

Simplify the development of distributed applications with .NET Aspire

Thang Chung
November 23, 2024



Thang Chung

Technical Manager at NashTech VN
Microsoft Azure MVP

Creator of Vietnam Microservices Group on Facebook (>20k members):
<https://www.facebook.com/groups/645391349250568>

Experience: >18 years in software consult, design, development, and deployment software for outsourcing, product, and startup companies.

Expertise in cloud computing, cloud-native platform, serverless, and WebAssembly/WASI.

Blog: <https://dev.to/thangchung>

GitHub: <https://github.com/thangchung>

LinkedIn: <https://www.linkedin.com/in/thangchungatwork>

X (former Twitter): @thangchung

Agenda

01

.NET Aspire

02

What's new in .NET
Aspire 9?

03

Intelligent Apps
with .NET Aspire

01

.NET Aspire

9

Build anything with a unified platform

.NET



Cloud



Web



Desktop



Mobile



Gaming



IoT



AI



Visual Studio



Visual Studio
Code



CLI



GitHub
Copilot

Tools



Windows



Linux



macOS

Operating systems



NuGet



Github



.NET
Aspire



Components, tools,
library vendors

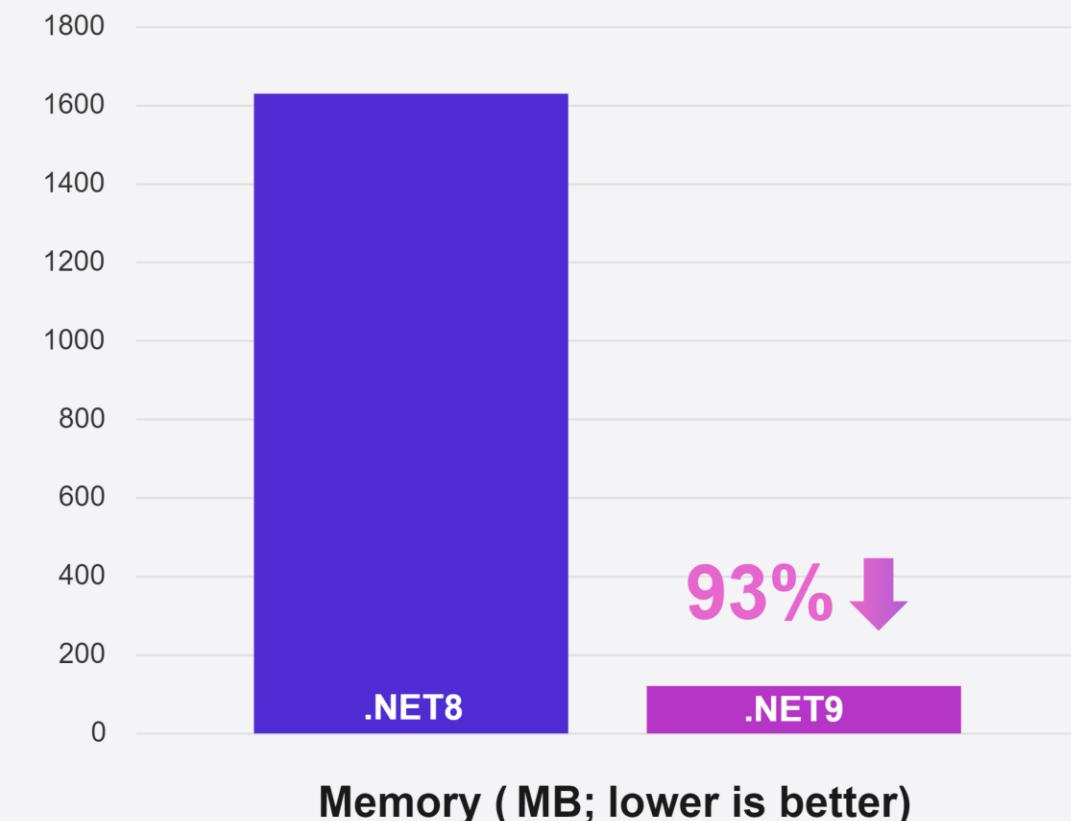
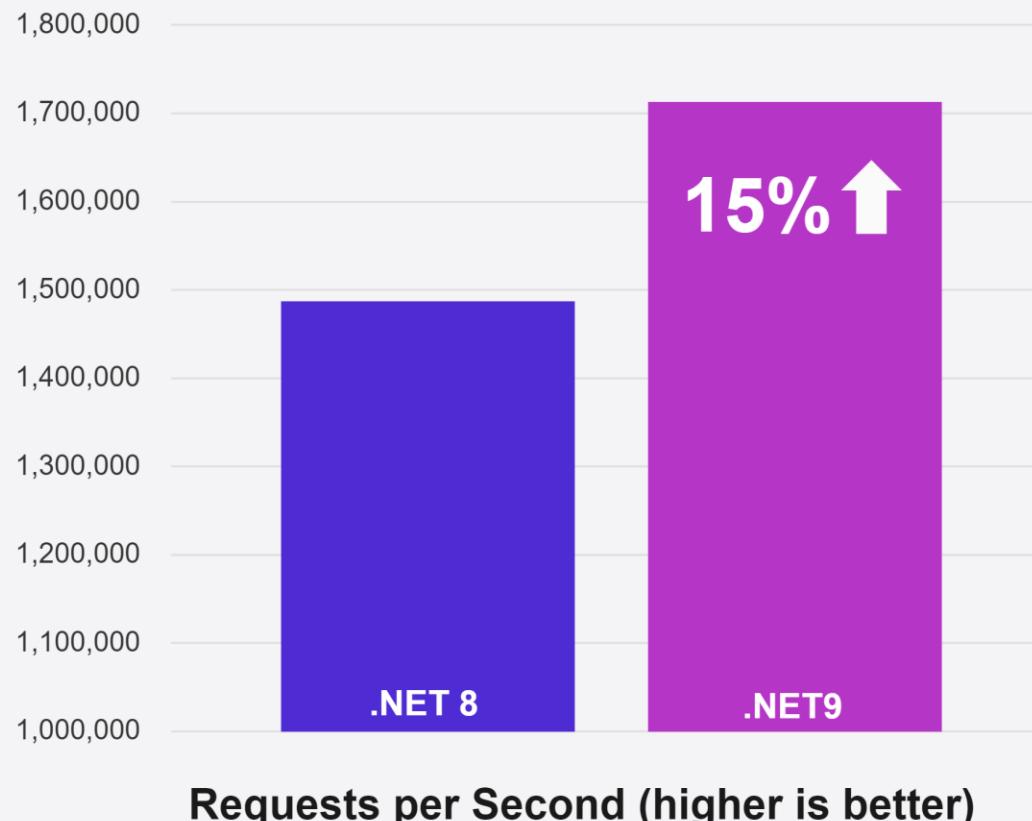
Ecosystem

