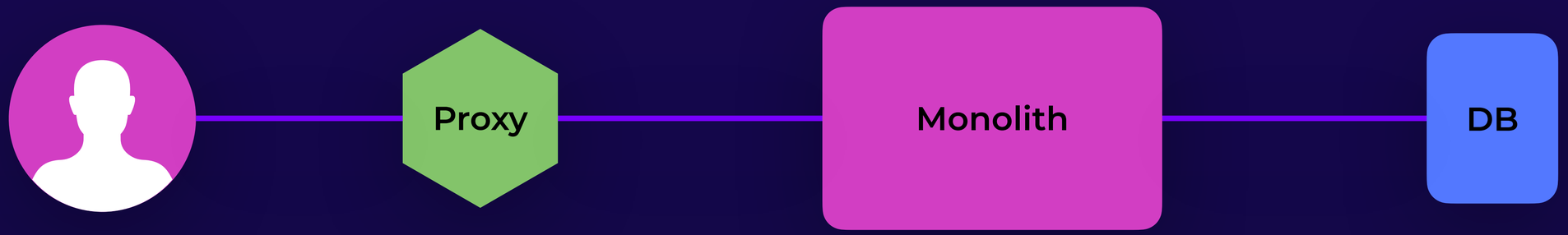
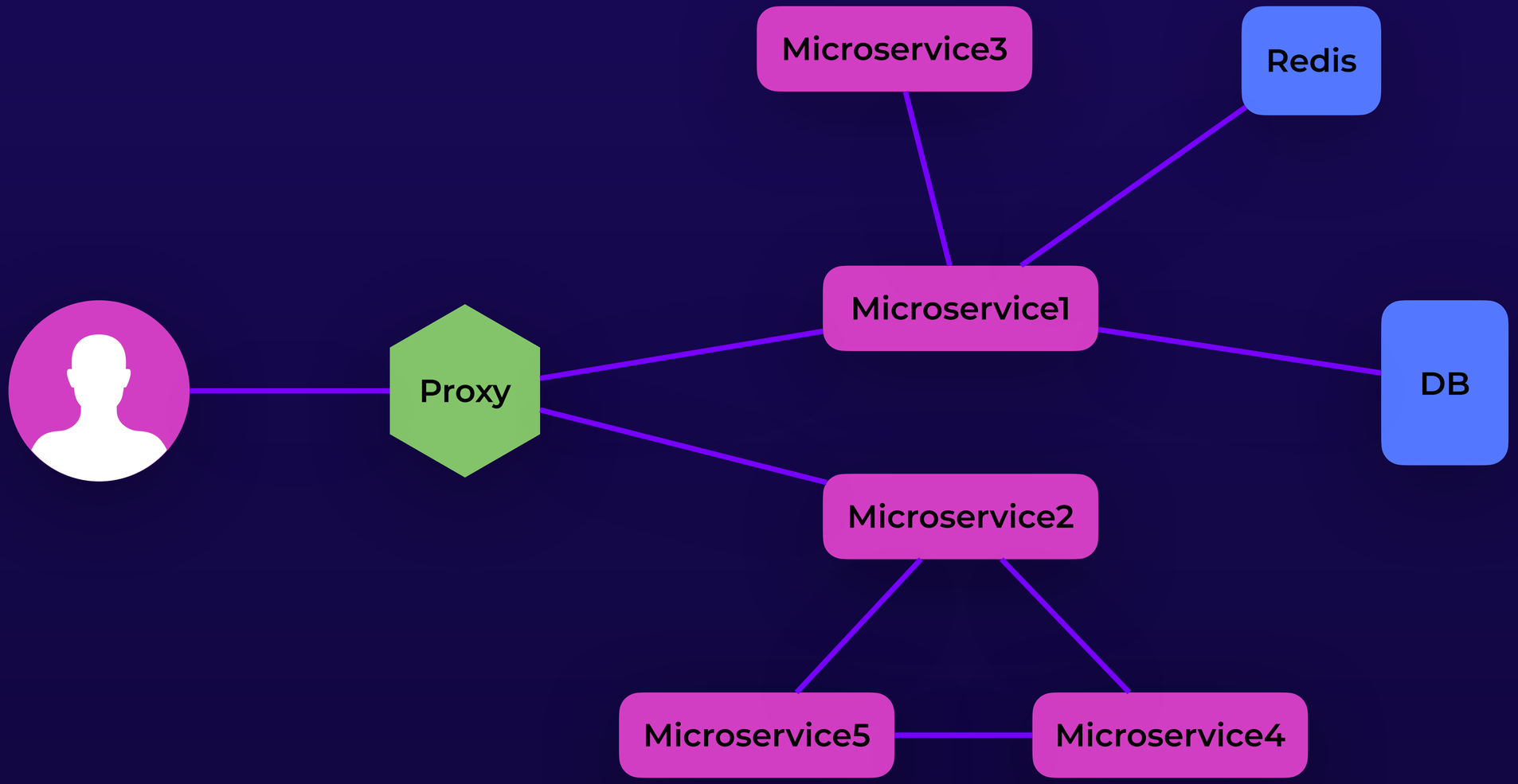
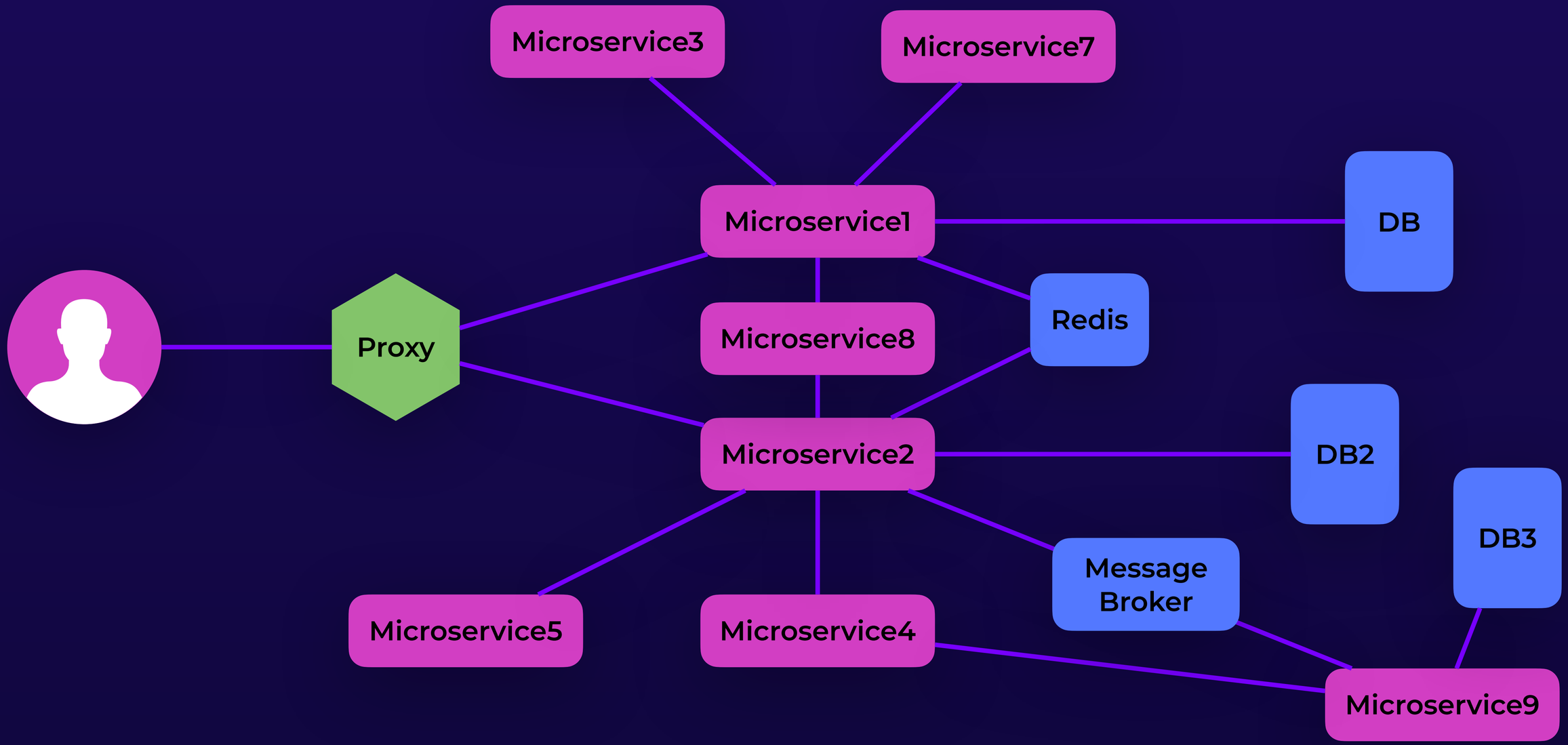
.NET Aspire

Гольдебаев Александр


Proxy
Monolith
DB


Microservice3
Redis
Microservice1
DB
Proxy
Microservice2
Microservice5
Microservice4


Microservice3
Microservice7
Microservice1
DB
Proxy
Microservice8
Redis
Microservice2
DB2
DB3
Message Broker
Microservice5
Microservice4
Microservice9

Проблемы?

При создании распределённых cloud-native приложений

Проблемы?

При создании распределённых cloud-native приложений

- Конфигурация окружения

Проблемы?

При создании распределённых cloud-native приложений

- Конфигурация окружения
- Переизобретение колеса

Проблемы?

При создании распределённых cloud-native приложений

- Конфигурация окружения
- Переизобретение колеса
- Отказоустойчивость

Проблемы?

При создании распределённых cloud-native приложений

- Конфигурация окружения
- Переизобретение колеса
- Отказоустойчивость
- Наблюдаемость

Решение!

В виде дополнительного слоя абстракции

Решение!

В виде дополнительного слоя абстракции

- Сборка и запуск всех сервисов по одной кнопке

Решение!

В виде дополнительного слоя абстракции

- Сборка и запуск всех сервисов по одной кнопке
- Упрощенный интерфейс оркестрации и конфигурации

Решение!

