors Rate

	Last *	Min	Max
All	0%	0%	45.1%
4XX	0%	0%	45.1%
5XX	0%	0%	0%

Top 10 Requested Endpoints

Endpoint	Requests
post /_blazor/negotiate	432
get /_blazor/initializers/	409
get /_blazor	329
get error	213
post /_blazor/disconnect/	91

Top 10 Unhandled Exception Endpoints

Endpoint	Requests
get error	117

ASP.NET Core updates in .NET 8 Preview 4 - .NET Blog (microsoft.com)

ASP.NET: IExceptionHandler middleware #46280

```
public interface IExceptionHandler
{
   ValueTask<bool> TryHandleAsync(HttpContext httpContext, Exception exception, CancellationToken cancellationToken);
}
public static class ExceptionHandlerServiceCollectionExtensions
{
   public static IServiceCollection AddExceptionHandler<T>(this IServiceCollection services) where T is IExceptionHandler;
}
```

„We don't want to encourage developers to use exceptions as normal control flow, it's extremely in-efficient and poor design. We should document that expectation and avoid using this capability in our own components."

```
services.AddExceptionHandler<DBExceptionHandler>();
```

```
public class DBExceptionHandler : IExceptionHandler
{
  ValueTask<bool> TryHandleAsync(HttpContext httpContext, Exception exception, CancellationToken cancellationToken)
  {
    if (exception is DBException) ...
  };
}
```

ASP.NET: Route Short Circuit middleware #46071

```
public static class RouteShortCircuitEndpointConventionBuilderExtensions
{
    public static IEndpointConventionBuilder ShortCircuit(this IEndpointConventionBuilder builder) { }
}

public static class RouteShortCircuitEndpointRouteBuilderExtensions
{
    public static IEndpointConventionBuilder MapShortCircuit(this IEndpointConventionBuilder builder, params string[] routePrefixes);
}
```

```
app.MapGet("/favicon.ico", httpContext =>
{
    // httpContext.Response.StatusCode will default to 404 here.
    httpContext.RequestServices.GetService<ILogger<MyThing>>().LogTrace("Skipped favicon.ico");
}).ShortCircuit();

app.MapGet("/short-circuit-prefix/**", httpContext => { .. }).ShortCircuit();

// So /favicon.ico/test/extra/path would also be rejected
// MapRejects 404s and implies ShortCircuit()
app.MapShortCircuit("/favicon.ico", "/short-circuit-prefix", "/another-prefix");
```

ASP.NET: even more

- Introduce IResettable to streamline object pool usage #44901
- Request timeout middleware #46115
- Add Counters & Logs for 404s and MapFallback #46404
- Make Http Logging Middleware Endpoint aware #43222
- HttpLoggingMiddleware could allow custom code to decide whether to log or not #39200
- HttpLogging redaction and enrichment #31844

Extensions for building cloud services

.NET 8

dotnet/extensions

This repository contains a suite of libraries that provide facilities commonly needed when creating production-ready applications. Initially developed to support high-scale and high-availability services within Microsoft, such as Microsoft Teams, these libraries deliver functionality that can help make applications more efficient, more robust, and more manageable.

Testing

- Fake It Til You Make It...To Production - .NET Blog (microsoft.com)
- Logging (Microsoft.Extensions.Diagnostics.Testing)
  - FakeLogger, FakeLogger<T>, FakeLoggerProvider
  - FakeLogger exposes .Collector property which returns FakeLogCollector
  - FakeLogCollector exposes Count, GetSnapshot() and LatestRecord
- Metrics (Microsoft.Extensions.Diagnostics.Testing)
  - MetricCollector<T> for recording, provides LastMeasurement and GetMeasurementSnapshot
- FakeTimeProvider (Microsoft.Extensions.TimeProvider.Testing)
  - You can set specific time and increment it
  - You can specify a value that is added every time the time is read

Microsoft.Extensions.Http.Resilience

- HTTP-based resilience APIs build atop Polly:
  - Rate limiter (max number of concurrent requests to dependency)
  - Total request timeout
  - Retry (for transient errors and slow responses)
  - Circuit breaker (stop on too many failures/timeouts)
  - Attempt timeout
- Microsoft.Extensions.Resilience: minimal set of APIs, enrich Polly's metrics
- Building resilient cloud services with .NET 8 - .NET Blog (microsoft.com)
- Meet Polly: The .NET resilience library | Polly (pollydocs.org)

