.NET Aspire

Александр Гольдебаев

Cloud-native разработка

```yaml
version: '3.9'

networks:
  some-net:
    driver: bridge

services:
  api:
    container_name: api
    image: borntowhine/someapp_api
    build:
      context: .
      dockerfile: SomeApp.API/Dockerfile
    networks:
      - some-net
    depends_on:
      - db

  app:
    container_name: app
    image: borntowhine/someapp_main
    build:
      context: someapp-frontend/
      dockerfile: SomeApp.APP/Dockerfile
    networks:
      - some-net
    depends_on:
      - proxy
```

Project Tye

tye.yml

```yaml
name: tyeapp
registry: mydockerregistry
services:
- name: frontend
  project: Path/to/frontend.csproj
  bindings:
    - port: 7000
- name: backend
  project: Path/to/backend.csproj
  bindings:
    - port: 8000
- name: redis-cache
  image: redis:latest
```

Project Tye

tye.yml

```yaml
name: tyeapp
registry: mydockerregistry
services:
- name: frontend
  project: Path/to/frontend.csproj
  bindings:
    - port: 7000
- name: backend
  project: Path/to/backend.csproj
  bindings:
    - port: 8000
- name: redis-cache
  image: redis:latest
```

Program.cs

```csharp
string backendUri = builder.Configuration
    .GetServiceUri("backend");
```

Project Tye

Tye Dashboard

Home

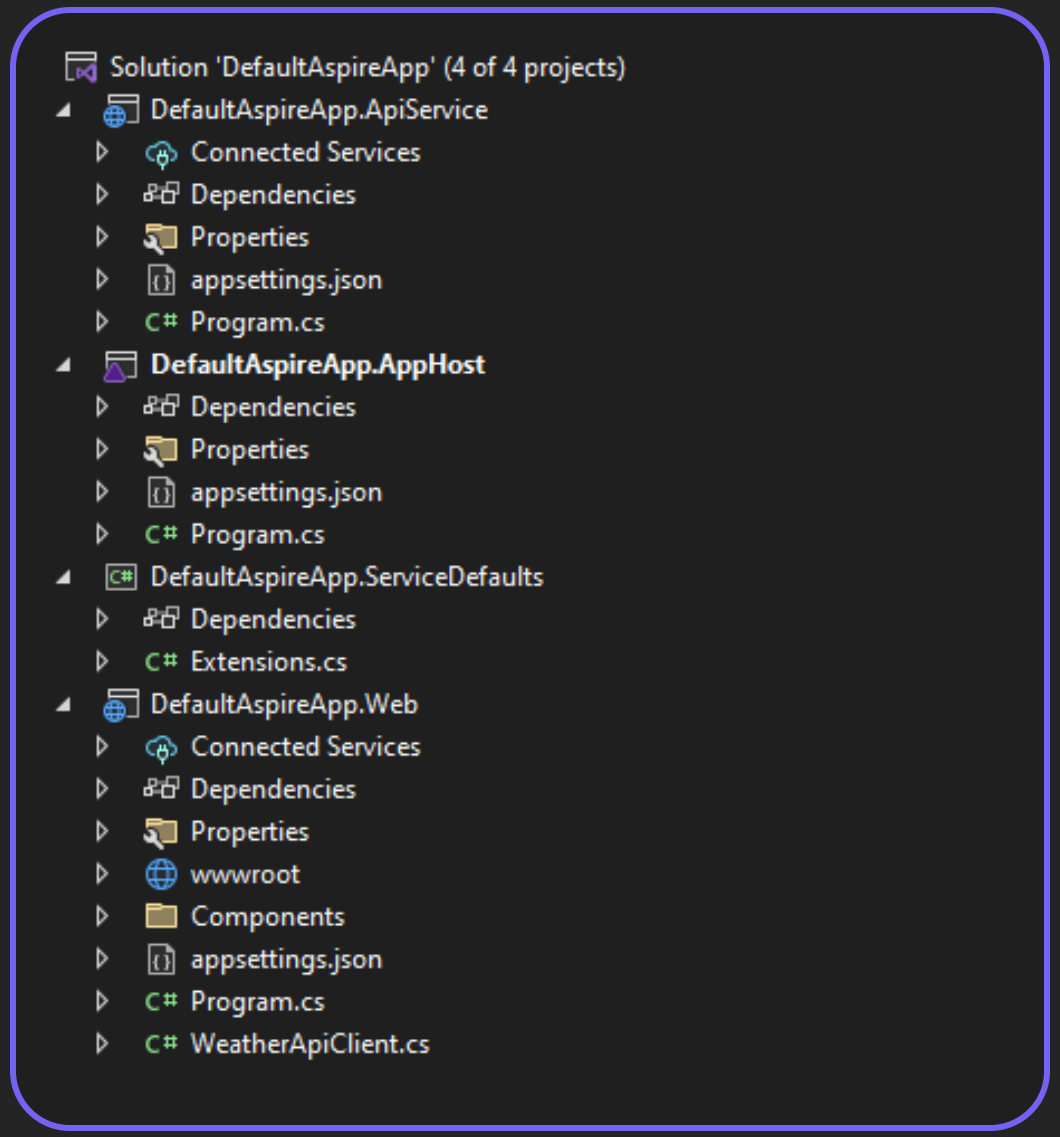
Services

Name	Type	Source	Bindings	Replicas	Restarts	Logs
backend	Project	C:\Source\nishanil\tye-demos\microservices\backend\backend.csproj	http://localhost:50653 https://localhost:50654	1/1	0	View
frontend	Project	C:\Source\nishanil\tye-demos\microservices\frontend\frontend.csproj	http://localhost:50655 https://localhost:50656	1/1	0	View
redis	Container	redis	tcp://localhost:6379	1/1	0	View
redis-cli	Container	redis	none	1/1	0	View