.NET 9 Minimal API Performance

JSON Benchmarks



Intel Gold 56 cores (logical) Linux
Source: aka.ms/aspnet/benchmarks



Every App Needs



Observability



Resiliency



Scalability



Manageability

.NET gives you a lot in the box



Observability

Built in metrics with dimensions

DI integration for metrics

Better Logging support
(faster, can object serialization)

Enrichment

Redaction

Testing fakes for Logging & Metrics



Resiliency

New Polly based
resiliency packages

SignalR Stateful Reconnect



Scalability

AOT
(increased density)

Performance

Chiseled Ubuntu



Manageability

Certificate auto-rotation
support in Kestrel

It's Still Not Easy 😞



Complex



Getting Started



Choices



Paved Path



.NET Aspire

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



.NET Aspire

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

Smart Defaults

Developer Dashboard

Orchestration

Service Discovery

Integrations

Deployment



.NET Aspire

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

Smart Defaults

Developer Dashboard

Orchestration

Service Discovery

Integrations

Deployment

.NET Aspire Service Defaults



Observability



Resiliency



Health Checks

DEMO

.NET Aspire - Smart Defaults

Clone: <https://github.com/thangchung/practical-dotnet-aspire>



.NET Aspire

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

Smart Defaults

Developer Dashboard

Orchestration

Service Discovery

Integrations

Deployment



.NET Aspire

Developer Dashboard



Structured Logs



Metrics



Distributed Traces



Dependencies

The screenshot shows a dark-themed developer dashboard for the 'AspireApp' project. On the left, there's a sidebar with icons for 'Console', 'Structured', 'Traces', and 'Metrics'. The main area is titled 'Resources' and displays a table with the following data:

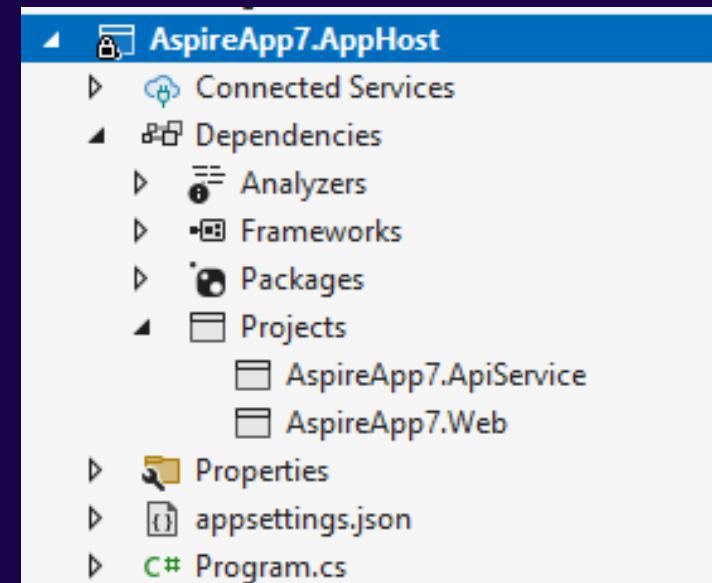
| Type | Name | State | Start time | Source | Endpoints | Logs | Details |
|-----------|-------------|---------|-------------|----------------------------|----------------------------|----------------------|----------------------|
| Container | cache | Running | 12:48:23 PM | redis:7.2.4 | tcp://localhost:53683 | View | View |
| Project | apiservice | Running | 12:48:23 PM | AspireApp.ApiService.cs... | +2 | View | View |
| Project | webfrontend | Running | 12:48:23 PM | AspireApp.Web.csproj | https://localhost:7234, +1 | View | View |

Integrating the Developer Dashboard

Standalone

```
docker run --rm -it `  
-p 18888:18888 `  
-p 4317:18889 -d `  
--name aspire-dashboard  
mcr.microsoft.com/dotnet/aspire-dashboard:8.0.0
```