AddStandardResilienceHandler: defaults

```
using Http.Resilience.Example;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

HostApplicationBuilder builder = Host.CreateApplicationBuilder(args);
IServiceCollection services = builder.Services;

services.AddHttpClient("my-client")
        .AddStandardResilienceHandler(options =>
        {
            // Configure standard resilience options here
        });

// Use the client
var host = builder.Build();
var httpClient = host.Services
    .GetRequiredService<IHttpClientFactory>()
    .CreateClient("my-client");

// Make resilient HTTP request
HttpResponseMessage response = await httpClient.GetAsync("https://jsonplaceholder.typicode.com/comments");
```

AddResilienceHandler: customization

```
services.AddHttpClient("my-client")
        .AddResilienceHandler("my-pipeline", builder =>
        {
            // Refer to https://www.pollydocs.org/strategies/retry.html#defaults for retry defaults
            builder.AddRetry(new HttpRetryStrategyOptions
            {
                MaxRetryAttempts = 4,
                Delay = TimeSpan.FromSeconds(2),
                BackoffType = DelayBackoffType.Exponential
            });

            // Refer to https://www.pollydocs.org/strategies/timeout.html#defaults for timeout defaults
            builder.AddTimeout(TimeSpan.FromSeconds(5));
        });
```

Hedging

improve request latency by issuing multiple concurrent requests

```
services
    .AddHttpClient("my-client")
    .AddStandardHedgingHandler(routingBuilder =>
    {
        routingBuilder.ConfigureOrderedGroups(options =>
        {
            options.Groups.Add(new UriEndpointGroup
            {
                Endpoints =
                {
                    new() { Uri = new("https://example.net/api/a"), Weight = 95 },
                    new() { Uri = new("https://example.net/api/b"), Weight = 5 },
                }
            });

            options.Groups.Add(new UriEndpointGroup
            {
                Endpoints =
                {
                    new() { Uri = new("https://example.net/api/c"), Weight = 95 },
                    new() { Uri = new("https://example.net/api/d"), Weight = 5 },
                }
            });
        });
    });
```

Blog posts coming „soon” ;)

- Microsoft.AspNetCore.AsyncState
- Microsoft.AspNetCore.Diagnostics.Middleware
- Microsoft.AspNetCore.HeaderParsing
- Microsoft.AspNetCore.Testing
- Microsoft.Extensions.AmbientMetadata.Application
- Microsoft.Extensions.AsyncState
- Microsoft.Extensions.AuditReports
- Microsoft.Extensions.Compliance.Abstractions
- Microsoft.Extensions.Compliance.Redaction
- Microsoft.Extensions.Compliance.Testing
- Microsoft.Extensions.DependencyInjection.AutoActivation
- Microsoft.Extensions.Diagnostics.ExceptionSummarization
- Microsoft.Extensions.Diagnostics.HealthChecks.Common
- Microsoft.Extensions.Diagnostics.HealthChecks.ResourceUtilization
- Microsoft.Extensions.Diagnostics.Probes
- Microsoft.Extensions.Diagnostics.ResourceMonitoring
- Microsoft.Extensions.Diagnostics.Testing
- Microsoft.Extensions.ExtraAnalyzers
- Microsoft.Extensions.Hosting.Testing
- Microsoft.Extensions.Http.Diagnostics
- Microsoft.Extensions.Http.Resilience
- Microsoft.Extensions.ObjectPool.DependencyInjection
- Microsoft.Extensions.Options.Contextual
- Microsoft.Extensions.Resilience
- Microsoft.Extensions.Telemetry
- Microsoft.Extensions.Telemetry.Abstractions
- Microsoft.Extensions.TimeProvider.Testing

https://github.com/dotnet/extensions-samples

Is it enough to easily build cloud services?

.NET Aspire

A cloud ready stack for building observable, production ready, distributed applications