В виде дополнительного слоя абстракции

- Сборка и запуск всех сервисов по одной кнопке
- Упрощенный интерфейс оркестрации и конфигурации
- Настройки по-умолчанию для:
  - Сбора и экспорта телеметрии
  - Отказоустойчивости (Retries, Timeouts, Circuit-breakers)
  - Связи между сервисами (Service Discovery)

Project Tye

Основные идеи

`tye run` `tye.yaml`

Project Tye

Основные идеи

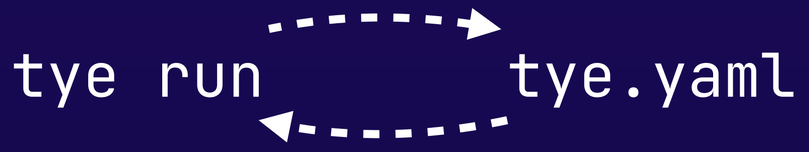
- .NET Tool

`tye run` `tye.yaml`

Project Tye

Основные идеи

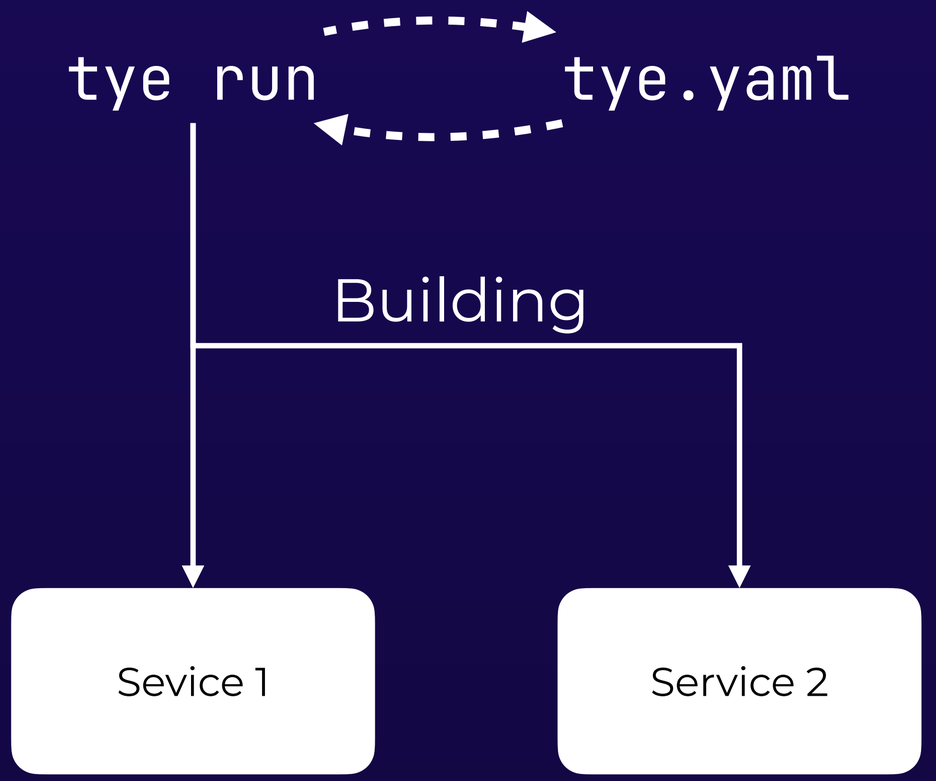
- .NET Tool
- Конфигурация на YAML



Project Tye

Основные идеи

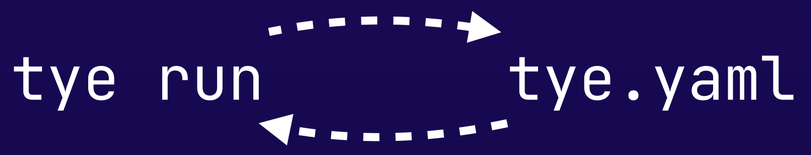
- .NET Tool
- Конфигурация на YAML
- Building
  (.csproj / docker-container / etc)



Project Tye

Основные идеи

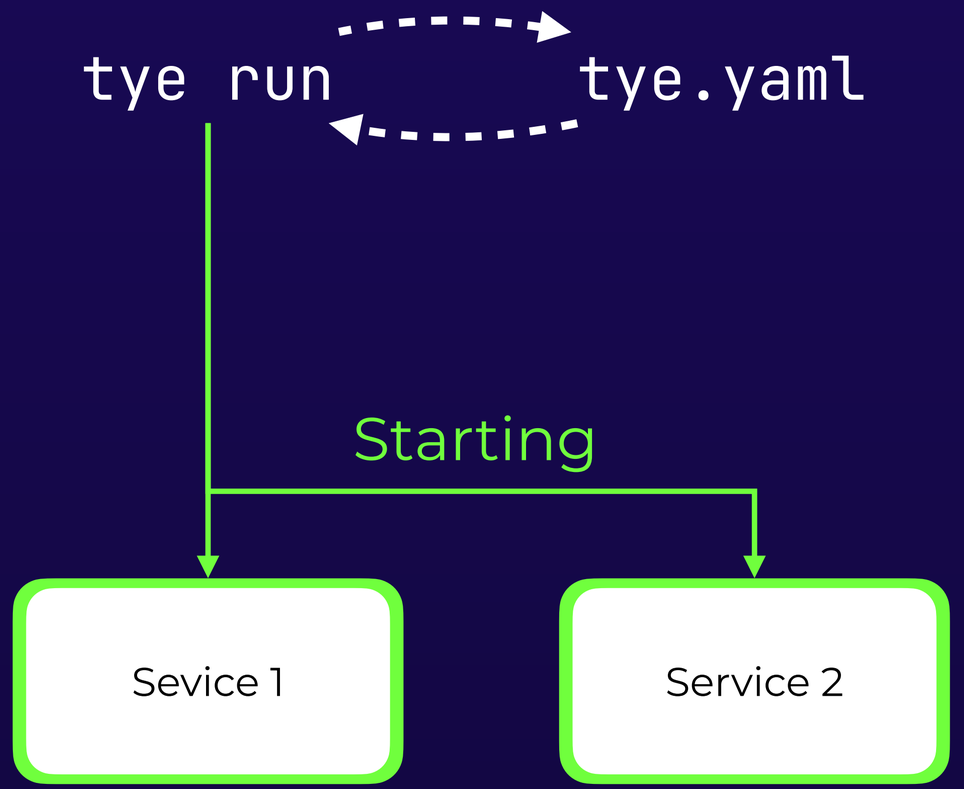
- .NET Tool
- Конфигурация на YAML
- Building
  (.csproj / docker-container / etc)



Project Tye

Основные идеи

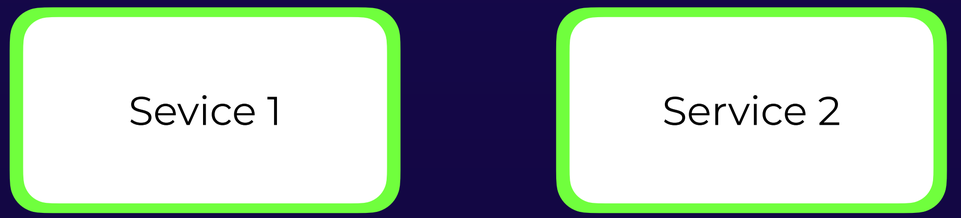
- .NET Tool
- Конфигурация на YAML
- Building
  (.csproj / docker-container / etc)



Project Tye

Основные идеи

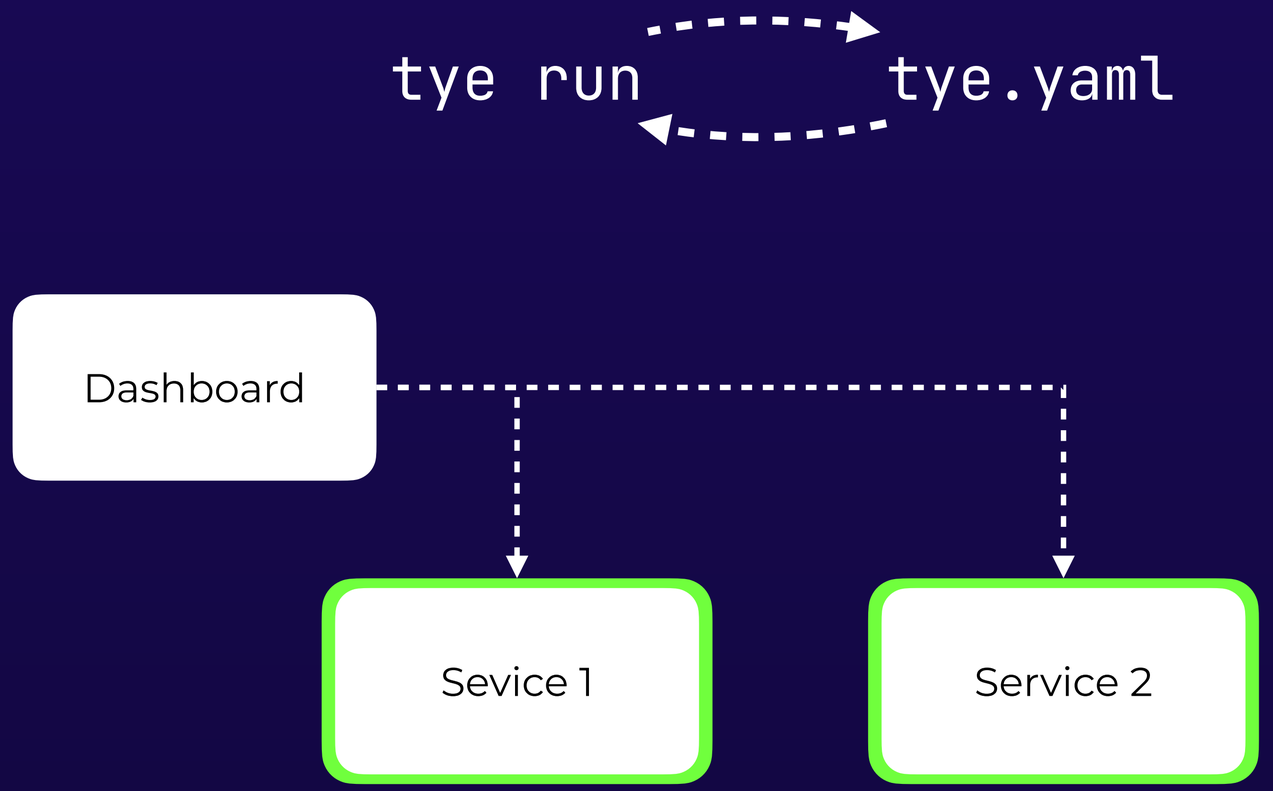
- .NET Tool
- Конфигурация на YAML
- Building
  (.csproj / docker-container / etc)