.NET Aspire App Host





.NET Aspire

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

Smart Defaults

Developer Dashboard

Orchestration

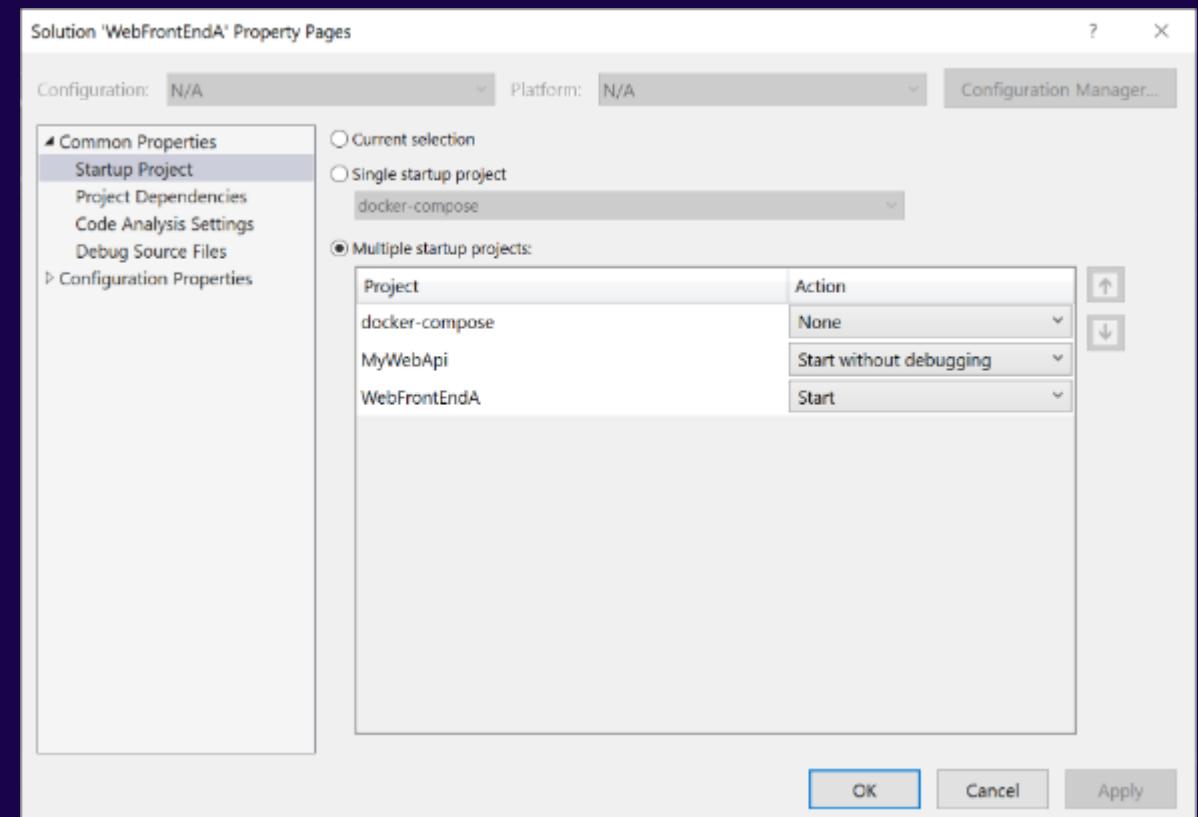
Service Discovery

Integrations

Deployment

Orchestration Before

```
{  
  "version": "0.2.0",  
  "compounds": [  
    {  
      "name": "Run all",  
      "configurations": [  
        "Run products",  
        "Run store",  
      ]  
    },  
    {  
      "name": "Run products",  
      "type": "dotnet",  
      "request": "launch",  
      "projectPath": "${workspaceFolder}\\Products\\Products.csproj"  
    },  
    {  
      "name": "Run store",  
      "type": "dotnet",  
      "request": "launch",  
      "projectPath": "${workspaceFolder}\\Store\\Store.csproj"  
    }  
  ]  
}
```





.NET Aspire

Orchestration

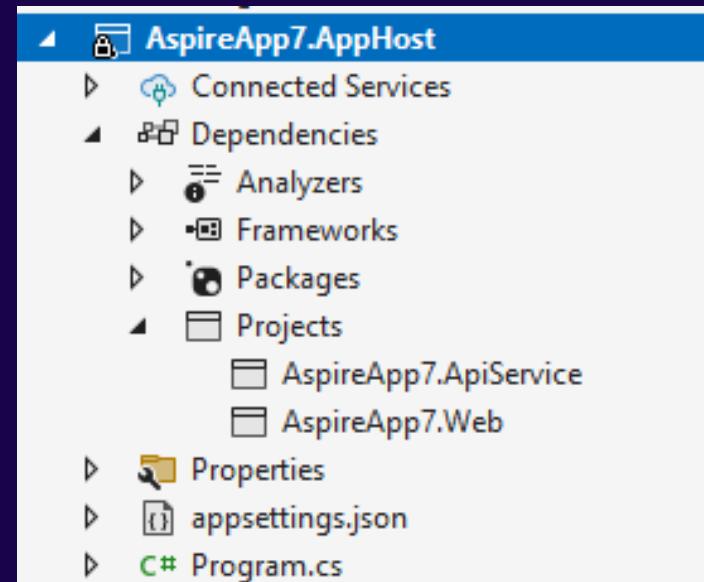
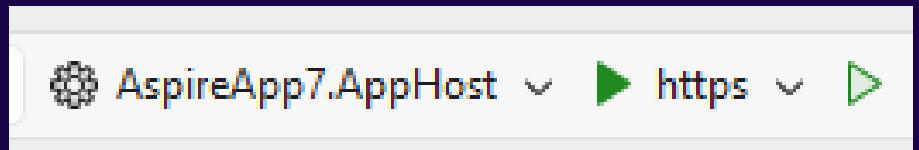
```
var builder = DistributedApplication.CreateBuilder(args);

builder.AddProject<Projects.ApiService>("apiservice");

builder.AddProject<Projects.Web>("webfrontend");

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

After with .NET Aspire



AspireApp7

Resources

| Type | Name | State | Start time | Source | Endpoints | Logs | Details |
|---------|-------------|---------|------------|-----------------------|--|----------------------|----------------------|
| Project | apiservice | Running | 2:34:57 PM | AspireApp7.ApiServ... | +2 | View | View |
| Project | webfrontend | Running | 2:34:57 PM | AspireApp7.Web.cs... | https://localhost:7107/ +1 | View | View |

Resources

Console

Structured

Traces

Metrics

DEMO

.NET Aspire - Orchestration

Clone: <https://github.com/thangchung/practical-dotnet-aspire>



.NET Aspire

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

Smart Defaults

Developer Dashboard

Orchestration

Service Discovery

Integrations

Deployment

Service Discovery Before

```
{  
  "Logging": {  
    "LogLevel": {  
      "Default": "Information",  
      "Microsoft.AspNetCore": "Warning"  
    }  
  },  
  "AllowedHosts": "*",  
  "ProductEndpoint": "http://localhost:5228",  
  "ProductEndpointHttps": "https://localhost:7130"  
}
```