First Preview Available Today

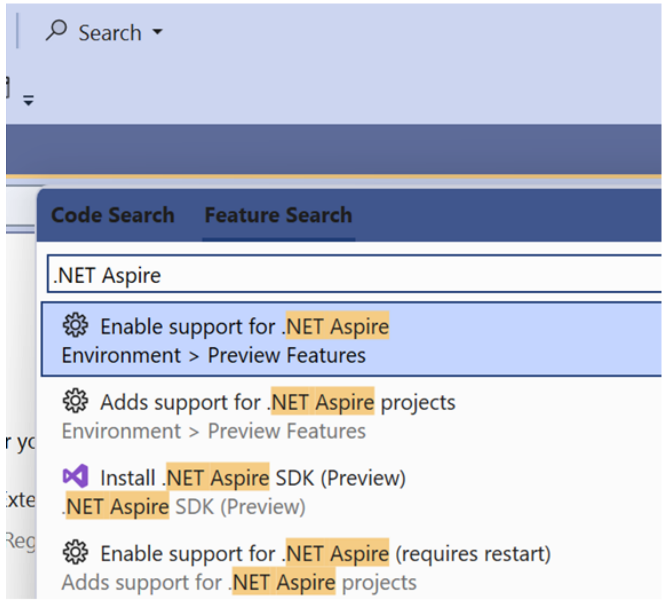
Engage with team on GitHub

aka.ms/dotnet-aspire

github.com/dotnet/aspire

Dev Loop on Windows atm

- VS 2022 17.9 Preview 1+
- .NET Aspire SDK
- WSL
- Docker Desktop



Search

Code Search Feature Search

.NET Aspire

Enable support for .NET Aspire
Environment > Preview Features

Adds support for .NET Aspire projects
Environment > Preview Features

Install .NET Aspire SDK (Preview)
.NET Aspire SDK (Preview)

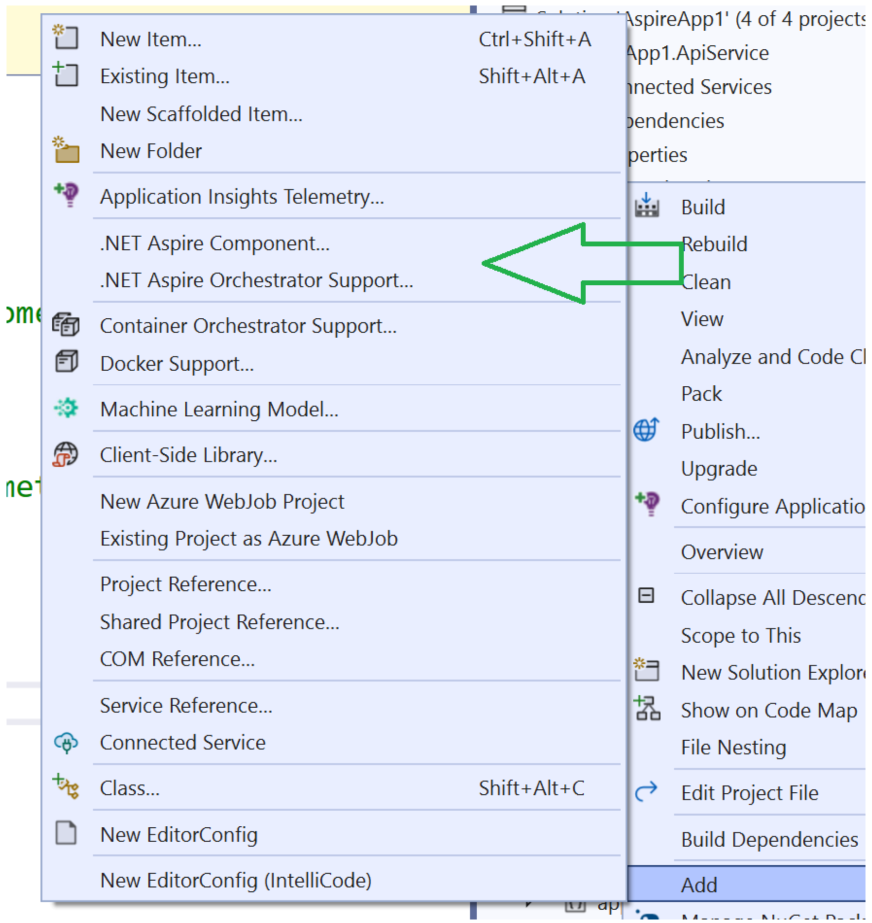
Enable support for .NET Aspire (requires restart)
Adds support for .NET Aspire projects

Non-VS scenarios

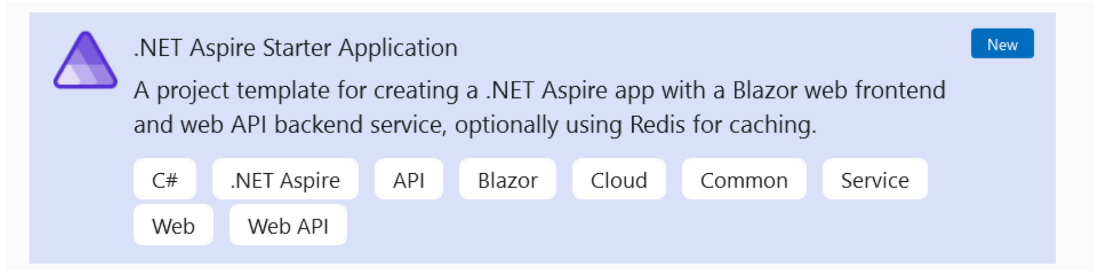
- https://learn.microsoft.com/dotnet/aspire/setup-tooling:

“Visual Studio provides additional features for working with .NET Aspire components and the App Host orchestrator project. These features are currently not available in Visual Studio Code or through the CLI.”