Project Tye

Основные идеи

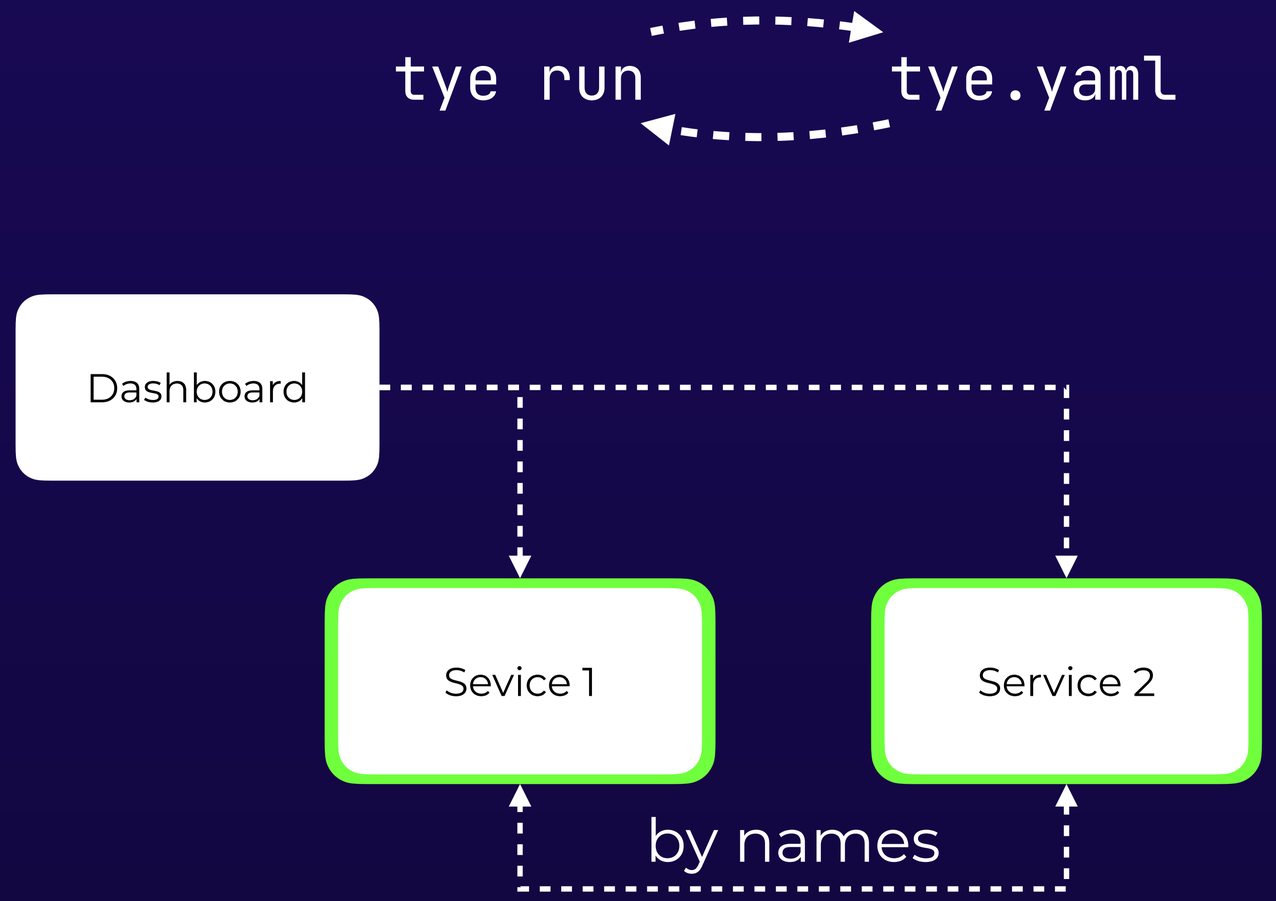
- .NET Tool
- Конфигурация на YAML
- Building
  (.csproj / docker-container / etc)
- Dashboard



Project Tye

Основные идеи

- .NET Tool
- Конфигурация на YAML
- Building
  (.csproj / docker-container / etc)
- Dashboard
- Service Discovery



.NET Aspire

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

Developer Dashboard

Orchestration

Components

Service Discovery

Deployment

Внедряем Aspire

Внедряем Aspire

```
dotnet workload install aspire
```

Внедряем Aspire

```
dotnet workload install aspire
```

Tooling:

Внедряем Aspire

```
dotnet workload install aspire
```

Tooling:

2022 (17.10)

By Default

Внедряем Aspire

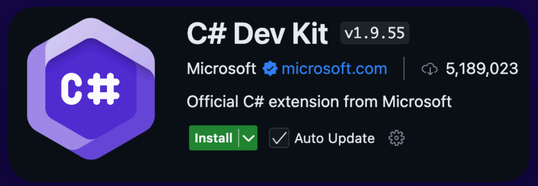
```
dotnet workload install aspire
```

Tooling:

2022 (17.10)

By Default

By Extension



Внедряем Aspire

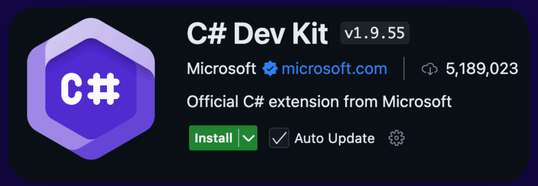
```
dotnet workload install aspire
```

Tooling:

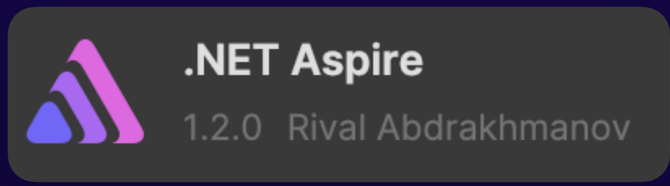
2022 (17.10)

By Default

By Extension



By Plugin



Внедряем Aspire

Внедряем Aspire

1. Создаём новый с помощью шаблона
    - Application (Empty)
    - Starter

Внедряем Aspire

1. Создаём новый с помощью шаблона
    - Application (Empty)
    - Starter

2. Внедряем в уже существующий проект
    - .NET Aspire Orchestration Support (for VS)
    - Ручками

Внедряем Aspire

1. Создаём новый с помощью шаблона
   - Application (Empty)
   - Starter

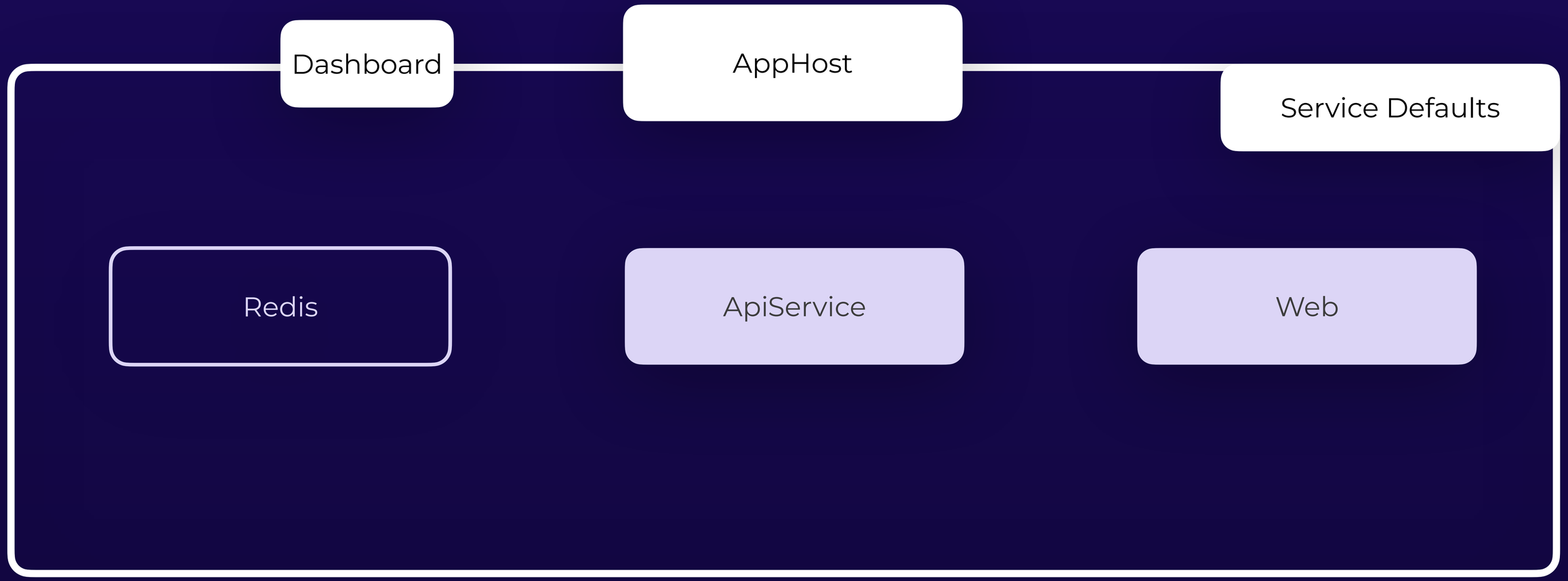
2. Внедряем в уже существующий проект
   - .NET Aspire Orchestration Support (for VS)
   - Ручками