```
builder.Services.AddHttpClient<ProductService>(c =>  
{  
  var url = builder.Configuration["ProductEndpoint"] ?? throw new  
  InvalidOperationException("ProductEndpoint is not set");  
  
  c.BaseAddress = new(url);  
});
```

Service Discovery

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

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

builder.AddProject<Projects.Web>("webfrontend")
    .WithReference(apiService);

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

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

There are still connection strings 😊

The screenshot shows the AspireApp7 application dashboard. On the left, there's a sidebar with icons for Resources, Console, Structured, Traces, and Metrics. The main area has a title bar "AspireApp7" and a "Resources" section. Below it, a modal window titled "Project: webfrontend" is open, showing various configuration settings.

| Name | Value |
|-----------------------------|---|
| OTEL_EXPORTER_OTLP_HEADERS | x-otlp-api-key=b273a95e153aa5a06b58cc1027daede3 |
| OTEL_EXPORTER_OTLP_PROTOCOL | grpc |
| OTEL_METRIC_EXPORT_INTERVAL | 1000 |
| OTEL_RESOURCE_ATTRIBUTES | service.instance.id=webfrontend-klumhn8 |
| OTEL_SERVICE_NAME | webfrontend |
| OTEL_TRACES_SAMPLER | always_on |
| services_apiservice_http_0 | http://localhost:5378 |
| services_apiservice_https_0 | https://localhost:7578 |

DEMO

.NET Aspire - Service Discovery

Clone: <https://github.com/thangchung/practical-dotnet-aspire>



.NET Aspire

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

Smart Defaults

Developer Dashboard

Orchestration

Service Discovery

Integrations

Deployment

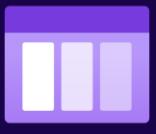


.NET Aspire

Integrations



RabbitMQ



Nuget Gallery | Home

← → ⌂ https://www.nuget.org

nuget Packages Upload Statistics Documentation Downloads Blog

Create .NET apps faster with NuGet

Search for packages...

398,762,018,894 package downloads

6,439,191 package versions

377,338 unique packages



.NET Aspire

Hosting Integration

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

var postgres = builder.AddPostgres("postgres").WithPgAdmin();
var db = postgres.AddDatabase("db");

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

var apiService = builder.AddProject<Projects.ApiService>("apiservice")
    .WithReference(db); ←

builder.AddProject<Projects.Web>("webfrontend")
    .WithReference(apiService)
    .WithReference(cache); ←

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

Automatically will pull down container image and start it in Docker or Podman!



.NET Aspire

Client Integration

```
var builder = WebApplication.CreateBuilder(args);
```

```
builder.AddServiceDefaults();  
builder.AddRedisOutputCache("cache");
```

```
builder.Services.AddRazorComponents()  
    .AddInteractiveServerComponents();
```

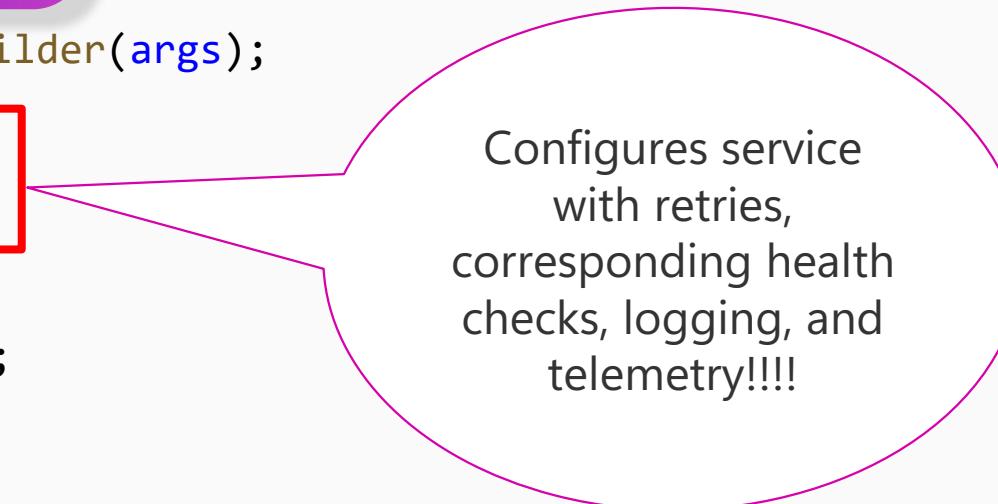
```
var app = builder.Build();
```

```
app.UseOutputCache();
```

```
app.MapRazorComponents<App>()  
    .AddInteractiveServerRenderMode();
```

```
app.MapDefaultEndpoints();
```

```
app.Run();
```



Configures service
with retries,
corresponding health
checks, logging, and
telemetry!!!!

DEMO

.NET Aspire - Integrations

Clone: <https://github.com/thangchung/practical-dotnet-aspire>



.NET Aspire

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

Smart Defaults

Developer Dashboard

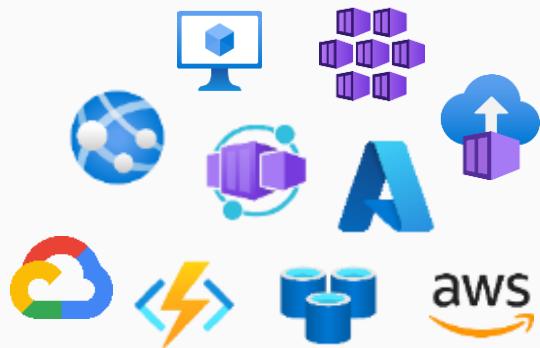
Orchestration

Service Discovery

Integrations

Deployment

Flexible Integrations & Deployment



How You Do
It Today!



AWS CDK



Azure Dev CLI



Visual Studio

The screenshot shows the Visual Studio IDE interface with the following details:

- Menu Bar:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Search Bar:** Search -> MyWeatherHub
- Toolbar:** Includes icons for Undo, Redo, Cut, Copy, Paste, Find, Replace, and others.
- Current Project:** AppHost
- Current View:** Debug -> Any CPU -> AppHost -> https
- Solution Explorer:** Shows the solution 'MyWeatherHub' (4 of 4 projects) with the 'AppHost' project selected. Inside 'AppHost', the 'Connected Services', 'Dependencies', 'Properties', 'appsettings.json', 'Program.cs', and 'ServiceDefaults' files are listed.
- Code Editor:** The 'Program.cs' file is open, showing C# code for building a distributed application using the DistributedApplication builder pattern. The code includes adding Redis cache, creating an API project, and a web project named 'myweatherhub'. The line 'var api = builder.AddProject<Projects.Api>("api")' is currently selected.
- Status Bar:** Shows 165%, No issues found, Ln: 6 Ch: 50 TABS CRLF, GitHub C..., Solution E..., Git Chang..., Notifications, Error List, Output, Ready, and a terminal window showing 'update-containers'.