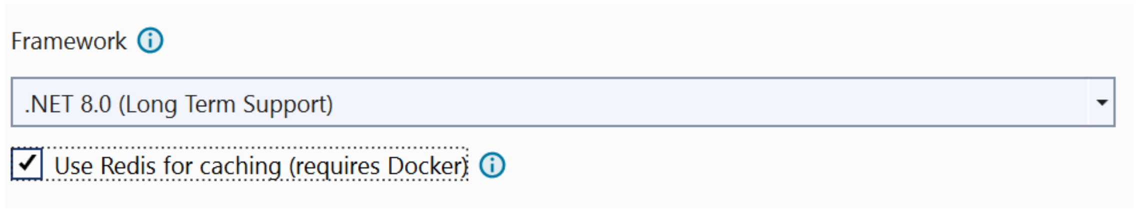
- No VS: dotnet workload install aspire
- [.NET Aspire support : RIDER-101760 (jetbrains.com)](#)



Demo: .NET Aspire Starter Application


.NET Aspire Starter Application

A project template for creating a .NET Aspire app with a Blazor web frontend and web API backend service, optionally using Redis for caching.

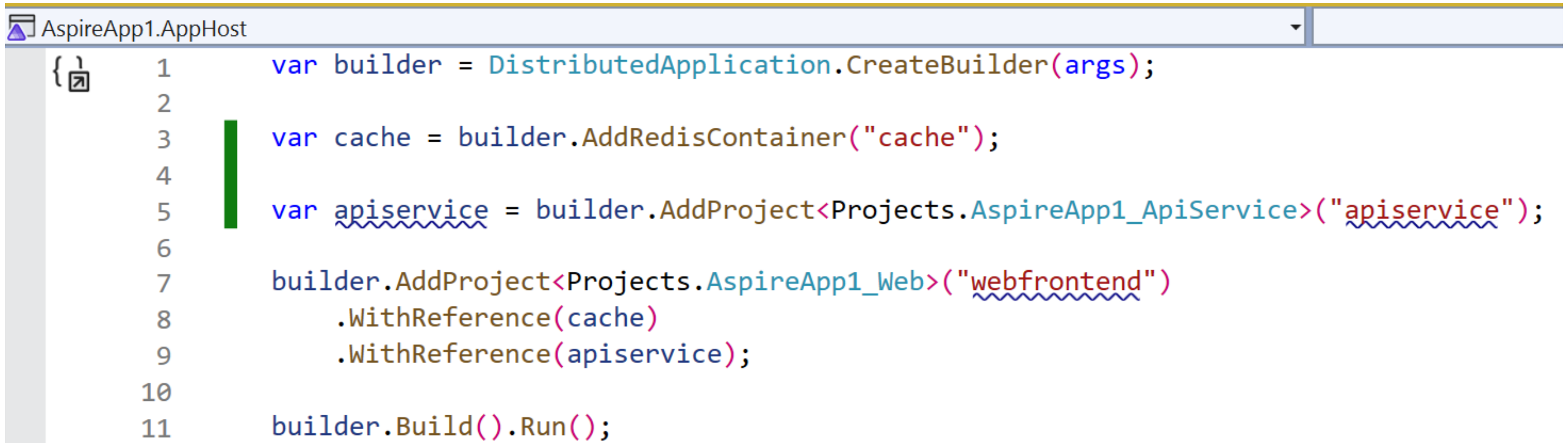

Framework: .NET 8.0 (Long Term Support)

[✓] Use Redis for caching (requires Docker)

Use ".NET Aspire Application" template for new real-life apps

VS Shortcuts: Zoom It: Ctrl+Shift+., Out: Ctrl+Shift+,

AppHost: the glue

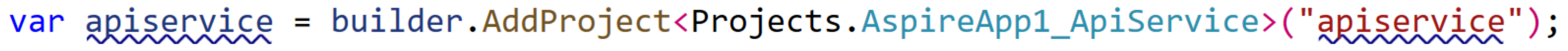


```csharp
var builder = DistributedApplication.CreateBuilder(args);

var cache = builder.AddRedisContainer("cache");

var apiservice = builder.AddProject<Projects.AspireApp1_ApiService>("apiservice");

builder.AddProject<Projects.AspireApp1_Web>("webfrontend")
    .WithReference(cache)
    .WithReference(apiservice);

builder.Build().Run();
```

- You express the needs of your distributed application:
  - What containers, executables and/or cloud resources you need.
  - Which projects should be added.
  - The references between the projects.
  - And IDs for all the above so they can communicate with each other.
- AppHost is responsible for orchestrating the running of your app on your dev machine.