Внедряем Aspire

1. Создаём новый с помощью шаблона
    - Application (Empty)
    - Starter

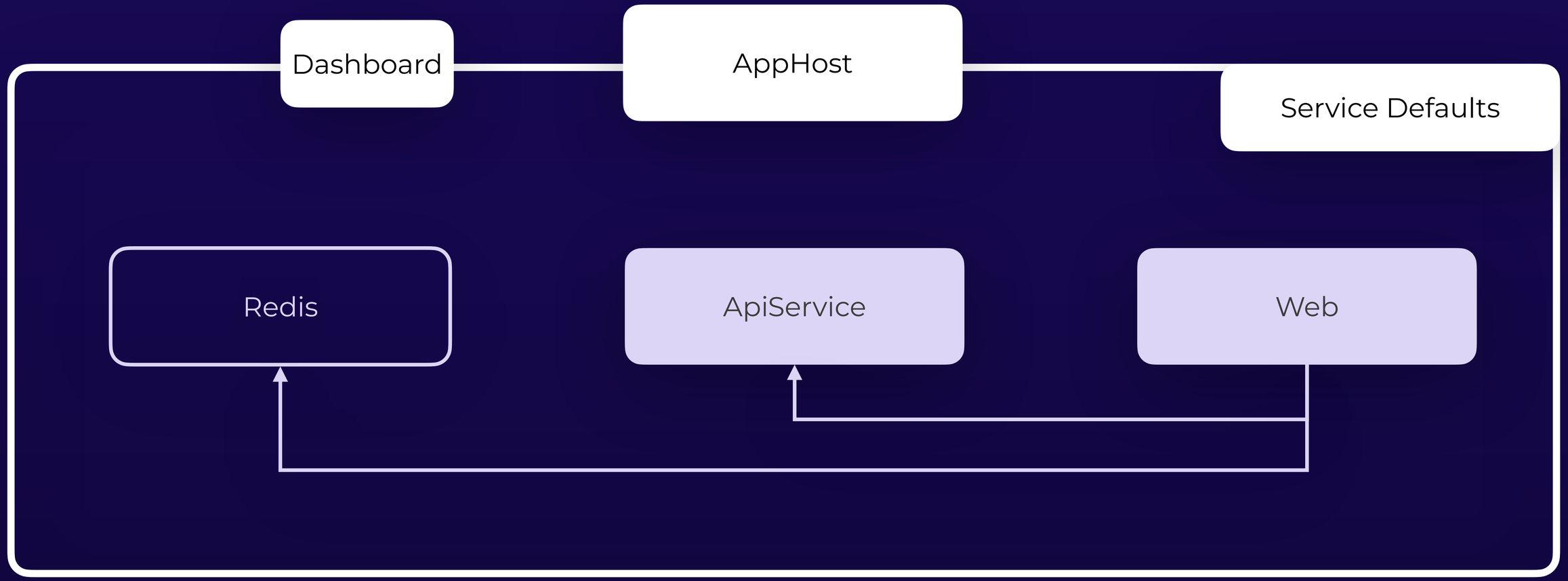
Внедряем Aspire

Starter



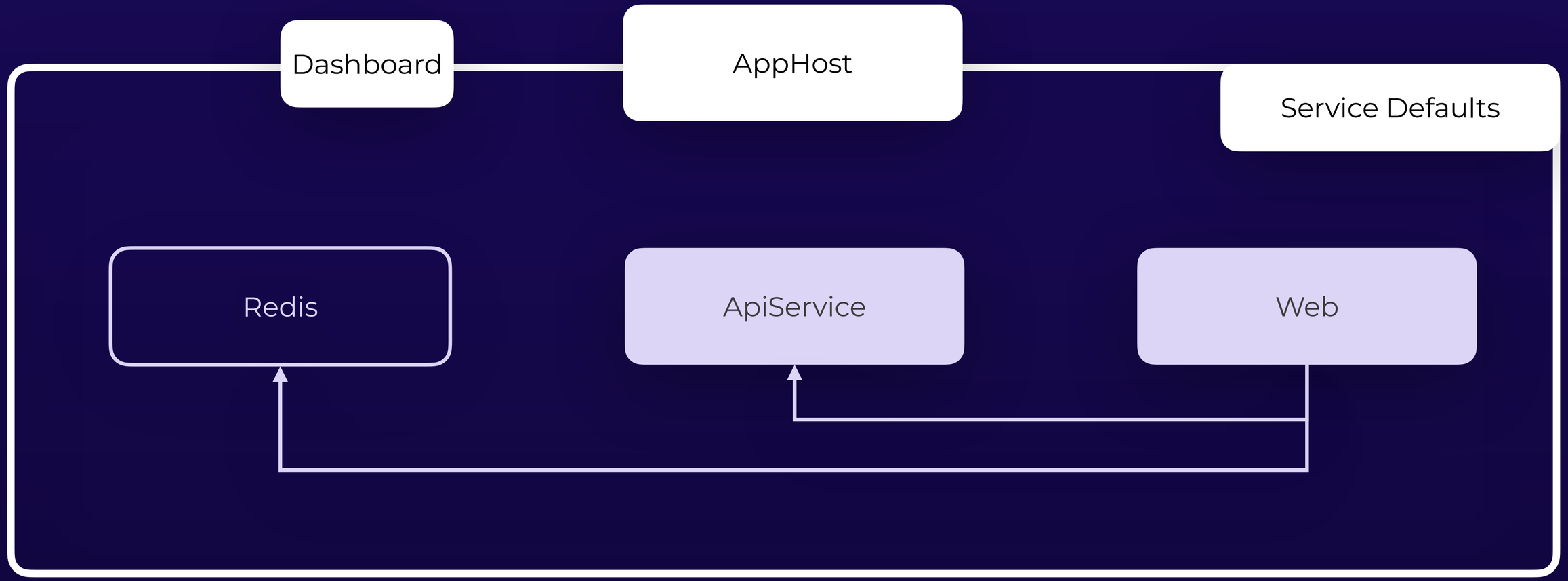
Внедряем Aspire

Starter



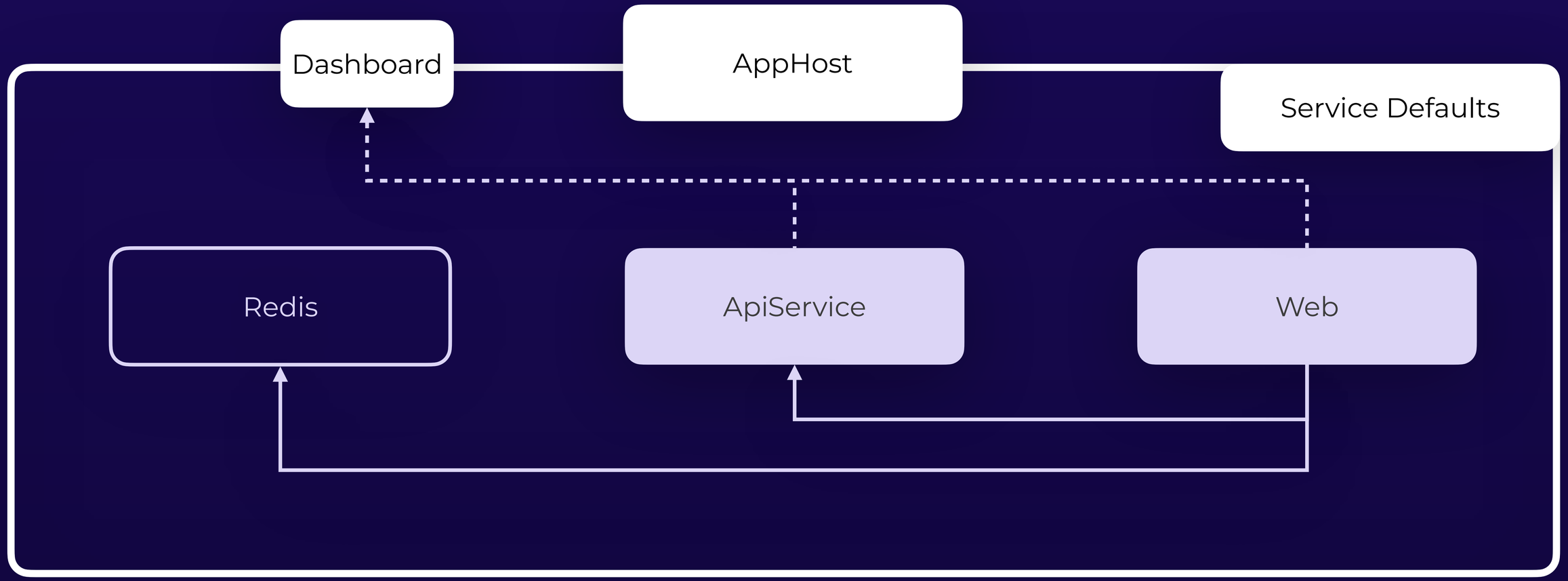
Внедряем Aspire

Starter



Внедряем Aspire

Starter



Внедряем Aspire

AppHost

Program.cs

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

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

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

builder.AddProject<Projects.AspireStarter_Web>("webfrontend")
    .WithExternalHttpEndpoints()
    .WithReference(cache)
    .WithReference(apiService);

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

Внедряем Aspire

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>
        <UserSecretsId>3FACCCB2-7688-403B-8485-1E49D621B2BD</UserSecretsId>
    </PropertyGroup>

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

    <ItemGroup>
        <PackageReference Include="Aspire.Hosting.AppHost" Version="8.0.1" />
        <PackageReference Include="Aspire.Hosting.Redis" Version="8.0.1" />
    </ItemGroup>

</Project>
```

Внедряем Aspire

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();
```

Внедряем Aspire

ApiService

Program.cs

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


builder.Services.AddProblemDetails();

var app = builder.Build();

app.UseExceptionHandler();

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

app.MapDefaultEndpoints();

app.Run();
```

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

Внедряем Aspire

ApiService

Program.cs

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


builder.Services.AddProblemDetails();

var app = builder.Build();

app.UseExceptionHandler();

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

app.MapDefaultEndpoints();

app.Run();
```

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

★ OpenTelemetry metrics and tracing.

Внедряем Aspire

ApiService

Program.cs

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


builder.Services.AddProblemDetails();

var app = builder.Build();

app.UseExceptionHandler();

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

app.MapDefaultEndpoints();

app.Run();
```

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

★ OpenTelemetry metrics and tracing.

★ Add default Health Check endpoints.

Внедряем Aspire

ApiService

Program.cs

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


builder.Services.AddProblemDetails();

var app = builder.Build();

app.UseExceptionHandler();

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

app.MapDefaultEndpoints();

app.Run();
```

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

★ OpenTelemetry metrics and tracing.

★ Add default Health Check endpoints.

★ Add Service Discovery functionality.

Внедряем Aspire

ApiService

Program.cs

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


builder.Services.AddProblemDetails();

var app = builder.Build();

app.UseExceptionHandler();

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

app.MapDefaultEndpoints();

app.Run();
```

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

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

Внедряем Aspire

ServiceDefaults

ServiceDefaults.csproj

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

    <PropertyGroup>
        <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="8.3.0" />
        <PackageReference Include="Microsoft.Extensions.ServiceDiscovery" Version="8.0.1" />
        <PackageReference Include="OpenTelemetry.Exporter.OpenTelemetryProtocol" Version="1.8.1" />
        <PackageReference Include="OpenTelemetry.Extensions.Hosting" Version="1.8.1" />
        <PackageReference Include="OpenTelemetry.Instrumentation.AspNetCore" Version="1.8.1" />
        <PackageReference Include="OpenTelemetry.Instrumentation.Http" Version="1.8.1" />
        <PackageReference Include="OpenTelemetry.Instrumentation.Runtime" Version="1.8.0" />
    </ItemGroup>

</Project>
```

Внедряем Aspire

ServiceDefaults

Extensions.cs

```
public static IHostApplicationBuilder AddServiceDefaults(this IHostApplicationBuilder builder)
{
    builder.ConfigureOpenTelemetry();

    builder.AddDefaultHealthChecks();

    builder.Services.AddServiceDiscovery();

    builder.Services.ConfigureHttpClientDefaults(http ⇒
    {
        http.AddStandardResilienceHandler();

        http.AddServiceDiscovery();
    });

    return builder;
}
```

Внедряем Aspire

ServiceDefaults

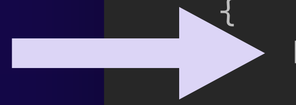
Extensions.cs

```
public static IHostApplicationBuilder AddServiceDefaults(this IHostApplicationBuilder builder)
{
    builder.ConfigureOpenTelemetry();

    builder.AddDefaultHealthChecks();

    builder.Services.AddServiceDiscovery();

    builder.Services.ConfigureHttpClientDefaults(http =>
    {
        http.AddStandardResilienceHandler();

        http.AddServiceDiscovery();
    });

    return builder;
}
```

Внедряем Aspire

Resilience

- 📦 Microsoft.Extensions.Resilience
- 📦 Microsoft.Extensions.Http.Resilience

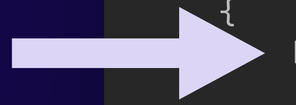


```
Ex

public static IHostApplicationBuilder AddSer
{
    builder.ConfigureOpenTelemetry();

    builder.AddDefaultHealthChecks();

    builder.Services.AddServiceDiscovery();

    builder.Services.ConfigureHttpClientDefa
    {
        http.AddStandardResilienceHandler();

        http.AddServiceDiscovery();
    });

    return builder;
}
```

Внедряем Aspire

Resilience

- 📦 Microsoft.Extensions.Resilience
- 📦 Microsoft.Extensions.Http.Resilience

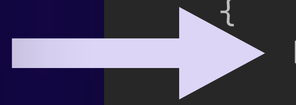


```
Ex

public static IHostApplicationBuilder AddSer
{
    builder.ConfigureOpenTelemetry();

    builder.AddDefaultHealthChecks();

    builder.Services.AddServiceDiscovery();

    builder.Services.ConfigureHttpClientDefa
    {
        http.AddStandardResilienceHandler();

        http.AddServiceDiscovery();
    });

    return builder;
}
```

Внедряем Aspire

Resilience

```
http.AddResilienceHandler("CustomPipeline", static builder ⇒
    {
        builder.AddRetry(new HttpRetryStrategyOptions
        {
            MaxRetryAttempts = 5,
        });
        builder.AddCircuitBreaker(new HttpCircuitBreakerStrategyOptions
        {
            SamplingDuration = TimeSpan.FromSeconds(10),
        });

        builder.AddTimeout(TimeSpan.FromSeconds(5));

        builder.AddFallback(new FallbackStrategyOptions<HttpResponseMessage>
        {
            FallbackAction = /* Some Fallback Action */
        });

        builder.AddPipeline(MyAlreadyCreatedPipeline);
        builder.AddHedging(new HttpHedgingStrategyOptions { /* */ });
    });
```



```
Ex

public static IHostApplicationBuilder AddSer
{
    builder.ConfigureOpenTelemetry();

    builder.AddDefaultHealthChecks();

    builder.Services.AddServiceDiscovery();

    builder.Services.ConfigureHttpClientDefa
    {
        http.AddStandardResilienceHandler();

        http.AddServiceDiscovery();
    });

    return builder;
}
```