Внедряем Aspire

1. Внедрение в уже существующий
   - .NET Aspire Orchestration Support (for VS)
2. Создание нового проекта
   - Empty Application
   - Starter Application

.NET Aspire Starter Application



AppHost

Program.cs

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

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

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

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

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

AppHost

AppHost.csproj

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

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <IsAspireHost>true</IsAspireHost>
  </PropertyGroup>

  <ItemGroup>
    <ProjectReference Include="..\AspireSample.ApiService\AspireSample.ApiService.csproj" />
    <ProjectReference Include="..\AspireSample.Web\AspireSample.Web.csproj" />
  </ItemGroup>

  <ItemGroup>
    <PackageReference Include="Aspire.Hosting" Version="8.0.0-preview.2.23619.3" />
  </ItemGroup>

</Project>
```

ApiService

Program.cs

```
var builder = WebApplication.CreateBuilder(args);
builder.AddServiceDefaults();
builder.Services.AddProblemDetails();

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

app.MapGet("/weatherforecast", () =>
{
    var forecast = /* */
    return forecast;
});

app.MapDefaultEndpoints();

app.Run();
```

ApiService

```
builder.AddServiceDefaults();
```

- OpenTelemetry metrics and tracing.
- Add default Health Check endpoints.
- Add Service Discovery functionality.
- Configures HttpClient to work with service discovery.

ServiceDefaults

ServiceDefaults.csproj

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Library</OutputType>
    <TargetFramework>net8.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <IsAspireSharedProject>true</IsAspireSharedProject>
  </PropertyGroup>

  <ItemGroup>
    <FrameworkReference Include="Microsoft.AspNetCore.App" />

    <PackageReference Include="Microsoft.Extensions.Http.Resilience" Version="8.0.0" />
    <PackageReference Include="Microsoft.Extensions.ServiceDiscovery" Version="8.0.0-preview.1.23557.2" />
    <PackageReference Include="OpenTelemetry.Exporter.OpenTelemetryProtocol" Version="1.7.0-alpha.1" />
    <PackageReference Include="OpenTelemetry.Extensions.Hosting" Version="1.7.0-alpha.1" />
    <PackageReference Include="OpenTelemetry.Instrumentation.AspNetCore" Version="1.6.0-beta.2" />
    <PackageReference Include="OpenTelemetry.Instrumentation.GrpcNetClient" Version="1.6.0-beta.2" />
    <PackageReference Include="OpenTelemetry.Instrumentation.Http" Version="1.6.0-beta.2" />
    <PackageReference Include="OpenTelemetry.Instrumentation.Runtime" Version="1.5.1" />
  </ItemGroup>
</Project>
```

ServiceDefaults

Extensions.cs

```
public static IHostApplicationBuilder AddServiceDefaults(this IHostApplicationBuilder builder)
{
    builder.ConfigureOpenTelemetry();
    builder.AddDefaultHealthChecks();
    builder.Services.AddServiceDiscovery();
    builder.Services.ConfigureHttpClientDefaults(http =>
    {
        http.AddStandardResilienceHandler();

        http.UseServiceDiscovery();
    });

    return builder;
}
```

Web

Program.cs

```
using DefaultAspireApp.Web;
using DefaultAspireApp.Web.Components;

var builder = WebApplication.CreateBuilder(args);

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

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

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

var app = builder.Build();

/* app.SomethingManipulation() */
```

Web

Program.cs

```
builder.AddServiceDefaults();

builder.AddRedisOutputCache("cache");

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

Конфигурирует проект

Web

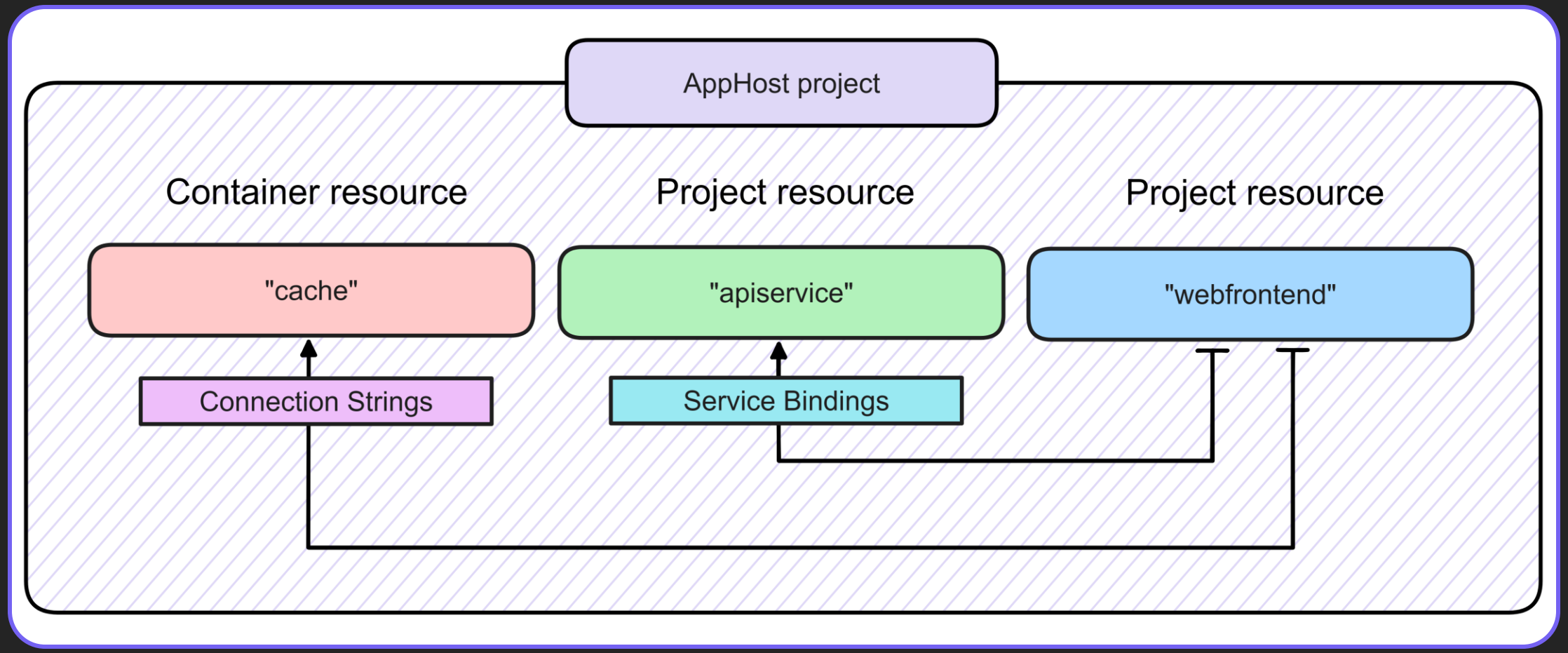
Program.cs