```
1 var builder = DistributedApplication.CreateBuilder(args);
2
3 var cache = builder.AddRedis("cache")
4     .WithRedisCommander();
5
6 var api = builder.AddProject<Projects.Api>("api")
7     .WithReference(cache);
8
9 var web = builder.AddProject<Projects.MyWeatherHub>("myweatherhub")
10    .WithReference(api);
11
12
13 builder.Build().Run();
14
```

02

.NET Aspire – What's new in .NET Aspire 9?

9



Azure Functions
preview support



Improved Azure
Container Apps
integration



Persistent
Containers

9

Support for
.NET 8 & .NET 9



More secure
defaults



“WaitFor” resources
to spin up



Start/Stop Resources



Improved
Azure
Configuration



Simplified
acquisition

.NET Aspire 9.0



OpenAI
Integration



AWS Stable
Integrations



Visual Studio
& C# Dev Kit
enhancements

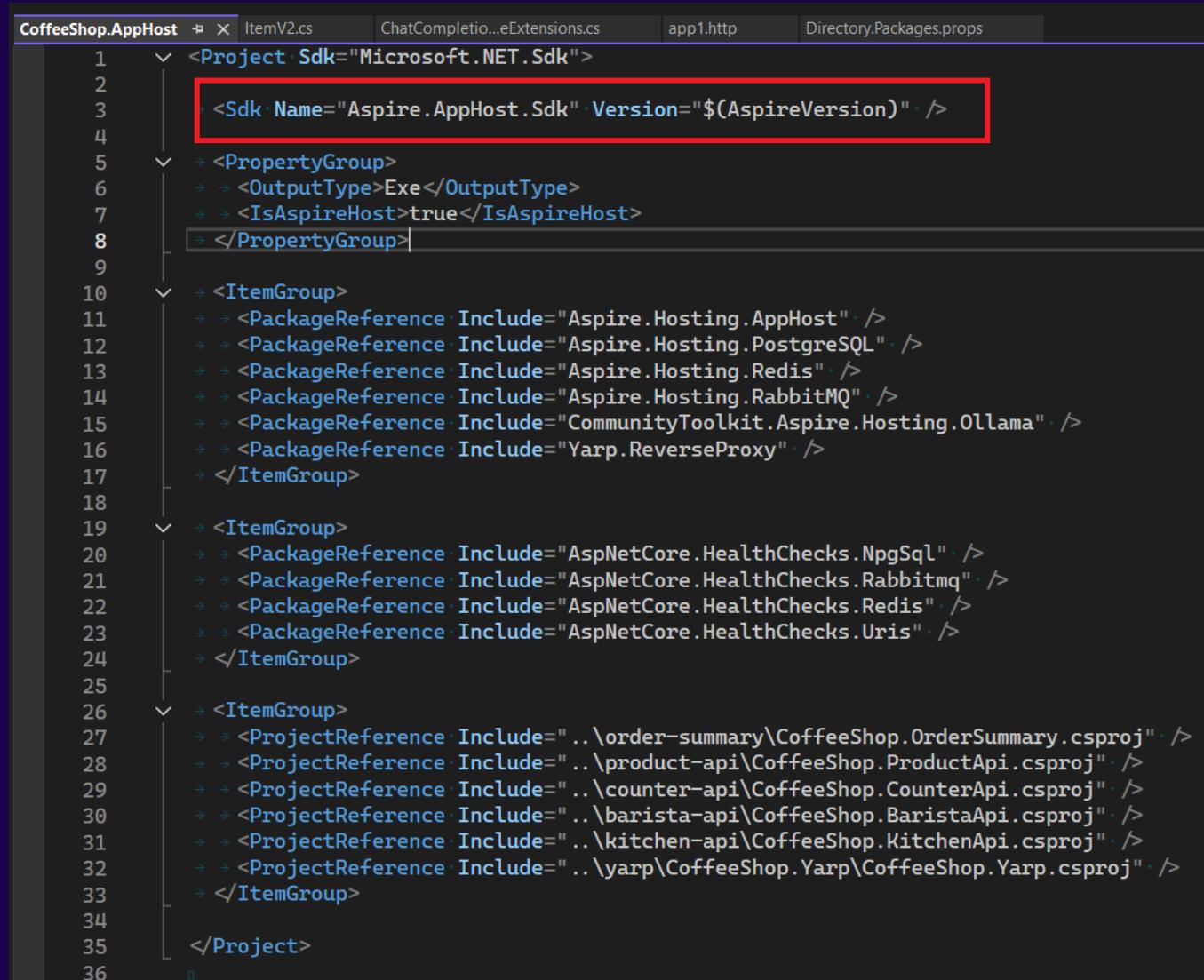


.NET Aspire
Community
Toolkit

Some important features I applied for practical-dotnet-aspire

Check it out...

No longer need a .NET workload