Внедряем Aspire

Resilience

Order	Strategy	Description	Defaults
1	Rate limiter	The rate limiter pipeline limits the maximum number of concurrent requests being sent to the dependency.	Queue: 0 Permit: 1_000
2	Total timeout	The total request timeout pipeline applies an overall timeout to the execution, ensuring that the request, including retry attempts, doesn't exceed the configured limit.	Total timeout: 30s
3	Retry	The retry pipeline retries the request in case the dependency is slow or returns a transient error.	Max retries: 3 Backoff: Exponential Use jitter: true Delay:2s
4	Circuit breaker	The circuit breaker blocks the execution if too many direct failures or timeouts are detected.	Failure ratio: 10% Min throughput: 100 Sampling duration: 30s Break duration: 5s
5	Attempt timeout	The attempt timeout pipeline limits each request attempt duration and throws if it's exceeded.	Attempt timeout: 10s

```
public static IHos
{
    builder.Config

    builder.AddDef

    builder.Servic

    builder.Servic
    {
        http.AddSt

        http.AddSe
    });

    return builder
}
```

Внедряем Aspire

Service Discovery

Extensions.cs

```
public static IHostApplicationBuilder AddServiceDefaults(this IHostApplicationBuilder builder)
{
    builder.ConfigureOpenTelemetry();

    builder.AddDefaultHealthChecks();

    builder.Services.AddServiceDiscovery();

    builder.Services.ConfigureHttpClientDefaults(http ⇒
    {
        http.AddStandardResilienceHandler();

        http.AddServiceDiscovery();
    });

    return builder;
}
```

Внедряем Aspire

Service Discovery

Extensions.cs

```
public static IHostApplicationBuilder AddServiceDefaults(this IHostApplicationBuilder builder)
{
    builder.ConfigureOpenTelemetry();

    builder.AddDefaultHealthChecks();

    builder.Services.AddServiceDiscovery();

    builder.Services.ConfigureHttpClientDefaults(http ⇒
    {
        http.AddStandardResilienceHandler();

        http.AddServiceDiscovery();
    });

    return builder;
}
```

Внедряем Aspire

Service Discovery

Extensions.cs

```
public static IHostApplicationBuilder AddServiceDefaults(this IHostApplicationBuilder builder)
{
    builder.ConfigureOpenTelemetry();

    builder.AddDefaultHealthChecks();

    builder.Services.AddServiceDiscovery();

    builder.Services.ConfigureHttpClientDefaults(http ⇒
    {
        http.AddStandardResilienceHandler();

        http.AddServiceDiscovery();
    });

   builder.Services.AddHttpClient<WeatherServiceClient>(
        static client ⇒
            client.BaseAddress = new("http://weather"))
    .AddServiceDiscovery();


    return builder;
}
```

Внедряем Aspire

Service Discovery

Extensions.cs

```
public static IHostApplicationBuilder AddServiceDefaults(this IHostApplicationBuilder builder)
{
    builder.ConfigureOpenTelemetry();

    builder.AddDefaultHealthChecks();

    builder.Services.AddServiceDiscovery();

    builder.Services.ConfigureHttpClientDefaults(http =>
    {
        http.AddStandardResilienceHandler();

        http.AddServiceDiscovery();
    });

   builder.Services.AddHttpClient<WeatherServiceClient>(
        static client =>
            client.BaseAddress = new("http://weather"))
    .AddServiceDiscovery();


    return builder;
}
```

Внедряем Aspire

Service Discovery

Extensions.cs

```
public static IHostApplicationBuilder AddServiceDefaults(this IHostApplicationBuilder builder)
{
    builder.ConfigureOpenTelemetry();

    builder.AddDefaultHealthChecks();

    builder.Services.AddServiceDiscovery();

    builder.Services.ConfigureHttpClientDefaults(http =>
    {
        http.AddStandardResilienceHandler();

        http.AddServiceDiscovery();
    });

   builder.Services.AddHttpClient<WeatherServiceClient>(
        static client =>
            client.BaseAddress = new("http://weather"))
    .AddServiceDiscovery();


    return builder;
}
```

Внедряем Aspire

Service Discovery

Extensions.cs

```
public static IHostApplicationBuilder AddServiceDefaults(this IHostApplicationBuilder builder)
{
    builder.ConfigureOpenTelemetry();

    builder.AddDefaultHealthChecks();

    builder.Services.AddServiceDiscovery();

    builder.Services.ConfigureHttpClientDefaults(http ⇒
    {
        http.AddStandardResilienceHandler();

        http.AddServiceDiscovery();
    });

   builder.Services.AddHttpClient<WeatherServiceClient>(
        static client ⇒
            client.BaseAddress = new("https+http://weather"))
    .AddServiceDiscovery();


    return builder;
}
```

Внедряем Aspire

Service Discovery

appsettings.json

```
{
  "MyServiceEndpoints": {
    “weather”: {
      "https": [
        "localhost:8080",
        "10.46.24.90:80"
      ]
    }
  }
}
```

Extensions.cs

```
public static IHostApplicationBuilder AddServiceDefaults(this IHostApplicationBuilder bu
{
    builder.ConfigureOpenTelemetry();

    builder.AddDefaultHealthChecks();

    builder.Services.AddServiceDiscovery(options ⇒
    {
        options.AllowAllSchemes = false;
        options.AllowedSchemes = ["https"];
    });
    builder.Services
        .Configure<ConfigurationServiceEndpointProviderOptions>(
            static options ⇒
    {
        options.ShouldApplyHostNameMetadata =
            endpoint ⇒
                endpoint.EndPoint is DnsEndPoint dnsEp
                && dnsEp.Host.StartsWith("internal");
        options.SectionName = "MyServiceEndpoints";
    });

    builder.Services.ConfigureHttpClientDefaults(http ⇒
    {
        http.AddStandardResilienceHandler();
```

Внедряем Aspire

Service Discovery

appsettings.json

```
{
  "MyServiceEndpoints": {
    “weather”: {
      "https": [
        "localhost:8080",
        "10.46.24.90:80"
      ],
      "data": "10.46.24.90:81"
    }
  }
}
```

```
builder.Services.AddHttpClient<WeatherServiceClient>(
        static client ⇒
            client.BaseAddress = new(“https+http://weather”))
```

```
builder.Services.AddHttpClient<WeatherDataServiceClient>(
    static client ⇒
        client.BaseAddress = new("https+http://_data.weather"));
```

Внедряем Aspire

ServiceDefaults

Extensions.cs

```
public static IHostApplicationBuilder AddServiceDefaults(this IHostApplicationBuilder builder)
{
    builder.ConfigureOpenTelemetry();

    builder.AddDefaultHealthChecks();

    builder.Services.AddServiceDiscovery();

    builder.Services.ConfigureHttpClientDefaults(http ⇒
    {
        http.AddStandardResilienceHandler();

        http.AddServiceDiscovery();
    });

    return builder;
}
```

Внедряем Aspire

ServiceDefaults

Extensions.cs

```
public static IHostApplicationBuilder ConfigureOpenTelemetry(this IHostApplicationBuilder builder)
{
    builder.Logging.AddOpenTelemetry(logging =>
    {
        logging.IncludeFormattedMessage = true;
        logging.IncludeScopes = true;
    });

    builder.Services.AddOpenTelemetry()
        .WithMetrics(metrics =>
        {
            metrics.AddAspNetCoreInstrumentation().AddHttpClientInstrumentation().AddRuntimeInstrumentation();
        })
        .WithTracing(tracing =>
        {
            tracing.AddAspNetCoreInstrumentation().AddHttpClientInstrumentation();
        });
    builder.AddOpenTelemetryExporters();
    return builder;
}
```

Внедряем Aspire

ServiceDefaults

Extensions.cs

```
private static IHostApplicationBuilder AddOpenTelemetryExporters(this IHostApplicationBuilder builder)
{
    var useOtlpExporter = !string.IsNullOrWhiteSpace(builder.Configuration["OTEL_EXPORTER_OTLP_ENDPOINT"]);

    if (useOtlpExporter)
    {
        builder.Services.AddOpenTelemetry().UseOtlpExporter();
    }

    //Enable the Azure Monitor exporter (requires the Azure.Monitor.OpenTelemetry.AspNetCore package)
    //if (!string.IsNullOrEmpty(builder.Configuration["APPLICATIONINSIGHTS_CONNECTION_STRING"]))
    //{
    //    builder.Services.AddOpenTelemetry()
    //       .UseAzureMonitor();
    //}

    return builder;
}
```

Внедряем Aspire

ServiceDefaults

Extensions.cs

```
public static IHostApplicationBuilder AddDefaultHealthChecks(this IHostApplicationBuilder builder)
{
    builder.Services.AddHealthChecks()
        .AddCheck("self", () => HealthCheckResult.Healthy(), ["live"]);

    return builder;
}

public static WebApplication MapDefaultEndpoints(this WebApplication app)
{
    if (app.Environment.IsDevelopment())
    {
        app.MapHealthChecks("/health");
        app.MapHealthChecks("/alive", new HealthCheckOptions
        {
            Predicate = r => r.Tags.Contains("live")
        });
    }

    return app;
}
```

Внедряем Aspire

Web

Program.cs

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

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

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

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

Внедряем Aspire

Web

```
builder.AddServiceDefaults();
```
- Конфигурация по-умолчанию

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

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

Внедряем Aspire

Web

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

- Конфигурация по-умолчанию

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

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

Используем Service Discovery

Внедряем Aspire

Web

```
builder.AddServiceDefaults();
```
- Конфигурация по-умолчанию

```
builder.AddRedisOutputCache("cache");
```
- Подключение Redis

```
builder.Services.AddHttpClient<WeatherApiClient>(client ⇒
    client.BaseAddress = new("https+http://apiservice"));
```
Используем Service Discovery

Оркестрация

Resources

Метод	Тип добавляемого ресурса	Описание

Оркестрация

Resources

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект

Оркестрация

Resources

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект

```
AddProject<TProject>(this IDistributedApplicationBuilder builder, string name)
```

Оркестрация

Resources

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект

```
AddProject<TProject>(this IDistributedApplicationBuilder builder, string name)

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

Оркестрация

Resources

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект

```
AddProject<TProject>(this IDistributedApplicationBuilder builder, string name)

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

AddProject<TProject>(this IDistributedApplicationBuilder builder, string name, string? launchProfileName)
```

Оркестрация

Resources

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект

```
AddProject<TProject>(this IDistributedApplicationBuilder builder, string name)

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

AddProject<TProject>(this IDistributedApplicationBuilder builder, string name, string? launchProfileName)

AddProject(this IDistributedApplicationBuilder builder, string name, string projectPath, string? launchProfileName)
```

Оркестрация

Resources

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект

Оркестрация

Resources

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект
AddContainer	ContainerResource	Docker/Podman контейнер

Оркестрация

Resources

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект
AddContainer	ContainerResource	Docker/Podman контейнер

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

Оркестрация

Resources

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект
AddContainer	ContainerResource	Docker/Podman контейнер

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

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

Оркестрация

Resources

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект
AddContainer	ContainerResource	Docker/Podman контейнер

Оркестрация

Resources

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект
AddContainer	ContainerResource	Docker/Podman контейнер
AddExecutable	ExecutableResource	Исполняемый файл

Оркестрация

Resources

Метод	Тип добавляемого ресурса	Описание
AddProject	ProjectResource	Любой .NET проект
AddContainer	ContainerResource	Docker/Podman контейнер
AddExecutable	ExecutableResource	Исполняемый файл

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

Оркестрация

Inner-loop Networking

Оркестрация

Inner-loop Networking

★ Launch profiles

Оркестрация

Inner-loop Networking

★ Launch profiles

★ Kestrel configuration

Оркестрация

Inner-loop Networking

★ Launch profiles

★ Kestrel configuration

★ Service bindings (Endpoint configuration)

Оркестрация

Inner-loop Networking

★ Launch profiles

★ Kestrel configuration

★ Service bindings (Endpoint configuration)