```
builder.AddServiceDefaults();

builder.AddRedisOutputCache("cache");

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

Конфигурирует проект

Используем название сервиса

Оркестрация



Оркестрация

Метод	Тип добавляемого ресурса	Описание

Оркестрация

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект

```
AddProject (this IDistributedApplicationBuilder builder,
    string name, string projectPath);
```

```
AddProject<TProject> (this IDistributedApplicationBuilder builder, string name)
where TProject : Aspire.Hosting.IServiceMetadata, new();
```

Оркестрация

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект

```
builder.AddProject<Projects.SomeProject_Web>("webfrontend");
```

Оркестрация

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект
AddContainer	ContainerResource	Образ контейнера, например образ Docker

```
AddContainer (this IDistributedApplicationBuilder builder, string name, string image);
```

```
AddContainer (this IDistributedApplicationBuilder builder,
    string name, string image, string tag);
```

Оркестрация

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект
AddContainer	ContainerResource	Образ контейнера, например образ Docker

```
builder.AddContainer("database", "postgres", "alpine:3.19");
```

Оркестрация

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект
AddContainer	ContainerResource	Образ контейнера, например образ Docker
AddExecutable	ExecutableResource	Исполняемый файл

```
AddExecutable (this IDistributedApplicationBuilder builder,
    string name, string command, string workingDirectory, params string[]? args);
```

Зависимости

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

builder.AddProject<Projects.AspireApp_Web>("webfrontend")
      .WithReference(cache);
```

Метод	Создаваемая переменная окружения

Зависимости

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

builder.AddProject<Projects.AspireApp_Web>("webfrontend")
      .WithReference(cache);
```

Метод	Создаваемая переменная окружения
WithReference(cache)	ConnectionStrings__cache="localhost:6379"

Зависимости

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

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

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

Метод	Создаваемая переменная окружения
WithReference(cache)	ConnectionStrings__cache="localhost:6379"

Зависимости

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

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

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

Метод	Создаваемая переменная окружения
WithReference(cache)	ConnectionStrings__cache="localhost:6379"
WithReference(apiservice)	services__apiservice__0="http://_http.localhost:8034"
	services__apiservice__1="http://localhost:8034"

Компоненты

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

Компоненты

1. Схема для конфигурации
2. Обеспечение высокой доступности
3. Отслеживание состояния
4. Современные абстракции
5. Внедрение в DI контейнер

Aspire.Npgsql

1. Установить пакет Aspire.Npgsql

Aspire.Npgsql

1. Установить пакет Aspire.Npgsql
2. Зарегестрировать NpgsqlDataSource в SomeProj

SomeProj

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

builder.AddNpgsqlDataSource("customers");
builder.AddServiceDefaults();

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

Aspire.Npgsql

1. Установить пакет Aspire.Npgsql
2. Зарегестрировать NpgsqlDataSource в SomeProj
3. Добавить в AppHost ссылку на SomeProj

Aspire.Npgsql

1. Установить пакет Aspire.Npgsql
2. Зарегестрировать NpgsqlDataSource в SomeProj
3. Добавить в AppHost ссылку на SomeProj
4. Сконфигурировать AppHost

AppHost

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

var database = builder.AddPostgres("postgresql")
    .AddDatabase("customers");

builder.AddProject<Projects.SomeProj>("someprojname")
    .WithReference(database)

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

Aspire.Npgsql

1. Установить пакет Aspire.Npgsql
2. Зарегестрировать NpgsqlDataSource в SomeProj
3. Добавить в AppHost ссылку на SomeProj
4. Сконфигурировать AppHost
5. Использовать

Конфигурация компонентов

appsettings.json

```
{
  "Aspire":  {
    "Npgsql":  {
      "HealthChecks": false,
      "Tracing": false
    }
  }
}
```

Program.cs

```
builder.AddNpgsqlDataSource(
    "PostgreSqlConnection",
    static settings => settings.HealthChecks = false);
```

Внедрение зависимостей

```
public class ExampleService(NpgsqlDataSource dataSource)
{
    // Work with database
}
```

Внедрение зависимостей с ключом

```
builder.AddKeyedNpgsqlDataSource(
    "PostgreSqlConnection",
    static settings => settings.HealthChecks = false);
```

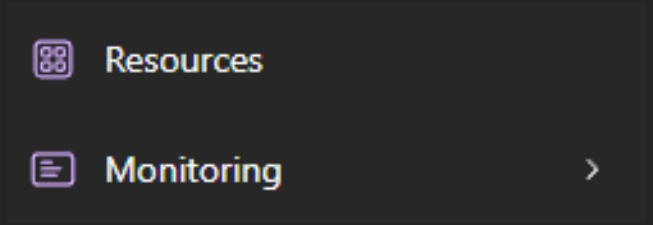
```
public class ExampleService(
    [FromKeyedServices("PostgreSqlConnection")] NpgsqlDataSource dataSource)
{
    // Work with database
}
```

Итого

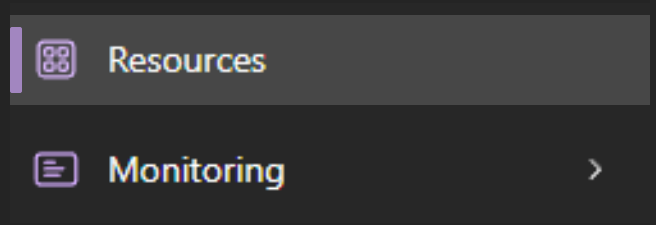
1. Cloud-native из коробки
2. Компоненты – сборники best practice
3. Одна точка входа
4. Удобная конфигурация и отладка

ВСЁ!