```
CoffeeShop.AppHost ItemV2.cs ChatCompleto...eExtensions.cs app1.http Directory.Packages.props
1 <Project Sdk="Microsoft.NET.Sdk">
2   <Sdk Name="Aspire.AppHost.Sdk" Version="$(AspireVersion)" />
3   <PropertyGroup>
4     <OutputType>Exe</OutputType>
5     <IsAspireHost>true</IsAspireHost>
6   </PropertyGroup>
7
8   <ItemGroup>
9     <PackageReference Include="Aspire.Hosting.AppHost" />
10    <PackageReference Include="Aspire.Hosting.PostgreSQL" />
11    <PackageReference Include="Aspire.Hosting.Redis" />
12    <PackageReference Include="Aspire.Hosting.RabbitMQ" />
13    <PackageReference Include="CommunityToolkit.Aspire.Hosting.Ollama" />
14    <PackageReference Include="Yarp.ReverseProxy" />
15  </ItemGroup>
16
17  <ItemGroup>
18    <PackageReference Include="AspNetCore.HealthChecks.Npgsql" />
19    <PackageReference Include="AspNetCore.HealthChecks.Rabbitmq" />
20    <PackageReference Include="AspNetCore.HealthChecks.Redis" />
21    <PackageReference Include="AspNetCore.HealthChecks.Uris" />
22  </ItemGroup>
23
24  <ItemGroup>
25    <ProjectReference Include="..\order-summary\CoffeeShop.OrderSummary.csproj" />
26    <ProjectReference Include="..\product-api\CoffeeShop.ProductApi.csproj" />
27    <ProjectReference Include="..\counter-api\CoffeeShop.CounterApi.csproj" />
28    <ProjectReference Include="..\barista-api\CoffeeShop.BaristaApi.csproj" />
29    <ProjectReference Include="..\kitchen-api\CoffeeShop.KitchenApi.csproj" />
30    <ProjectReference Include="..\yarp\CoffeeShop.Yarp\CoffeeShop.Yarp.csproj" />
31  </ItemGroup>
32
33  </Project>
34
35
36
```

Waiting for dependencies

```
● ● ●

using CoffeeShop.AppHost;

var builder = DistributedApplication.CreateBuilder(args);

var postgresQL = builder.AddPostgres("postgresQL")
    .WithPgWeb();
var postgres = postgresQL.AddDatabase("postgres");

var redis = builder.AddRedis("redis")
    .WithRedisCommander();

var rabbitmq = builder.AddRabbitMQ("rabbitmq")
    .WithManagementPlugin();

var productApi = builder.AddProject<Projects.CoffeeShop_ProductApi>("product-
api")           .WithReference(postgres).WaitFor(postgres);

var counterApi = builder.AddProject<Projects.CoffeeShop_CounterApi>("counter-
api")           .WithReference(productApi).WaitFor(productApi)
    .WithReference(rabbitmq).WaitFor(rabbitmq);

builder.AddProject<Projects.CoffeeShop_BaristaApi>("barista-api")
    .WithReference(rabbitmq).WaitFor(rabbitmq);

builder.AddProject<Projects.CoffeeShop_KitchenApi>("kitchen-api")
    .WithReference(rabbitmq).WaitFor(rabbitmq);

builder.AddProject<Projects.CoffeeShop_Yarp>("yarp")
    .WithReference(productApi).WaitFor(productApi)
    .WithReference(counterApi).WaitFor(counterApi);

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

Resource health checks



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

var catalogApi = builder.AddContainer("catalog-api", "catalog-
api")
    .WithHttpEndpoint(targetPort: 8080)
    .WithHttpHealthCheck("/health");

builder.AddProject<Projects.WebApplication1>("store")
    .WithReference(catalogApi.GetEndpoint("http"))
    .WaitFor(catalogApi);

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

| Type | Name | State |
|---------|-------------|---------------------|
| Project | catalog-api | Running (Unhealthy) |
| Project | store | Waiting |

Persistent containers

```
● ● ●  
using CoffeeShop.AppHost;  
  
var builder = DistributedApplication.CreateBuilder(args);  
  
var postgresQL = builder.AddPostgres("postgresQL")  
    .WithLifetime(ContainerLifetime.Persistent)  
    .WithPgWeb();  
var postgres = postgresQL.AddDatabase("postgres");  
  
var redis = builder.AddRedis("redis")  
    .WithLifetime(ContainerLifetime.Persistent)  
    .WithRedisCommander();  
  
var rabbitmq = builder.AddRabbitMQ("rabbitmq")  
    .WithLifetime(ContainerLifetime.Persistent)  
    .WithManagementPlugin();
```

| Type | Name | State |
|-----------|----------|---------|
| Container | rabbit ⚡ | Running |
| Project | api | Running |