```
builder.AddProject<Projects.AspireStarter_Web>("webfrontend")
    .WithEndpoint(port: 1234, targetPort: 5453, scheme: "http", name: "web", isExternal: true)
    .WithHttpEndpoint(/* same as ^ without scheme */)
    .WithHttpsEndpoint(/* same as ^ without scheme */)
    .WithExternalHttpEndpoints();
```

Оркестрация

Inner-loop Networking

★ Launch profiles

★ Kestrel configuration

★ Service bindings (Endpoint configuration)

Оркестрация

Inner-loop Networking

★ Launch profiles

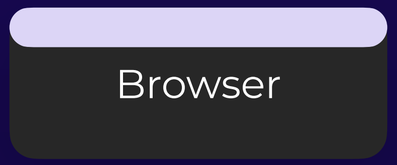
★ Kestrel configuration

★ Service bindings (Endpoint configuration)

★ Proxies

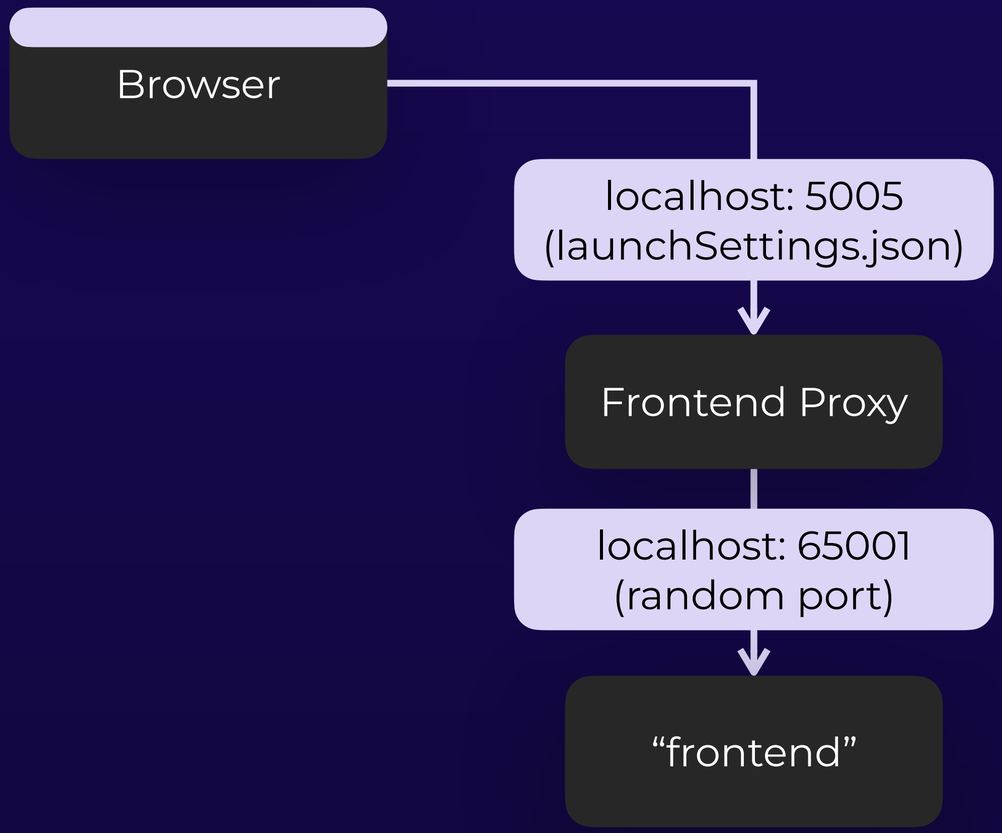
Оркестрация

Service bindings



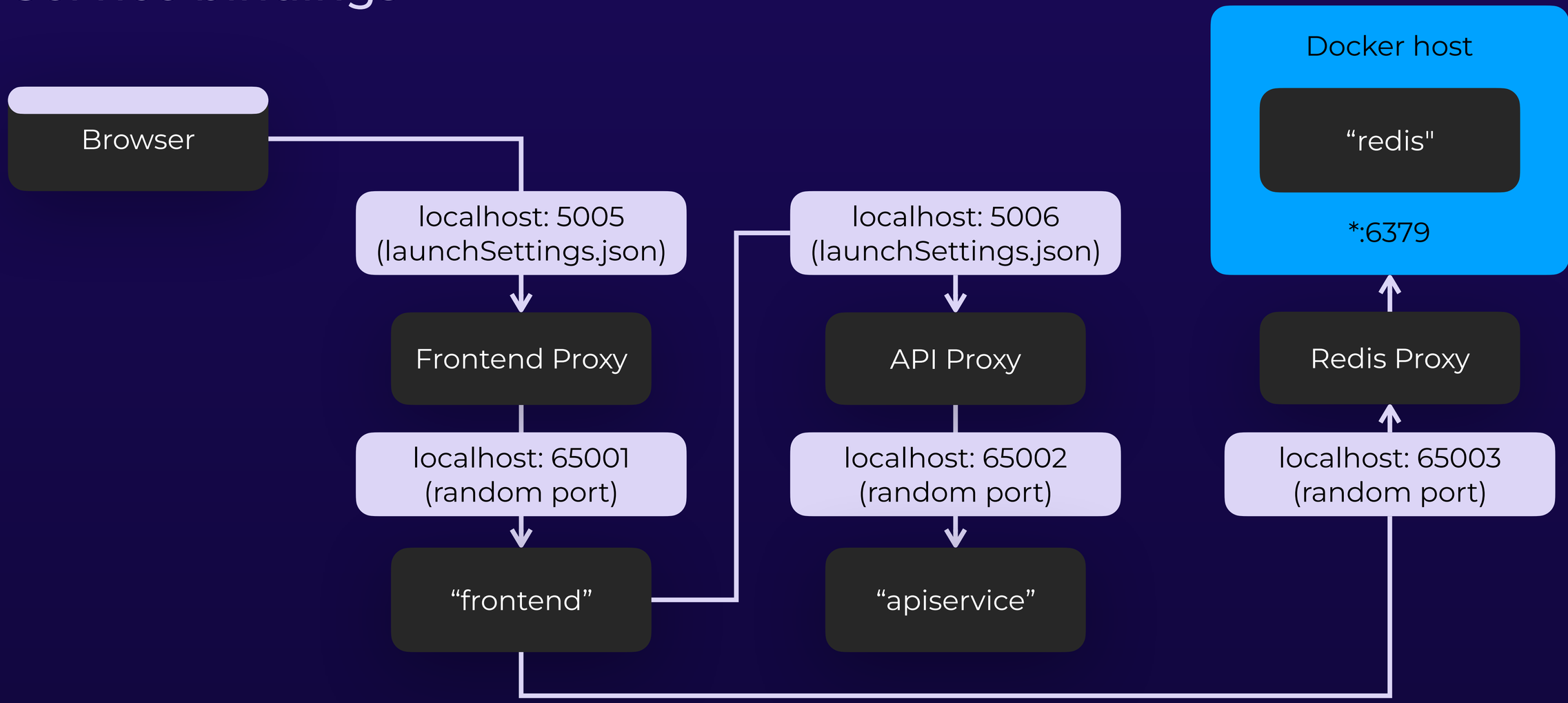
Оркестрация

Service bindings



Оркестрация

Service bindings



Оркестрация

Launch profiles

AppHost/launchSettings.json

```
{
  "$schema": "https://json.schemastore.org/launchsettings.json",
  "profiles": {
    "https": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "applicationUrl": "https://localhost:17045;http://localhost:15033",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development",
        "DOTNET_ENVIRONMENT": "Development",
        "DOTNET_DASHBOARD_OTLP_ENDPOINT_URL": "https://localhost:21188",
        "DOTNET_RESOURCE_SERVICE_ENDPOINT_URL": "https://localhost:22048"
      }
    }
  }
}
```

Оркестрация

Launch profiles precedence logic

Оркестрация

Launch profiles precedence logic

1. launchProfileName argument

Оркестрация

Launch profiles precedence logic

1. launchProfileName argument
2. Same name as the AppHost (*DOTNET_LAUNCH_PROFILE*)

Оркестрация

Launch profiles precedence logic

1. launchProfileName argument
2. Same name as the AppHost (*DOTNET_LAUNCH_PROFILE*)
3. First in *launchSettings.json*

Оркестрация

Launch profiles precedence logic

1. launchProfileName argument
2. Same name as the AppHost (*DOTNET_LAUNCH_PROFILE*)
3. First in *launchSettings.json*
4. Don’t use a launch profile

Оркестрация

Kestrel endpoint configuration

appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning",
      "Aspire.Hosting.Dcp": "Warning"
    }
  },
  "Kestrel": {
    "Endpoints": {
      "Https": {
        "Url": "https://*:5271"
      }
    }
  }
}
```

Оркестрация

Kestrel endpoint configuration

appsettings.json

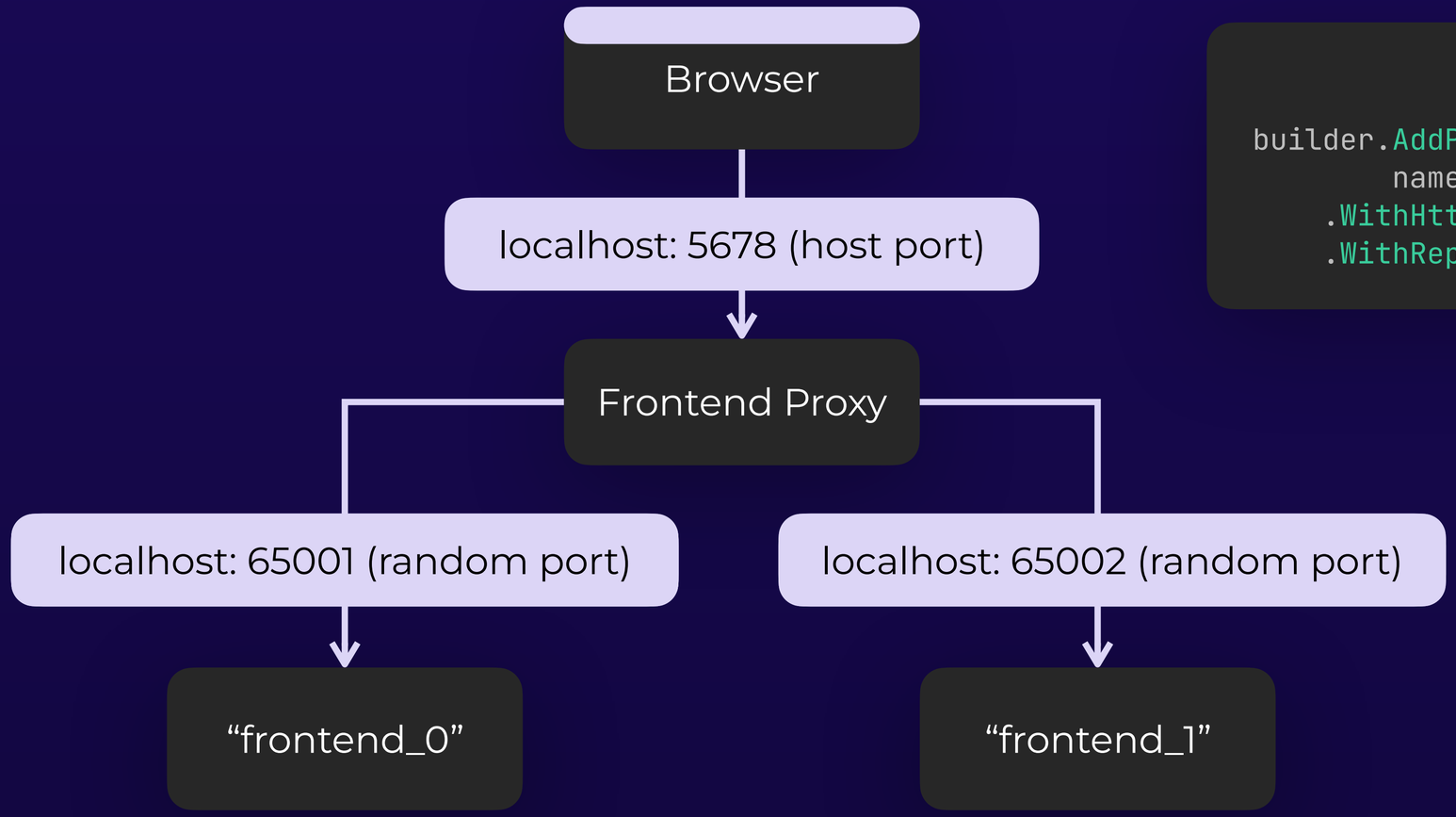
```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning",
      "Aspire.Hosting.Dcp": "Warning"
    }
  },
  "Kestrel": {
    "Endpoints": {
      "Https": {
        "Url": "https://*:5271"
      }
    }
  }
}
```