Dashboard



Dashboard

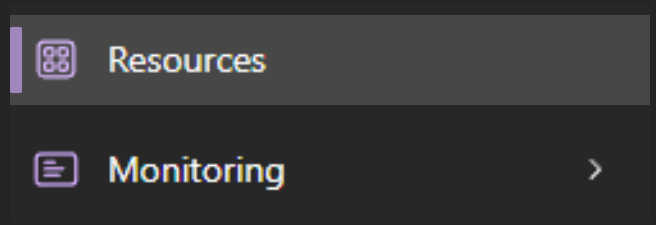


Resources

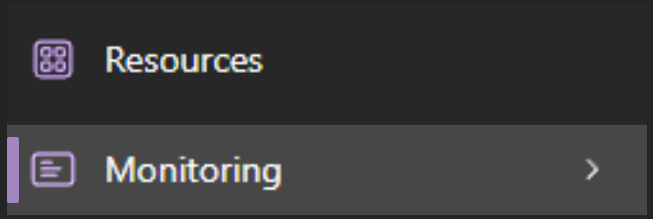
Filter...

Type	Name	State	Start Time	Source	Endpoints	Enviro...	Logs
Container	cache e01b41d0	Running	11-Feb-24 17:34:33	redis:latest 6379	None	View	View
Project	apiservice 95944	Running 1	11-Feb-24 17:34:33	D:\CSharp\Rofls\DefaultA...pireApp.ApiS...	http://localhost:5325/we...	View	View
Project	webfrontend 91192	Running	11-Feb-24 17:34:33	D:\CSharp\Rofls\DefaultA...faultAspireA...	http://localhost:5127	View	View

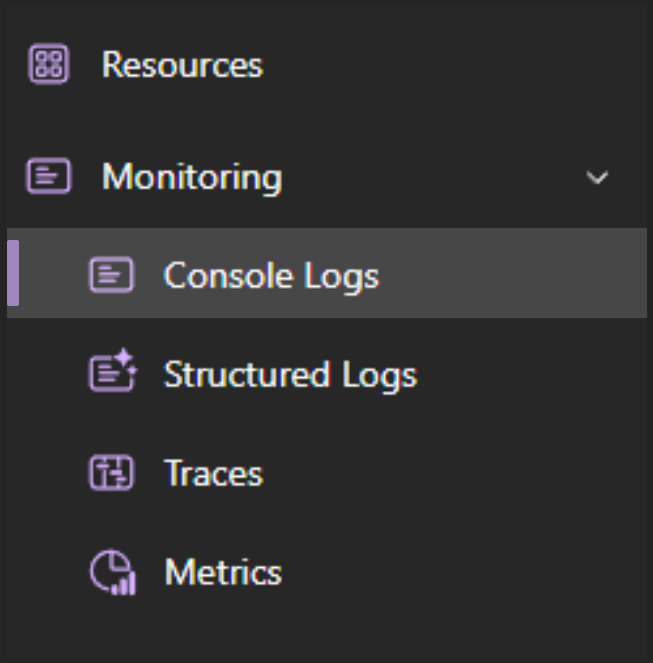
Dashboard



Dashboard



Dashboard


Resources
Monitoring
Console Logs
Structured Logs
Traces
Metrics

Console Logs

apiservice

Watching logs...

```
2024-02-11T17:34:36.2832582 info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:59721
2024-02-11T17:34:36.4676061 info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
2024-02-11T17:34:36.4769898 info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
2024-02-11T17:34:36.4802849 info: Microsoft.Hosting.Lifetime[0]
      Content root path: D:\CSharp\Rofls\DefaultAspireApp\DefaultAspireApp.ApiService
```