Manage resource lifecycle

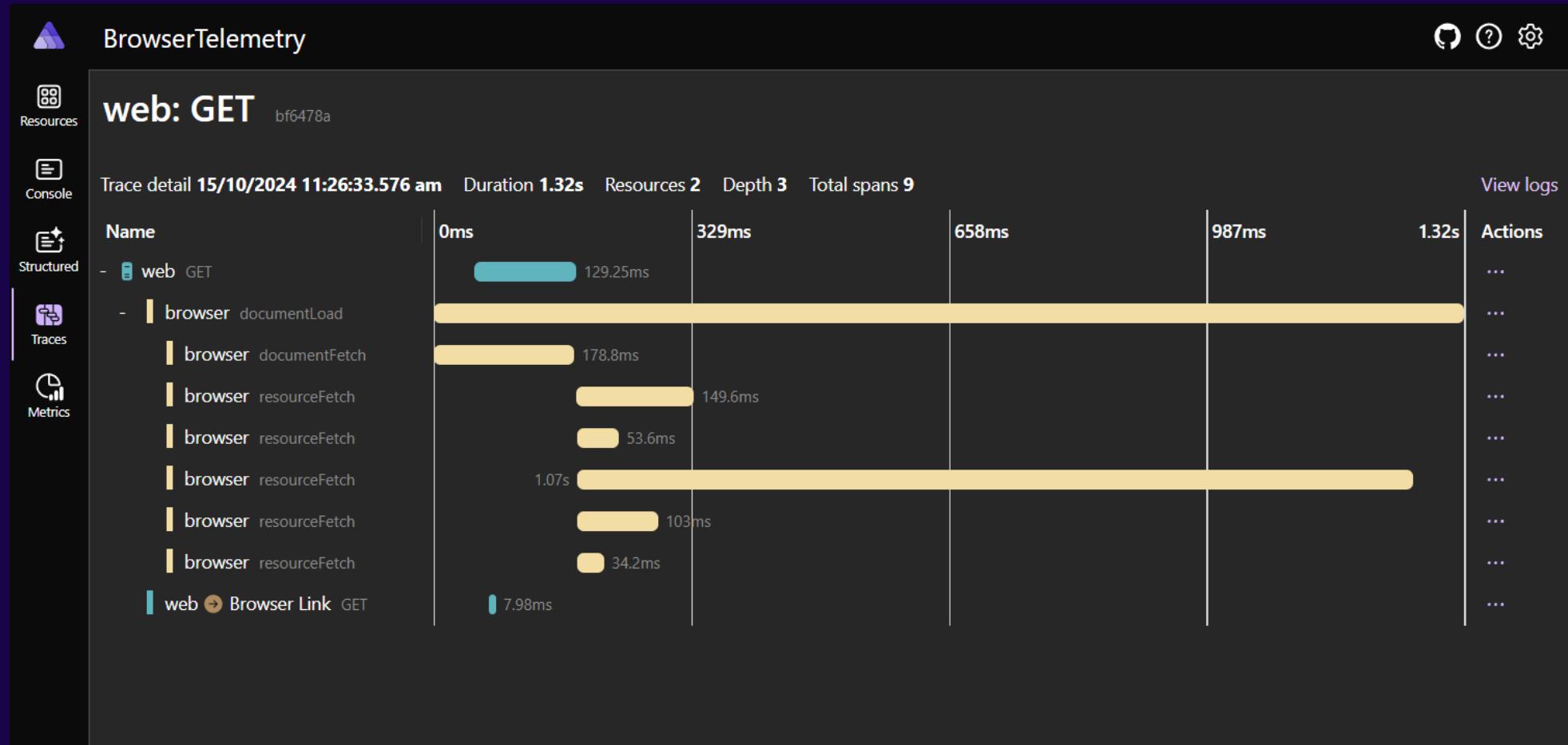
AspireSample

cache "Stop" succeeded

| Type | Name | State | Start time | Source | Endpoints | Actions |
|-----------|-------------|-----------|------------|--------------------------------|---|---------|
| Container | cache | ⚠ Exited | 9:51:25 AM | docker.io/library/redis:7.4 | - | |
| Project | apiservice | ✓ Running | 9:51:22 AM | AspireSample.ApiService.csproj | https://localhost:7352/weatherfore... | |
| Project | webfrontend | ✓ Running | 9:51:25 AM | AspireSample.Web.csproj | https://localhost:7163 | |

Browser telemetry support

The dashboard supports OpenTelemetry Protocol (OTLP) over HTTP and cross-origin resource sharing (CORS). These features unlock the ability to send OpenTelemetry from browser apps to the .NET Aspire dashboard.



OpenAI (Preview)



```
// https://www.nuget.org/packages/Aspire.OpenAI/9.0.0  
  
builder.AddOpenAIClientFromConfiguration("openai");
```

03

.NET Aspire - Intelligent Apps with .NET Aspire

9

.NET Application

Leveraging AI

Microsoft.Extensions.AI (Middleware)

AI, Data Tools, & Services

UI Components
(Smart Components)

AI SDKs
(OllamaSharp, OpenAI, Azure AI Inference)

Libraries
(Semantic Kernel, AutoGen)

Vector Store SDKs

Developer Tools (Promptly)

Microsoft.Extensions.AI.Abstractions

Microsoft.Extensions.VectorData

Sample: eShopSupport

<https://github.com/dotnet/eShopSupport>



Public site

The screenshot shows a mobile application window with a dark grey header bar containing a back arrow icon and a close button (an 'X'). Below the header is a purple navigation bar with the text 'Outdoor Activities eCommerce Site' on the left and 'Customer' on the right. The main content area has a white background and features a section titled 'My support enquiries'. This section includes a table with four columns: 'ID', 'Date', 'Type', and 'Product'. There are two rows of data: one for a 'Returns' enquiry about a 'Bäevolk 4-person tent' and another for a 'Question' about a 'TCX Kayak 2000'. Each row has a 'View >' button on the right. At the bottom of the screen is a purple button with the text '+ Create new enquiry'. A mouse cursor is visible at the bottom right corner of the window.

| ID | Date | Type | Product | |
|-------|-----------|----------|-----------------------|---------------------------|
| 23433 | Today | Returns | Bäevolk 4-person tent | View > |
| 23419 | Fri 5 May | Question | TCX Kayak 2000 | View > |

+ Create new enquiry

Staff web UI

The diagram illustrates a Staff web UI interface for managing tickets. The main view shows a list of tickets with columns for #, Date, Type, Title, and Satisfaction. Annotations highlight specific features: 'Summarization' points to the top of the list; 'Classification' points to the 'Type' column; 'Semantic search' points to the search bar; and 'Sentiment score' points to the satisfaction levels (OK, Great, Bad).

| # | Date | Type | Title | Satisfaction |
|-------|-----------|-----------|--------------------------------------|--------------|
| 23433 | Today | Returns | Return of Bäevolk 4-person tent | OK |
| 23419 | Today | Question | Asks how to update kayak firmware | Great |
| 23414 | Yesterday | Question | Asks what other colors are available | OK |
| 23413 | Yesterday | Complaint | T-Shirt printing inadequate quality | Bad |
| 23407 | Yesterday | Returns | Delivery late; return initiated | OK |
| 23399 | ... | ... | ... | OK |
| 23287 | ... | ... | ... | Bad |
| 23190 | ... | ... | ... | OK |

Staff web UI

The screenshot displays a staff web interface with several AI-powered features highlighted by purple callout bubbles:

- Classification**: Points to the top navigation bar.
- Summarization**: Points to the summary section below the search bar.
- Typeahead (content generation)**: Points to the customer message input field.
- Semantic search**: Points to the search bar.
- Q&A (RAG)**: Points to the AI assistant panel.
- Citation**: Points to the AI-generated response at the bottom right.