AppHost/Program.cs

```
builder.AddProject<Projects.AspireStarter_Web>(
        name: "webfrontend",
        configure: static project =>
        {
            project.ExcludeLaunchProfile = true;
            project.ExcludeKestrelEndpoints = false;
        })
    .WithHttpsEndpoint();
```

Оркестрация

Ports & proxies

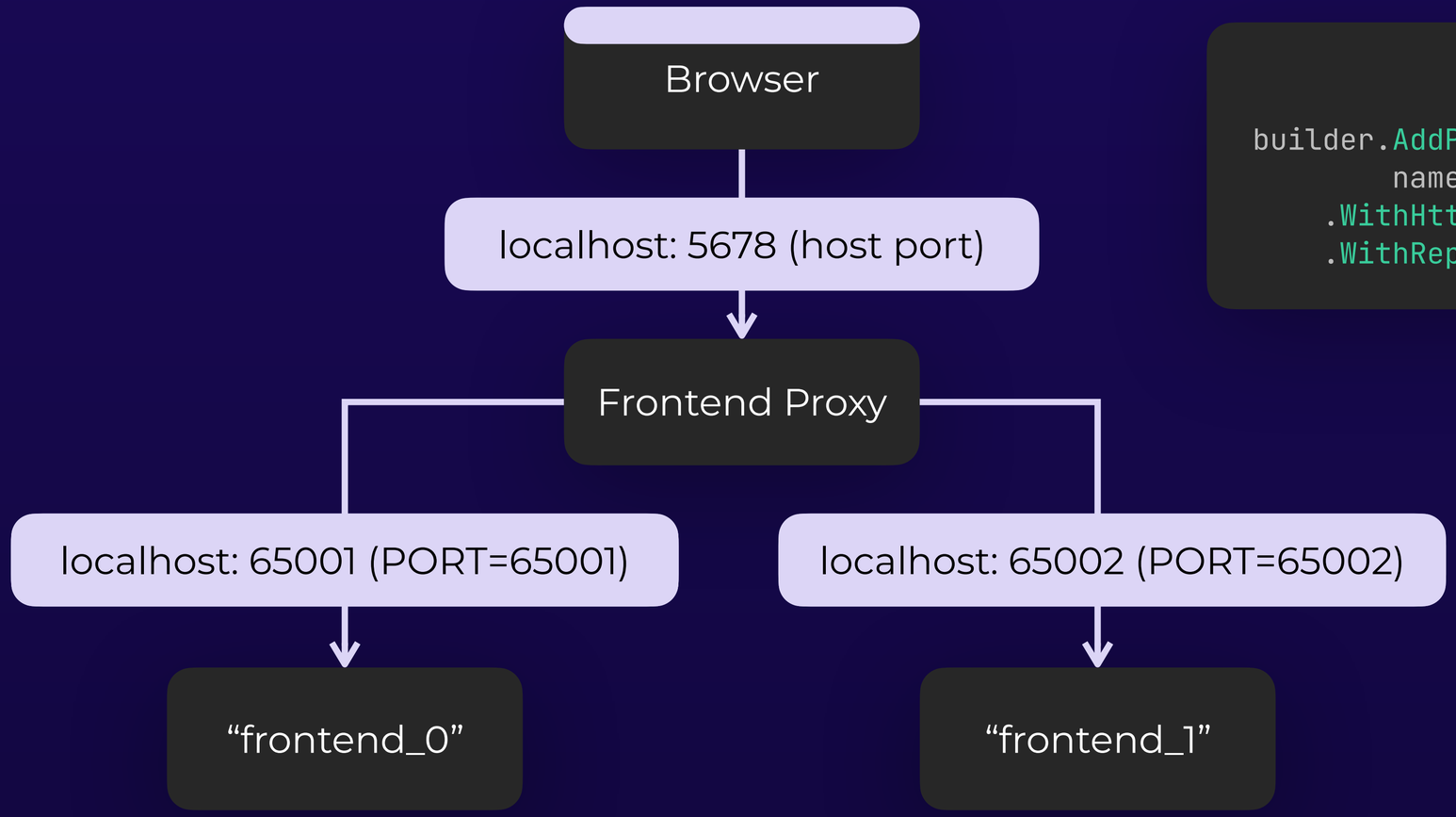


AppHost/Program.cs

```
builder.AddProject<Projects.AspireStarter_Web>(
        name: "webfrontend")
    .WithHttpEndpoint(port: 5678)
    .WithReplicas(2);
```

Оркестрация

Ports & proxies

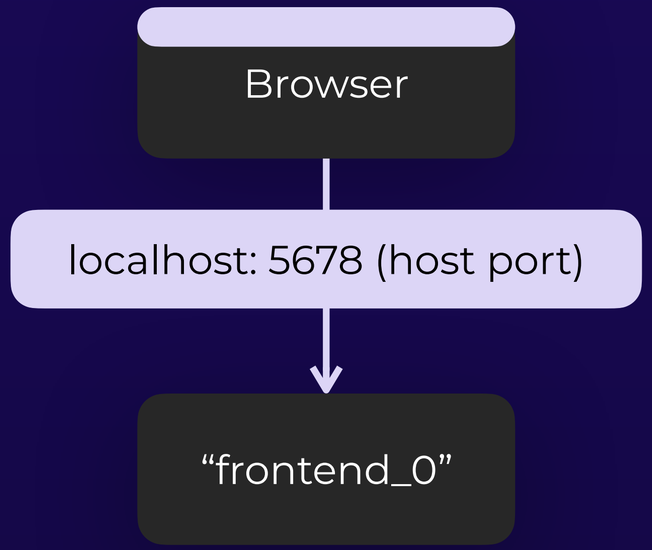


AppHost/Program.cs

```
builder.AddProject<Projects.AspireStarter_Web>(
        name: "webfrontend")
    .WithHttpEndpoint(port: 5678, env: "PORT")
    .WithReplicas(2);
```

Оркестрация

Ports & proxies

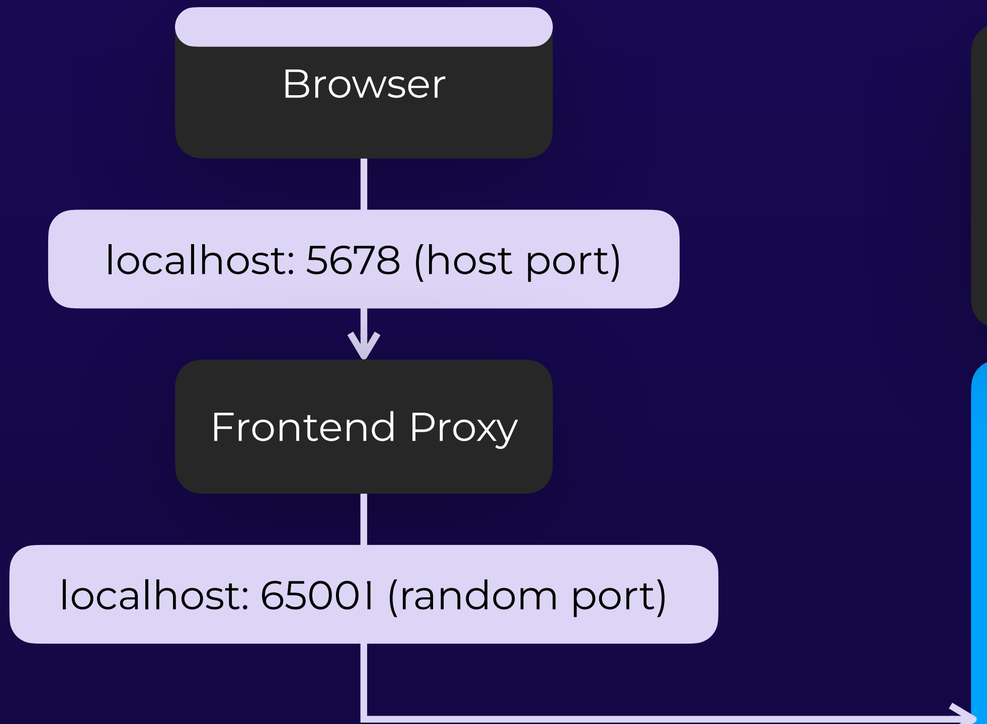


AppHost/Program.cs

```
builder.AddProject<Projects.AspireStarter_Web>(
        name: "webfrontend")
    .WithHttpEndpoint(port: 5678, isProxied: false);
```