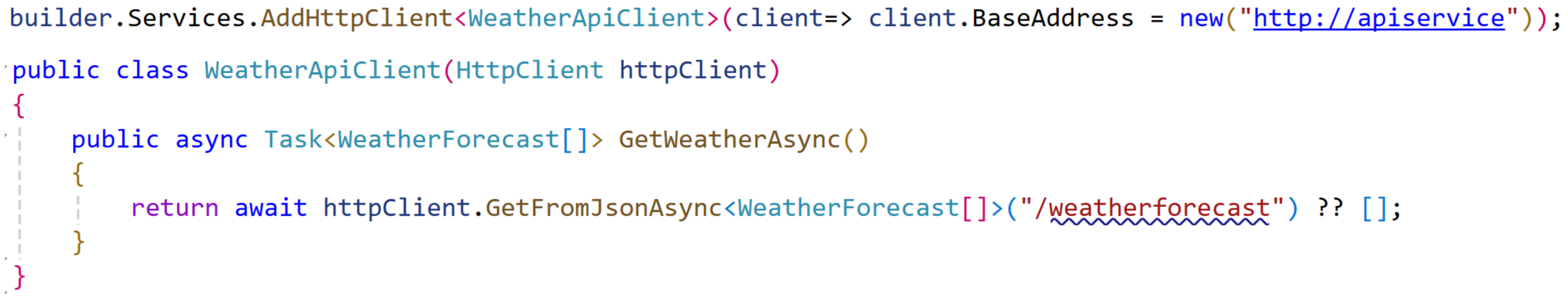
Named endpoints

- AppHost:



```
var apiservice = builder.AddProject<Projects.AspireApp1_ApiService>("apiservice");
```

- App:



```
builder.Services.AddHttpClient<WeatherApiClient>(client=> client.BaseAddress = new("http://apiservice"));

public class WeatherApiClient(HttpClient httpClient)
{
    public async Task<WeatherForecast[]> GetWeatherAsync()
    {
        return await httpClient.GetFromJsonAsync<WeatherForecast[]>("/weatherforecast") ?? [];
    }
}
```

- .NET Aspire service discovery - .NET Aspire | Microsoft Learn

Components

- NuGet package per client library, name: “Aspire.$ClientLibraryName”
  - Examples:
    - Aspire.Npgsql
    - Aspire.Azure.Storage.Queues
    - Aspire.StackExchange.Redis
- Register client in the DI, following best practices (singletons etc)
- Add a health check (enabled by default)
- Offer integrated logging, metrics and tracing (enabled by default)
- Offer resiliency with opinionated, reasonable defaults
- Provide JSON schema for completions in appsettings.json (Demo)