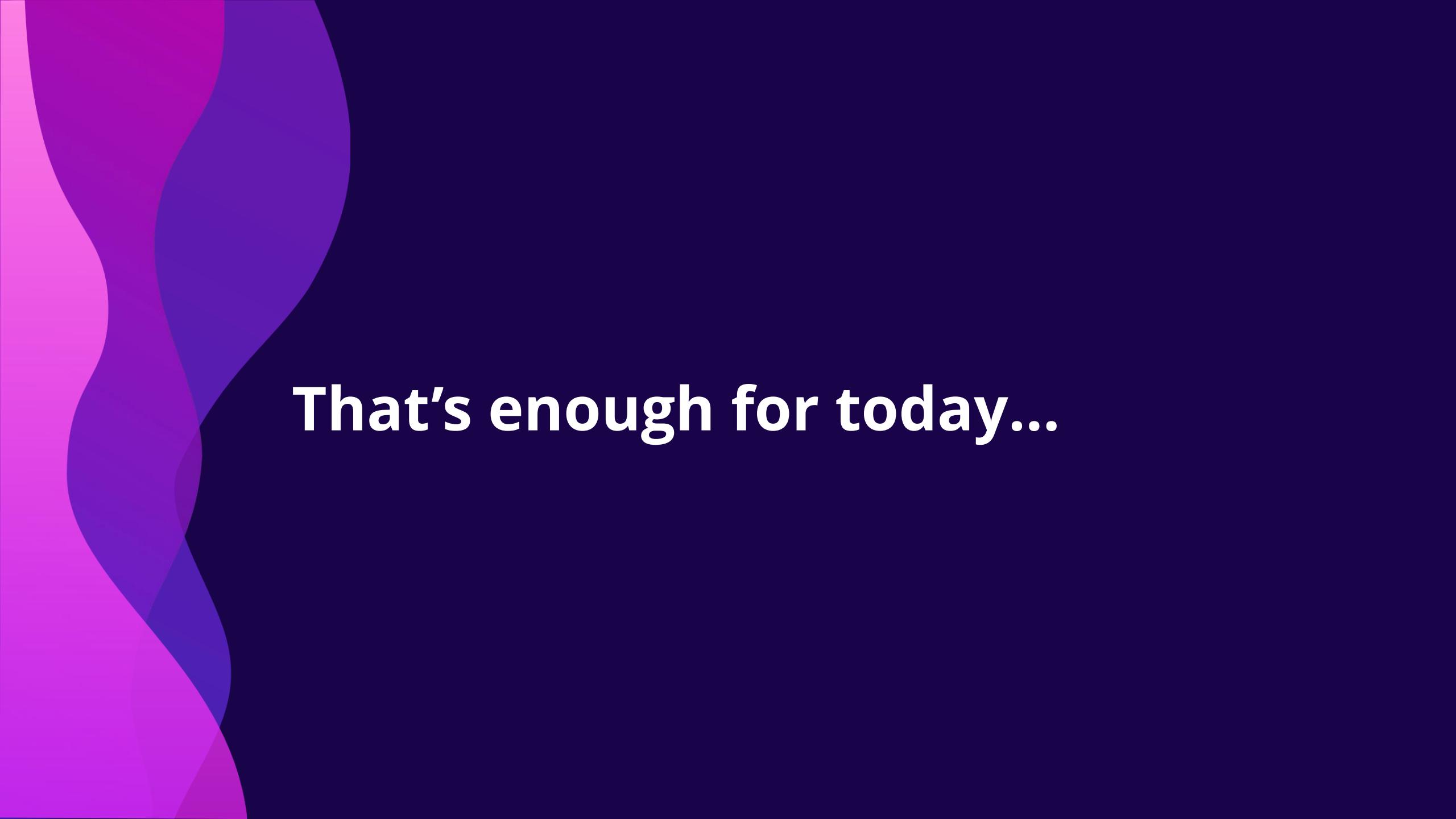
UI Elements and Data:

- Header:** Shows a timestamp (2024-01-19), a user icon labeled "Staff", and a close button (X).
- Search Bar:** Includes dropdowns for "Case type" (Question), "Status" (Open), and "Product" (TCX Kayak 2000), along with a search icon.
- Summary Section:** Contains a yellow box with the text: "Summary: Customer wishes to perform backflips and asks how to update kayak firmware."
- Customer Message:** A message from "Customer" saying: "OK yeah but what about blah blah blah because my nan says blah blah and my friend Dave says you can but blah blah".
- AI Assistant Response:** A blue box containing: "Thanks for your enquiry! The TCX kayak supports blah blah blah." Below it, a gray box contains: "Page 44 of the manual states that backflips are supported as of firmware version 3.17." and a link: "Link: Manual.pdf page 44".
- Bottom Response:** A white box with the text: "We have looked into this, and found that the TCX Kayak model 2000 does support backflips as of v3.17. To upgrade, please see http://...".
- Bottom Input:** A gray box with the placeholder text: "Ask the AI assistant...".

DEMO

.NET Aspire – Microsoft.Extensions.AI (Seed data and Sematic Search)

Clone: <https://github.com/thangchung/practical-dotnet-aspire>



That's enough for today...



Get .NET 9



Download .NET 9
aka.ms/get-dotnet-9

More on

[https://learn.microsoft.com/en-
us/dotnet/aspire/whats-new/dotnet-aspire-](https://learn.microsoft.com/en-us/dotnet/aspire/whats-new/dotnet-aspire-)



.NET Aspire

Learning Resources

aka.ms/letslearn/dotnet/aspire

.NET Aspire Videos

aka.ms/aspire/videos

Documentation

aka.ms/dotnet-aspire

Engage with team on GitHub

github.com/dotnet/aspire

Thang Chung
NashTech



@thangchung



<https://www.linkedin.com/in/thangchungatwork>

Thank you