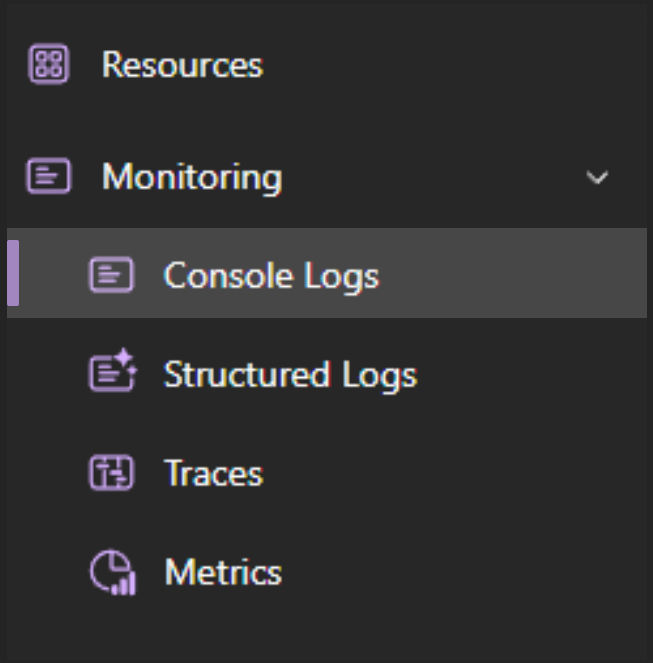
Console Logs

cache

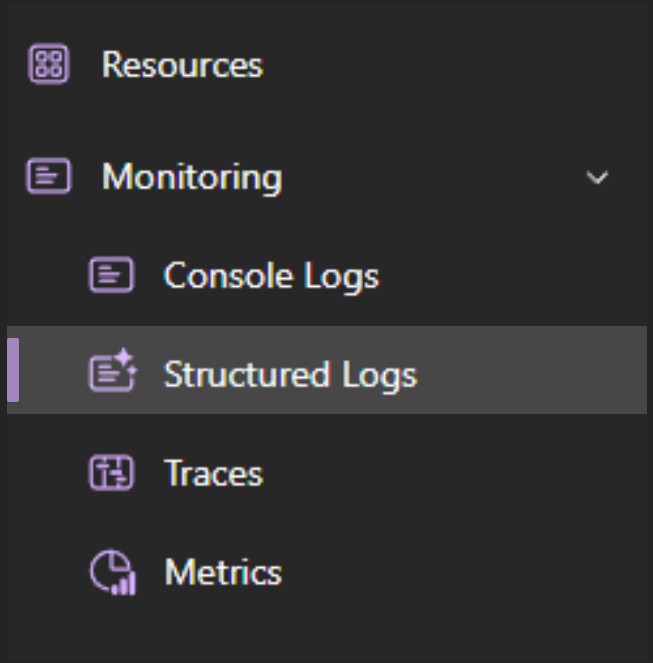
Watching logs...

```
2024-02-11T19:31:50.8086591 1:C 11 Feb 2024 16:31:50.808 # WARNING Memory overcommit must be enabled! Without it, a background save or replication may fail under low memory condition. Being disabled, it can also cause failures without low memory condition, see https://github.com/jemalloc/jemalloc/issues/1328. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
2024-02-11T19:31:50.8089400 1:C 11 Feb 2024 16:31:50.808 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
2024-02-11T19:31:50.8089549 1:C 11 Feb 2024 16:31:50.808 * Redis version=7.2.4, bits=64, commit=00000000, modified=0, pid=1, just started
2024-02-11T19:31:50.8089605 1:C 11 Feb 2024 16:31:50.808 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
2024-02-11T19:31:50.8092782 1:M 11 Feb 2024 16:31:50.809 * monotonic clock: POSIX clock_gettime
2024-02-11T19:31:50.8105184 1:M 11 Feb 2024 16:31:50.810 * Running mode=standalone, port=6379.
2024-02-11T19:31:50.8111490 1:M 11 Feb 2024 16:31:50.810 * Server initialized
2024-02-11T19:31:50.8112600 1:M 11 Feb 2024 16:31:50.811 * Ready to accept connections tcp
2024-02-11T20:31:51.0858077 1:M 11 Feb 2024 17:31:51.081 * 1 changes in 3600 seconds. Saving...
2024-02-11T20:31:51.0942201 1:M 11 Feb 2024 17:31:51.094 * Background saving started by pid 20
2024-02-11T20:31:51.1134831 20:C 11 Feb 2024 17:31:51.113 * DB saved on disk
2024-02-11T20:31:51.1152374 20:C 11 Feb 2024 17:31:51.114 * Fork CoW for RDB: current 0 MB, peak 0 MB, average 0 MB
2024-02-11T20:31:51.1947971 1:M 11 Feb 2024 17:31:51.194 * Background saving terminated with success
```

Dashboard


Resources
Monitoring
Console Logs
Structured Logs
Traces
Metrics

Dashboard


Resources
Monitoring
Console Logs
Structured Logs
Traces
Metrics

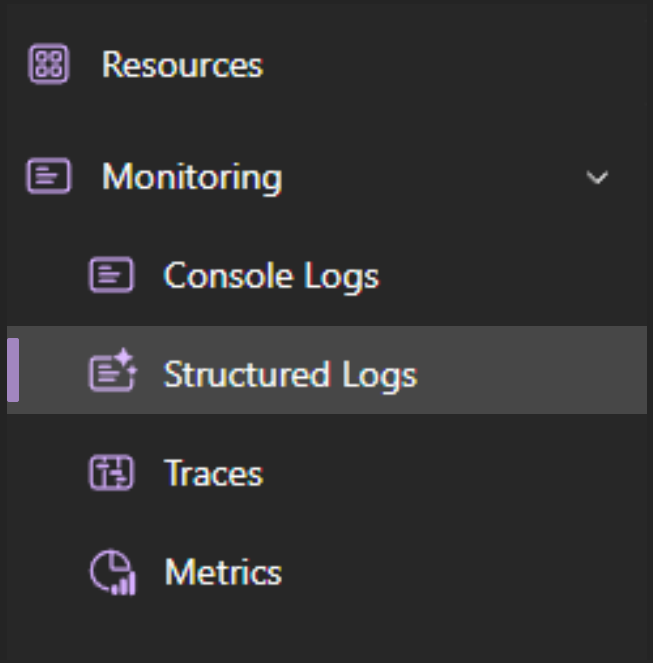
Structured Logs

(All)