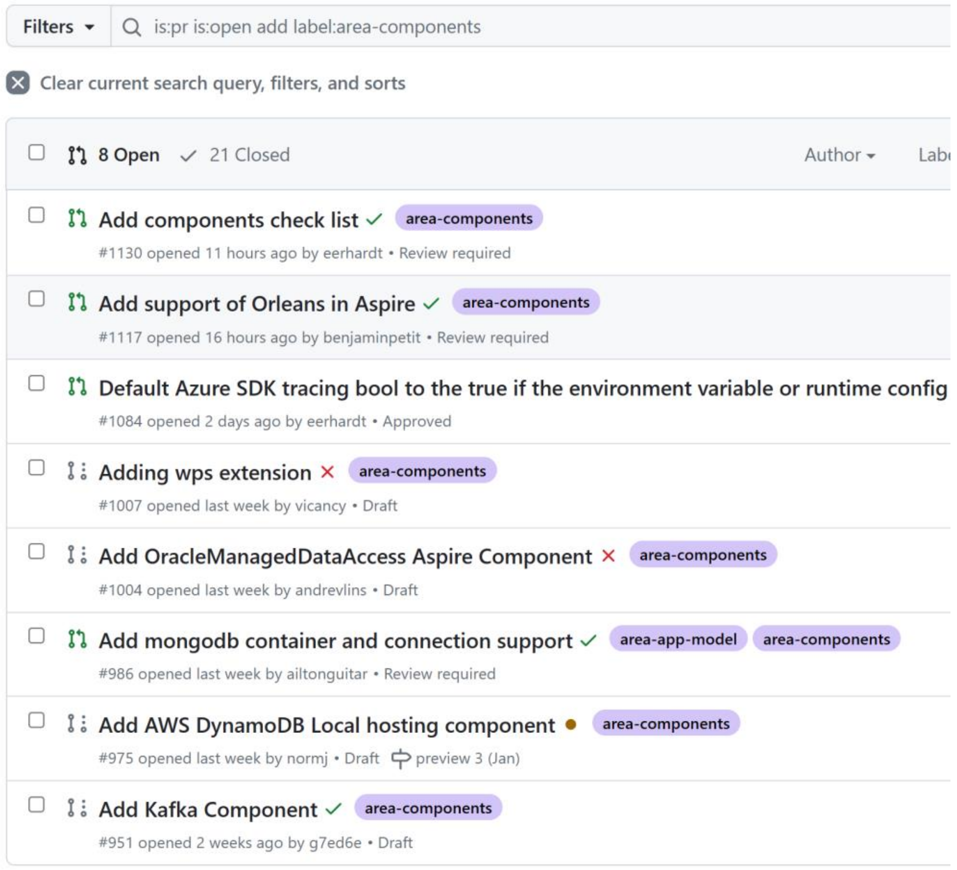
Cloud-agnostic Components

Component	Description
PostgreSQL Entity Framework Core	Provides a client library for accessing PostgreSQL databases using Entity Framework Core.
PostgreSQL	Provides a client library for accessing PostgreSQL databases.
RabbitMQ	Provides a client library for accessing RabbitMQ.
Redis Distributed Caching	Provides a client library for accessing Redis caches for distributed caching.
Redis Output Caching	Provides a client library for accessing Redis caches for output caching.
Redis	Provides a client library for accessing Redis caches.
SQL Server Entity Framework Core	Provides a client library for accessing SQL Server databases using Entity Framework Core.
SQL Server	Provides a client library for accessing SQL Server databases.

Cloud specific Components

Component	**Description**
Azure Blob Storage	Provides a client library for accessing Azure Blob Storage.
Azure Cosmos DB Entity Framework Core	Provides a client library for accessing Azure Cosmos DB databases with Entity Framework Core.
Azure Cosmos DB	Provides a client library for accessing Azure Cosmos DB databases.
Azure Key Vault	Provides a client library for accessing Azure Key Vault.
Azure Service Bus	Provides a client library for accessing Azure Service Bus.
Azure Storage Queues	Provides a client library for accessing Azure Storage Queues.

Incoming!



Filters ▾ is:pr is:open add label:area-components

Clear current search query, filters, and sorts

8 Open ✓ 21 Closed Author ▾ Lab

Add components check list ✓ area-components
#1130 opened 11 hours ago by eerhardt • Review required

Add support of Orleans in Aspire ✓ area-components
#1117 opened 16 hours ago by benjaminpetit • Review required

Default Azure SDK tracing bool to the true if the environment variable or runtime config
#1084 opened 2 days ago by eerhardt • Approved

Adding wps extension ✗ area-components
#1007 opened last week by vicancy • Draft

Add OracleManagedDataAccess Aspire Component ✗ area-components
#1004 opened last week by andrevlins • Draft