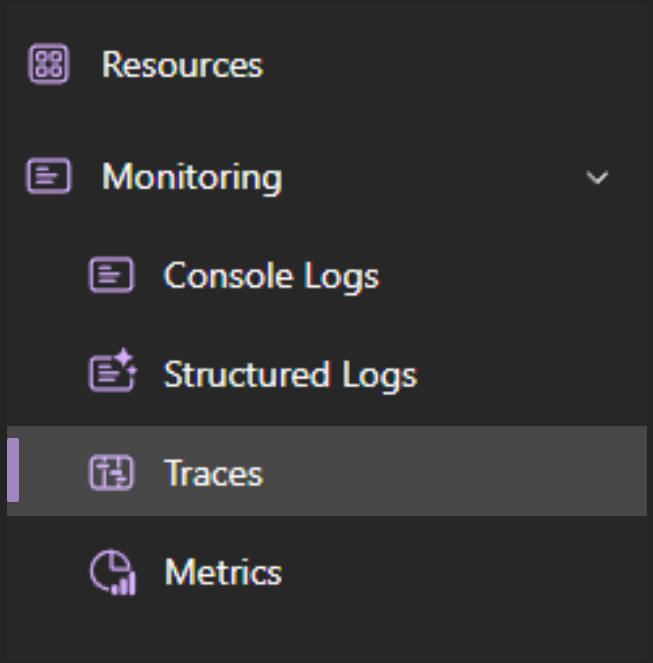
Filter... Level: (All) Filters: No Filters

Resource	Level	Timestamp	Message	Trace	Details
apiservice	Information	5:34:36.301 PM	Now listening on: http://localhost:59721		View
apiservice	Information	5:34:36.475 PM	Application started. Press Ctrl+C to shut down.		View
apiservice	Information	5:34:36.480 PM	Hosting environment: Development		View
apiservice	Information	5:34:36.482 PM	Content root path: D:\CSharp\Rofls\DefaultAspireApp\DefaultAspireApp.ApiService		View
webfrontend	Information	5:34:42.243 PM	Now listening on: http://localhost:59724		View
webfrontend	Information	5:34:42.277 PM	Application started. Press Ctrl+C to shut down.		View
webfrontend	Information	5:34:42.280 PM	Hosting environment: Development		View
webfrontend	Information	5:34:42.283 PM	Content root path: D:\CSharp\Rofls\DefaultAspireApp\DefaultAspireApp.Web		View

Dashboard


Resources
Monitoring
Console Logs
Structured Logs
Traces
Metrics

Dashboard


Resources
Monitoring
Console Logs
Structured Logs
Traces
Metrics

Traces

(All)

Filter...

Timestamp	Name	Spans	Durati...	
5:34:36.376 PM	apiservice: / 7dee8a5	apiservice (1)	105.31ms	View
5:39:41.112 PM	webfrontend: SET 63245cc	webfrontend (1)	5.12ms	View
5:39:41.122 PM	webfrontend: ZSCAN 490b8a7	webfrontend (1)	3.04ms	View
5:39:41.126 PM	webfrontend: ZREMRANGEBYSCORE 7398534	webfrontend (1)	2.91ms	View
5:39:41.130 PM	webfrontend: UNLINK 50bb72b	webfrontend (1)	980µs	View
5:43:35.225 PM	webfrontend: / 5b47f12	apiservice (1) webfrontend (1)	174.21ms	View
5:44:32.140 PM	webfrontend: SET b93740b	webfrontend (1)	1.46ms	View
5:44:32.142 PM	webfrontend: ZSCAN 9dd23c8	webfrontend (1)	1.09ms	View
5:44:32.143 PM	webfrontend: ZREMRANGEBYSCORE 3e0fa34	webfrontend (1)	871µs	View
5:44:32.144 PM	webfrontend: UNLINK 835c8d1	webfrontend (1)	667.6µs	View
5:48:59.213 PM	apiservice: / d3c8977	apiservice (1) webfrontend (1)	23.15ms	View

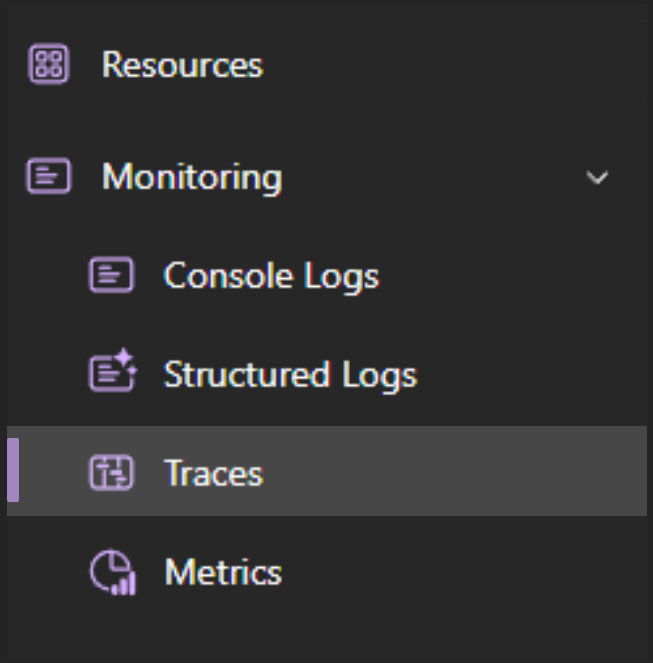
Traces

webfrontend: /weather 07de4c9

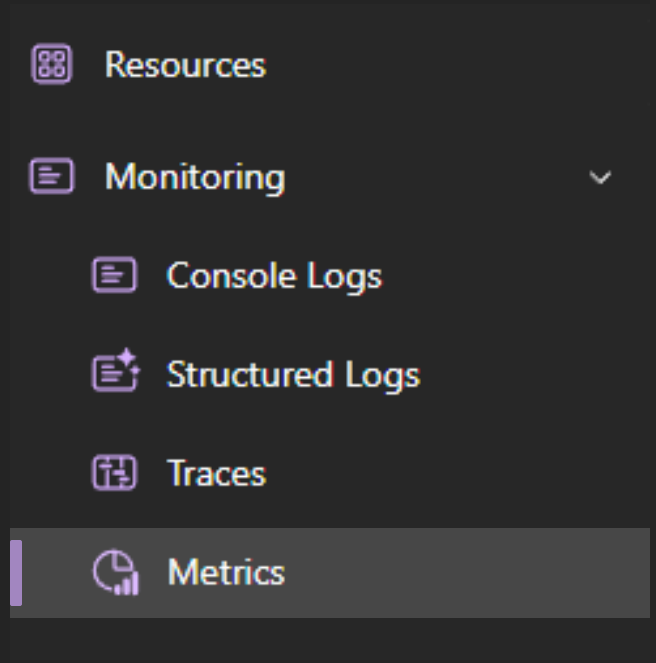
Trace Start **February 11 2024, 7:32:10.949 PM** Duration **279.32ms** Resources **2** Depth **3** Total Spans **6** View Logs Back

Name	0µs	69.83ms	139.66ms	209.49ms	279.32ms
webfrontend /weather					
webfrontend Browser Link HTTP GET 200	2.28ms				
webfrontend cache DATA redis GET	6.18ms				
webfrontend HTTP GET 200			21.66ms		
apiservice /weatherforecast			14.06ms		
webfrontend cache DATA redis SETEX					3.82ms

Dashboard


Resources
Monitoring
Console Logs
Structured Logs
Traces
Metrics

Dashboard


Resources
Monitoring
Console Logs
Structured Logs
Traces
Metrics

Metrics

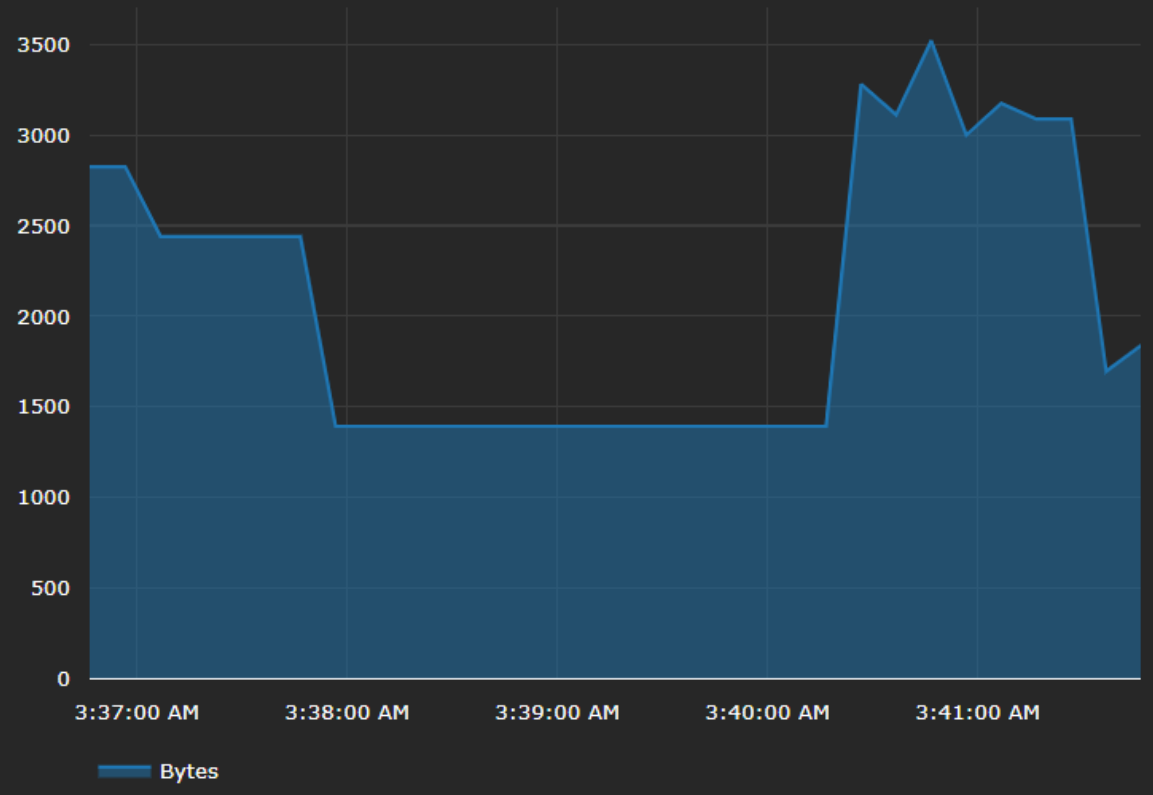
apiservice

Last 5 minutes

- Microsoft.AspNetCore.Server.Kestrel
  - kestrel.active_connections
  - kestrel.connection.duration
  - kestrel.queued_connections
- OpenTelemetry.Instrumentation.Runtime
  - process.runtime.dotnet.assemblies.co
  - process.runtime.dotnet.exceptions.co
  - process.runtime.dotnet.gc.allocations.
  - process.runtime.dotnet.gc.collections.
  - process.runtime.dotnet.gc.committed
  - process.runtime.dotnet.gc.duration
  - process.runtime.dotnet.gc.heap.fragm
  - process.runtime.dotnet.gc.heap.size
  - process.runtime.dotnet.gc.objects.size
  - process.runtime.dotnet.jit.compilatior
  - process.runtime.dotnet.jit.il_compiled
  - process.runtime.dotnet.jit.methods_cc