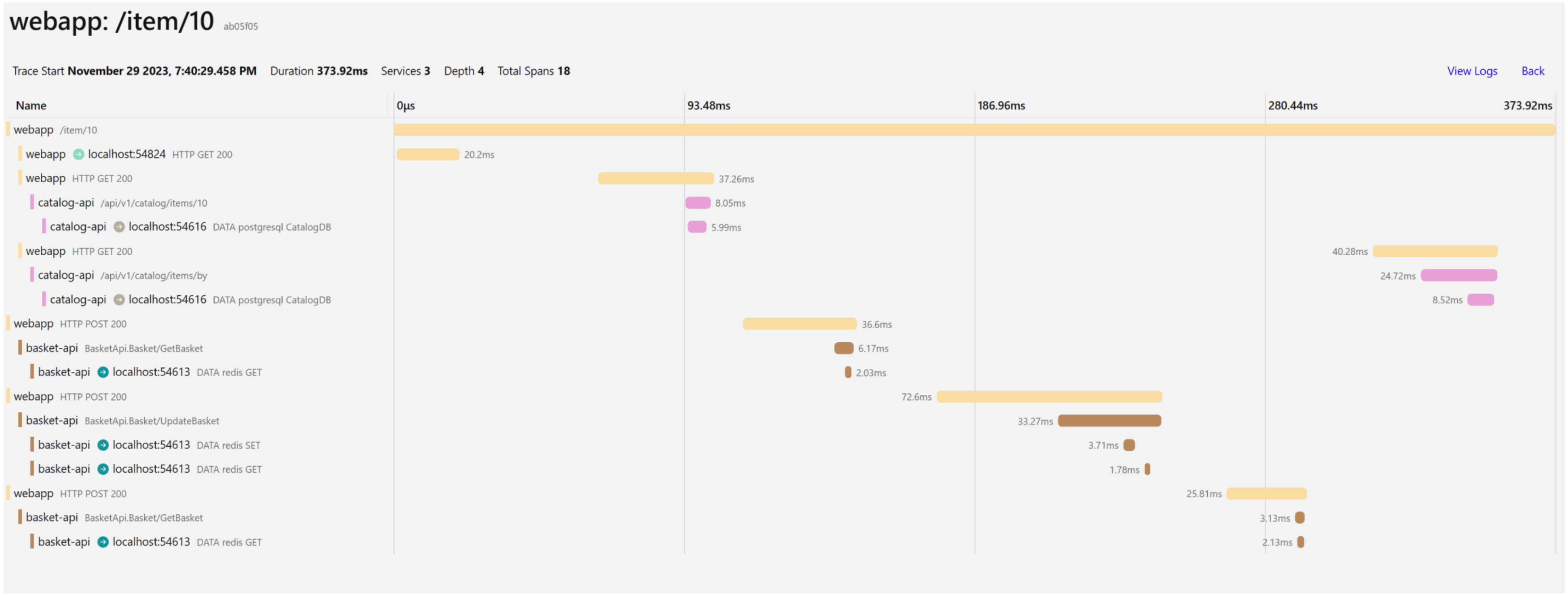
Add mongodb container and connection support ✓ area-app-model area-components
#986 opened last week by ailtonguitar • Review required

Add AWS DynamoDB Local hosting component ● area-components
#975 opened last week by normj • Draft preview 3 (Jan)

Add Kafka Component ✓ area-components
#951 opened 2 weeks ago by g7ed6e • Draft

Pull requests · dotnet/aspire (github.com)

Demo: Developer Dashboard (eShop)



webapp: /item/10 ab05f05

Trace Start **November 29 2023, 7:40:29.458 PM** Duration **373.92ms** Services **3** Depth **4** Total Spans **18** View Logs Back

Name	0µs · 93.48ms · 186.96ms · 280.44ms · 373.92ms
webapp /item/10	
webapp localhost:54824 HTTP GET 200	20.2ms
webapp HTTP GET 200	37.26ms
catalog-api /api/v1/catalog/items/10	8.05ms
catalog-api localhost:54616 DATA postgresql CatalogDB	5.99ms
webapp HTTP GET 200	40.28ms
catalog-api /api/v1/catalog/items/by	24.72ms
catalog-api localhost:54616 DATA postgresql CatalogDB	8.52ms
webapp HTTP POST 200	36.6ms
basket-api BasketApi.Basket/GetBasket	6.17ms
basket-api localhost:54613 DATA redis GET	2.03ms
webapp HTTP POST 200	72.6ms
basket-api BasketApi.Basket/UpdateBasket	33.27ms
basket-api localhost:54613 DATA redis SET	3.71ms
basket-api localhost:54613 DATA redis GET	1.78ms
webapp HTTP POST 200	25.81ms
basket-api BasketApi.Basket/GetBasket	3.13ms
basket-api localhost:54613 DATA redis GET	2.13ms