process.runtime.dotnet.gc.heap.fragmentation.size

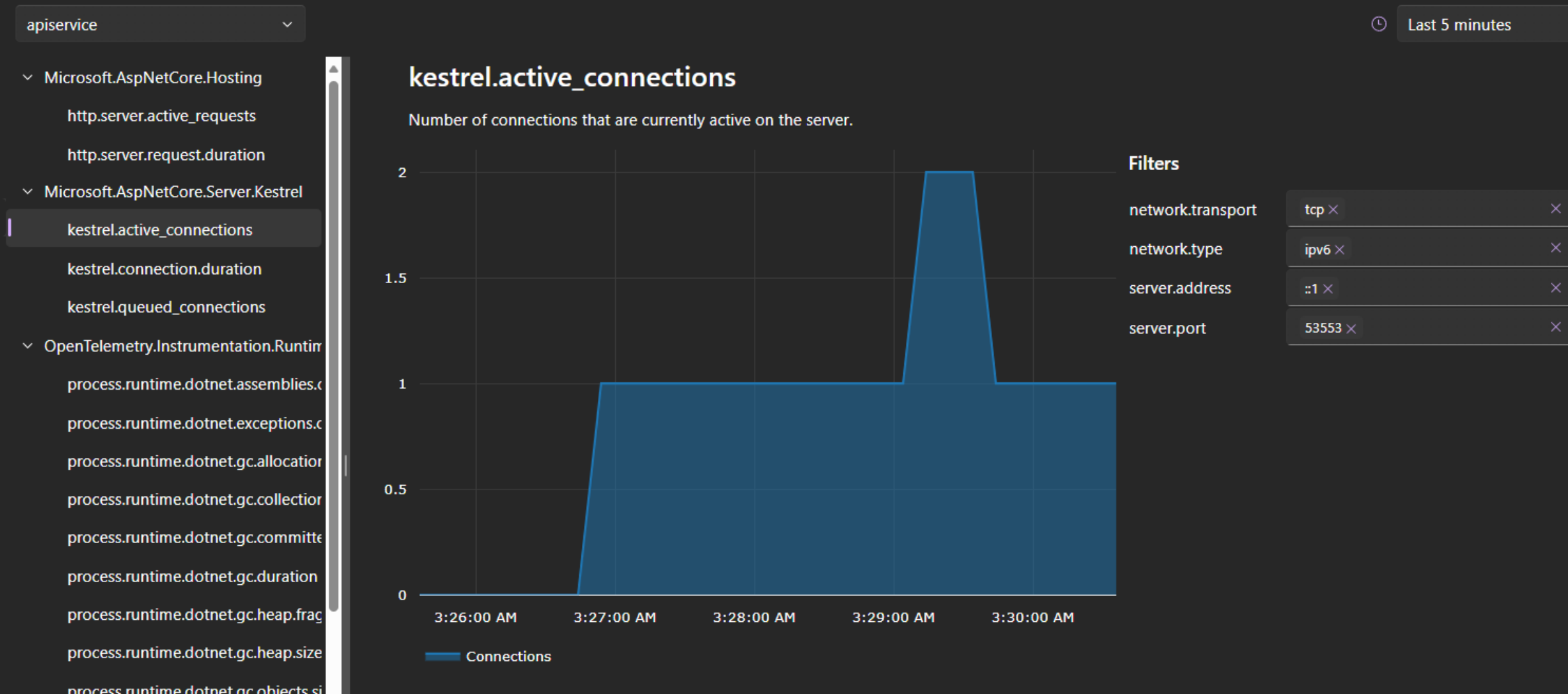
The heap fragmentation, as observed during the latest garbage collection. The value will be unavailable until at least one garbage collection has occurred.



Filters

generation gen1 × gen2 ×

Metrics


apiservice
Last 5 minutes
Microsoft.AspNetCore.Hosting
http.server.active_requests
http.server.request.duration
Microsoft.AspNetCore.Server.Kestrel
kestrel.active_connections
kestrel.connection.duration
kestrel.queued_connections
OpenTelemetry.Instrumentation.Runtim
process.runtime.dotnet.assemblies.c
process.runtime.dotnet.exceptions.c
process.runtime.dotnet.gc.allocatior
process.runtime.dotnet.gc.collectior
process.runtime.dotnet.gc.committe
process.runtime.dotnet.gc.duration
process.runtime.dotnet.gc.heap.frag
process.runtime.dotnet.gc.heap.size
kestrel.active_connections
Number of connections that are currently active on the server.
2
1.5
1
0.5
0
3:26:00 AM
3:27:00 AM
3:28:00 AM
3:29:00 AM
3:30:00 AM
Connections
Filters
network.transport
tcp
network.type
ipv6
server.address
::1
server.port
53553

Итого

1. Все сервисы в одном месте
2. Просмотр логов
3. Отслеживание ошибок
4. Метрики
5. Красивое…
6. Это отдельный компонент!

Использование Aspire сейчас

Плюсы:

- Оркестрация
- Компоненты
- Дашборд и отладка
- Просто

Использование Aspire сейчас

Плюсы:

- Оркестрация
- Компоненты
- Дашборд и отладка
- Просто

Минусы:

- Ориентирован на локальную разработку
- Не востребован на рынке
- Breaking Changes
- Непонятная поддержка компонентов

Использование Aspire сейчас

Плюсы:

- Оркестрация
- Компоненты
- Дашборд и отладка
- Просто

Минусы:

- Ориентирован на локальную разработку
- Не востребован на рынке
- Breaking Changes
- Непонятная поддержка компонентов

Подводные камни:

Ждём релиз?

- Осень 2024
- Кроссплатформенность
- Куча компонентов

Ждём релиз?

- Осень 2024
- Кроссплатформенность
- Куча компонентов

```
var database = builder.AddPostgres("postgresql")
    .AddDatabase("customers");
```

Ждём релиз?

- Осень 2024
- Кроссплатформенность
- Куча компонентов

```
var database = builder.AddPostgres("postgresql")
    .WithPgAdmin()
    .AddDatabase("customers");
```

Полезные ссылки:

RadioDotNet

https://www.youtube.com/playlist?list=PLbxr_aGL4q3SpQ9GRn2jv-NEpvN23CUC5

.NET Aspire docs

https://learn.microsoft.com/en-us/dotnet/aspire/

Полезные ссылки:

RadioDotNet

https://www.youtube.com/playlist?list=PLbxr_aGL4q3SpQ9GRn2jv-NEpvN23CUC5

.NET Aspire docs

https://learn.microsoft.com/en-us/dotnet/aspire/

Менее полезные:

@bornToWhine

Alexander Goldebaev

sgoldebaev@gmail.com

Alexanderbtw