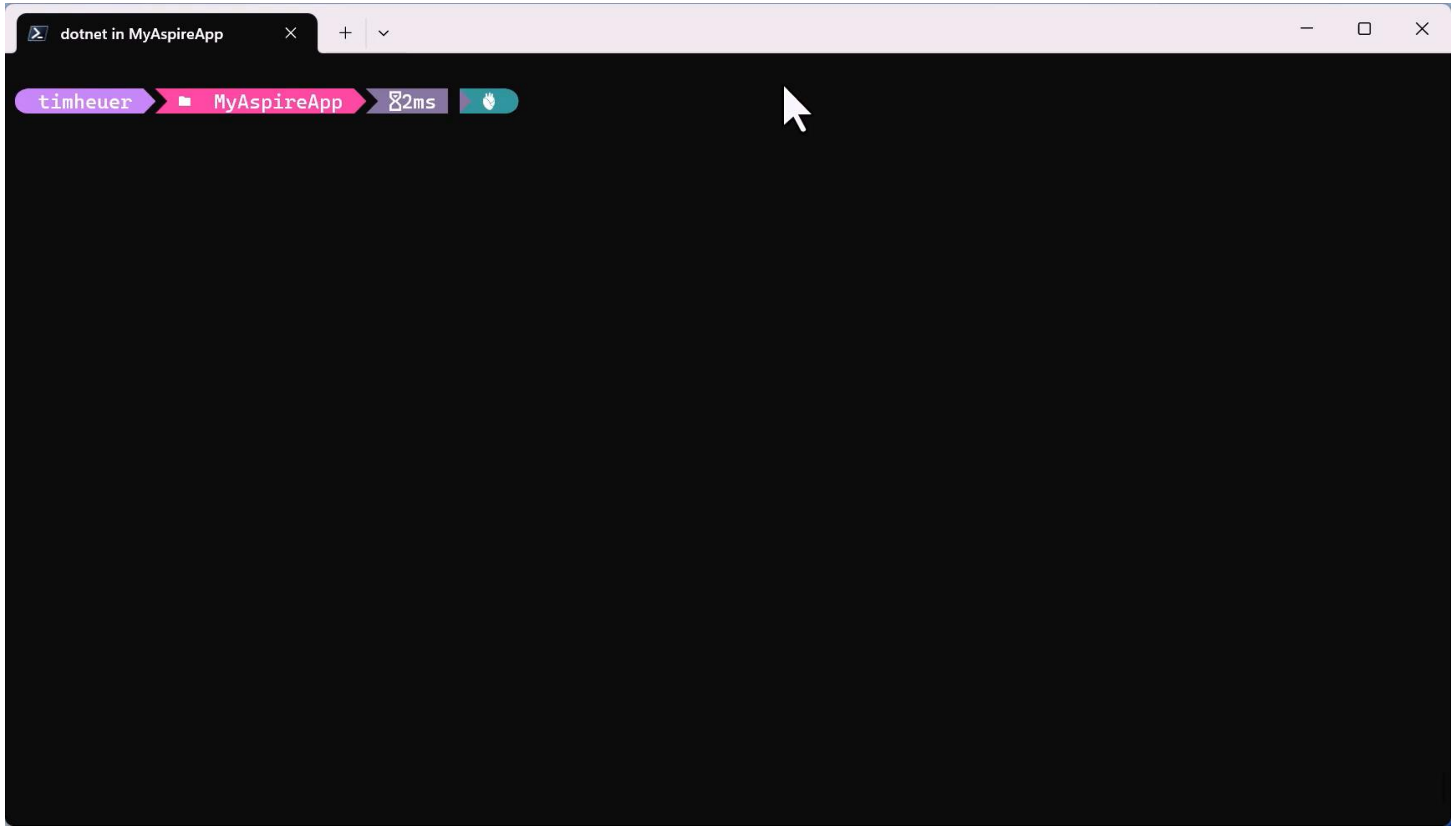
ServiceDefaults

- Defaults applied to all projects in the app:
  - Service discovery
  - Telemetry
  - Health Checks
- Exposing as a shared code in the project was our best idea so far

Deployment

- Aspire itself doesn’t natively provide a direct mechanism to deploy your applications to their final destination
- The application model can produce a manifest definition that describes all of these relationships and dependencies that tools can consume, augment, and build upon for deployment.
- More: *https://devblogs.microsoft.com/dotnet/introducing-dotnet-aspire-simplifying-cloud-native-development-with-dotnet-8/#deploying-a-net-aspire-application*

Video: Azure cli



```
timheuer  MyAspireApp 2ms
```

Sources

- Libraries work to support Cloud Native effort for .NET 8 · Issue #78518 · dotnet/runtime (github.com)
- Performance Improvements in .NET 8 - .NET Blog (microsoft.com)
- ASP.NET Core updates in .NET 8 Preview 4 - .NET Blog (microsoft.com)
- Understanding different metric APIs - .NET | Microsoft Learn
- Added missing metrics for .NET extensions. by IEvangelist · Pull Request #37875 · dotnet/docs (github.com)
- Fake It Till You Make It...To Production - .NET Blog (microsoft.com)
- Building resilient cloud services with .NET 8 - .NET Blog (microsoft.com)
- dotnet/extensions-samples (github.com)
- dotnet/aspire: .NET Aspire (github.com)
- Introducing .NET Aspire: Simplifying Cloud-Native Development with .NET 8 - .NET Blog (microsoft.com)
- .NET Aspire service discovery - .NET Aspire | Microsoft Learn