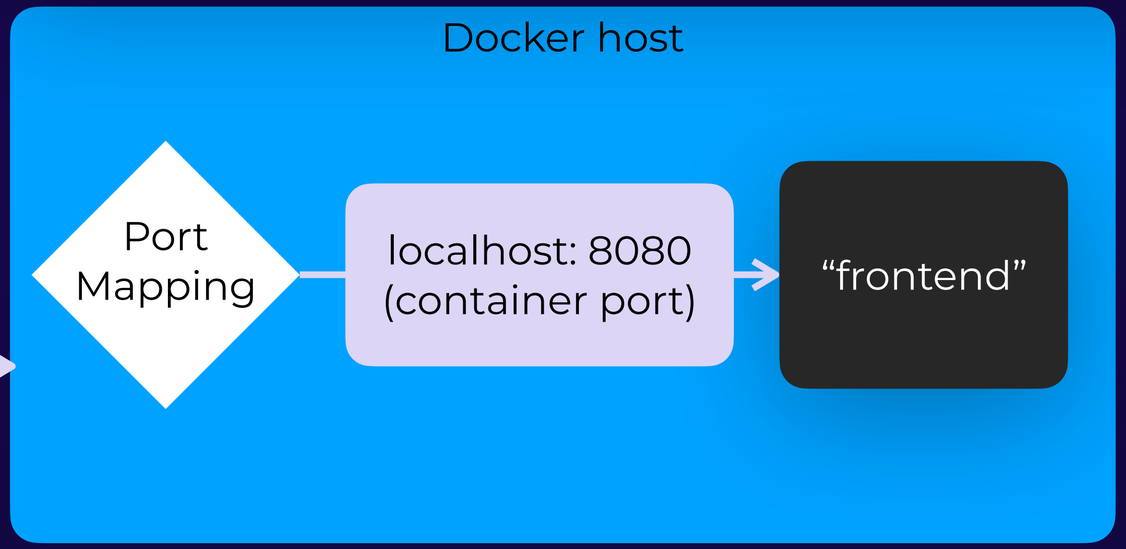
Оркестрация

Ports & proxies



AppHost/Program.cs

```
builder.AddContainer("frontend", "mcr.microsoft.com/dotnet/samples")
    .WithHttpEndpoint(port: 5678, targetPort: 8080);
```



Оркестрация

WithReference

Аргумент	Создаваемая переменная окружения

Оркестрация

WithReference

Аргумент	Создаваемая переменная окружения
ResourceBuilder<IResourceWithConnectionStrings>	

Оркестрация

WithReference

Аргумент

Создаваемая переменная окружения

ResourceBuilder<IResourceWithConnectionStrings>

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

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

Оркестрация

WithReference

Аргумент	Создаваемая переменная окружения
ResourceBuilder<IResourceWithConnectionStrings>	ConnectionStrings__cache="localhost:62354"

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

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

Оркестрация

WithReference

Аргумент	Создаваемая переменная окружения
ResourceBuilder<IResourceWithConnectionStrings>	ConnectionStrings__cache="localhost:62354"

Оркестрация

WithReference

Аргумент	Создаваемая переменная окружения
ResourceBuilder<IResourceWithConnectionStrings>	ConnectionStrings__cache="localhost:62354"
ResourceBuilder<IResourceWithServiceDiscovery>	

Оркестрация

WithReference

Аргумент	Создаваемая переменная окружения
ResourceBuilder<IResourceWithConnectionStrings>	ConnectionStrings__cache="localhost:62354"
ResourceBuilder<IResourceWithServiceDiscovery>	

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

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

Оркестрация

WithReference

Аргумент	Создаваемая переменная окружения
ResourceBuilder<IResourceWithConnectionStrings>	ConnectionStrings__cache="localhost:62354"
ResourceBuilder<IResourceWithServiceDiscovery>	services__apiservice__http__0="http://localhost:5455" services__apiservice__https__0="https://localhost:7356"

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

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

Оркестрация

WithReference

Аргумент	Создаваемая переменная окружения
ResourceBuilder<IResourceWithConnectionStrings>	ConnectionStrings__cache="localhost:62354"
ResourceBuilder<IResourceWithServiceDiscovery>	services__apiservice__http__0="http://localhost:5455" services__apiservice__https__0="https://localhost:7356"

Оркестрация

WithReference

Аргумент	Создаваемая переменная окружения
ResourceBuilder<IResourceWithConnectionStrings>	ConnectionStrings__cache="localhost:62354"
ResourceBuilder<IResourceWithServiceDiscovery>	services__apiservice__http__0="http://localhost:5455" services__apiservice__https__0="https://localhost:7356"
EndpointReference	

Оркестрация

WithReference

Аргумент	Создаваемая переменная окружения
ResourceBuilder<IResourceWithConnectionStrings>	ConnectionStrings__cache="localhost:62354"
ResourceBuilder<IResourceWithServiceDiscovery>	services__apiservice__http__0="http://localhost:5455" services__apiservice__https__0="https://localhost:7356"
EndpointReference	

```
var customContainer = builder.AddContainer("myapp", "mycustomcontainer").WithHttpEndpoint(port: 9043, name: "endpoint");
var endpoint = customContainer.GetEndpoint("endpoint");
builder.AddProject<Projects.AspireStarter_Web>("webfrontend").WithReference(endpoint);
```

Оркестрация

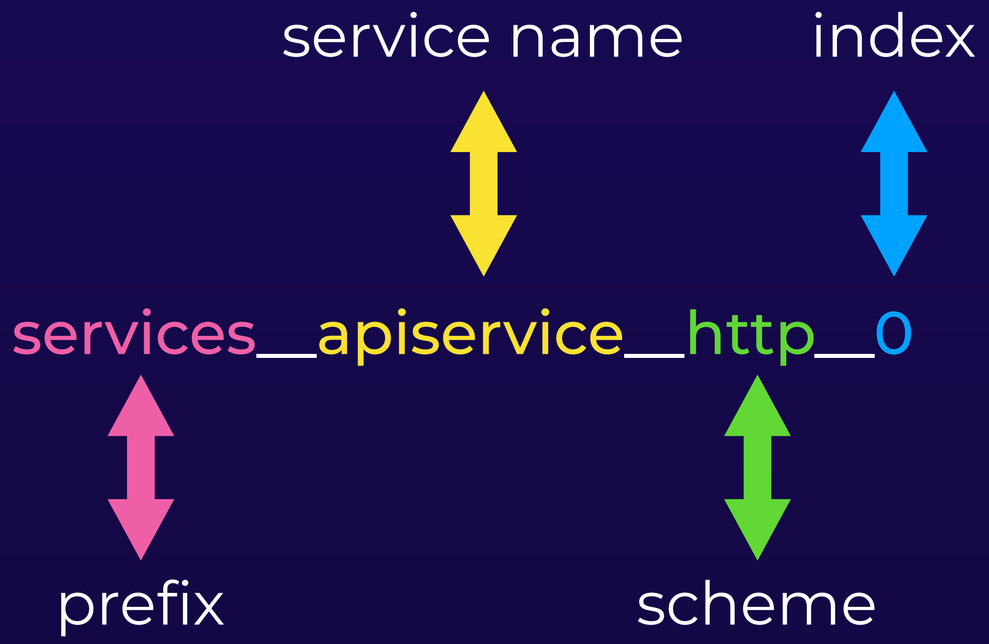
WithReference

Аргумент	Создаваемая переменная окружения
ResourceBuilder<IResourceWithConnectionStrings>	ConnectionStrings__cache="localhost:62354"
ResourceBuilder<IResourceWithServiceDiscovery>	services__apiservice__http__0="http://localhost:5455" services__apiservice__https__0="https://localhost:7356"
EndpointReference	services__myapp__endpoint__0="https://localhost:9043"

```
var customContainer = builder.AddContainer("myapp", "mycustomcontainer").WithHttpEndpoint(port: 9043, name: "endpoint");
var endpoint = customContainer.GetEndpoint("endpoint");
builder.AddProject<Projects.AspireStarter_Web>("webfrontend").WithReference(endpoint);
```

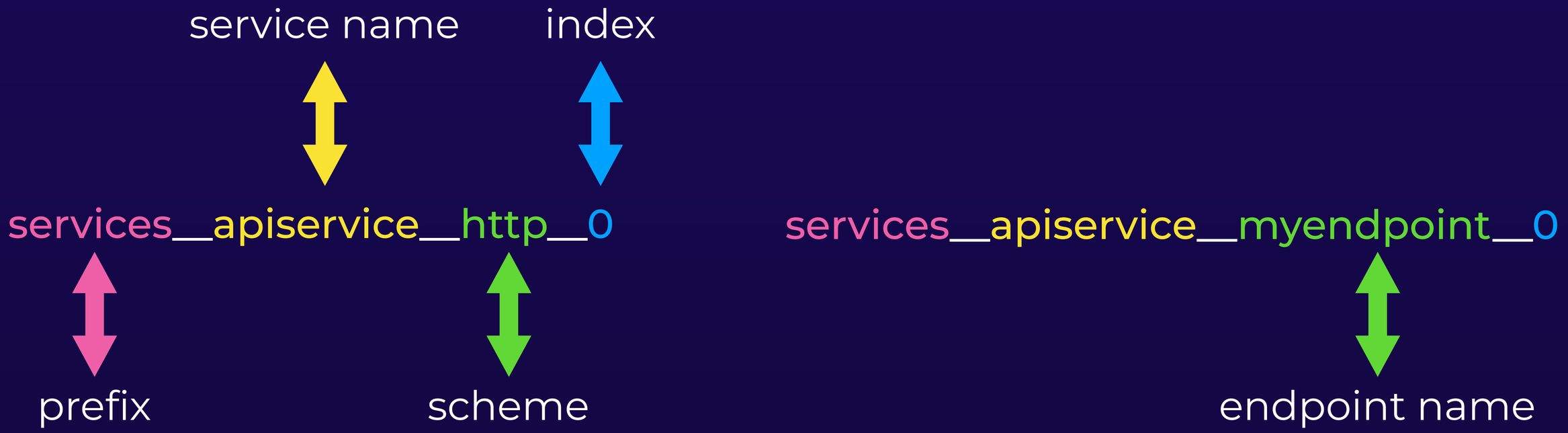
Оркестрация

Service endpoint environment variable format



Оркестрация

Service endpoint environment variable format



Оркестрация

External Parameters

AppHost/Program.cs

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

var parameter = builder.AddParameter("parameterName");

var apiService = builder.AddProject<Projects.AspireStarter_ApiService>("apiservice")
    .WithEnvironment("MY_PARAMETER", parameter);
```

AppHost/appsettings.json

```
"Parameters": {
  "parameterName": “parameter_value”
}
```

Оркестрация

External Parameters

AppHost/Program.cs

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

var secretParameter = builder.AddParameter("secretParameterName", secret: true);

var apiService = builder.AddProject<Projects.AspireStarter_ApiService>("apiservice")
    .WithEnvironment("MY_SECRET_PARAMETER", secretParameter);
```

AppHost/appsettings.json

```
"Parameters": {
  "secretParameterName": “super_secret_secret_value”
}
```

Оркестрация

External Parameters

AppHost/Program.cs

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

var secretParameter = builder.AddParameter("secretParameterName", secret: true);
var mssql = builder.AddConnectionString("mssql");

var apiService = builder.AddProject<Projects.AspireStarter_ApiService>("apiservice")
    .WithEnvironment("MY_SECRET_PARAMETER", secretParameter)
    .WithReference(mssql);
```

AppHost/appsettings.json

```
"Parameters": {
  "secretParameterName": “super_secret_secret_value”
},
"ConnectionStrings": {
  "mssql": "db-connection-string"
}
```

Оркестрация

Итого

Оркестрация

Итого

- Строго типизированная оркестрация (без YAML!)

Оркестрация

Итого

- Строго типизированная оркестрация (без YAML!)
- Единая точка входа

Оркестрация

Итого

- Строго типизированная оркестрация (без YAML!)
- Единая точка входа
- Service Discovery

Оркестрация

Итого

- Строго типизированная оркестрация (без YAML!)
- Единая точка входа
- Service Discovery
- Отказоустойчивость по-умолчанию

Оркестрация

Итого

- Строго типизированная оркестрация (без YAML!)
- Единая точка входа
- Service Discovery
- Отказоустойчивость по-умолчанию
- Health Checks

Оркестрация

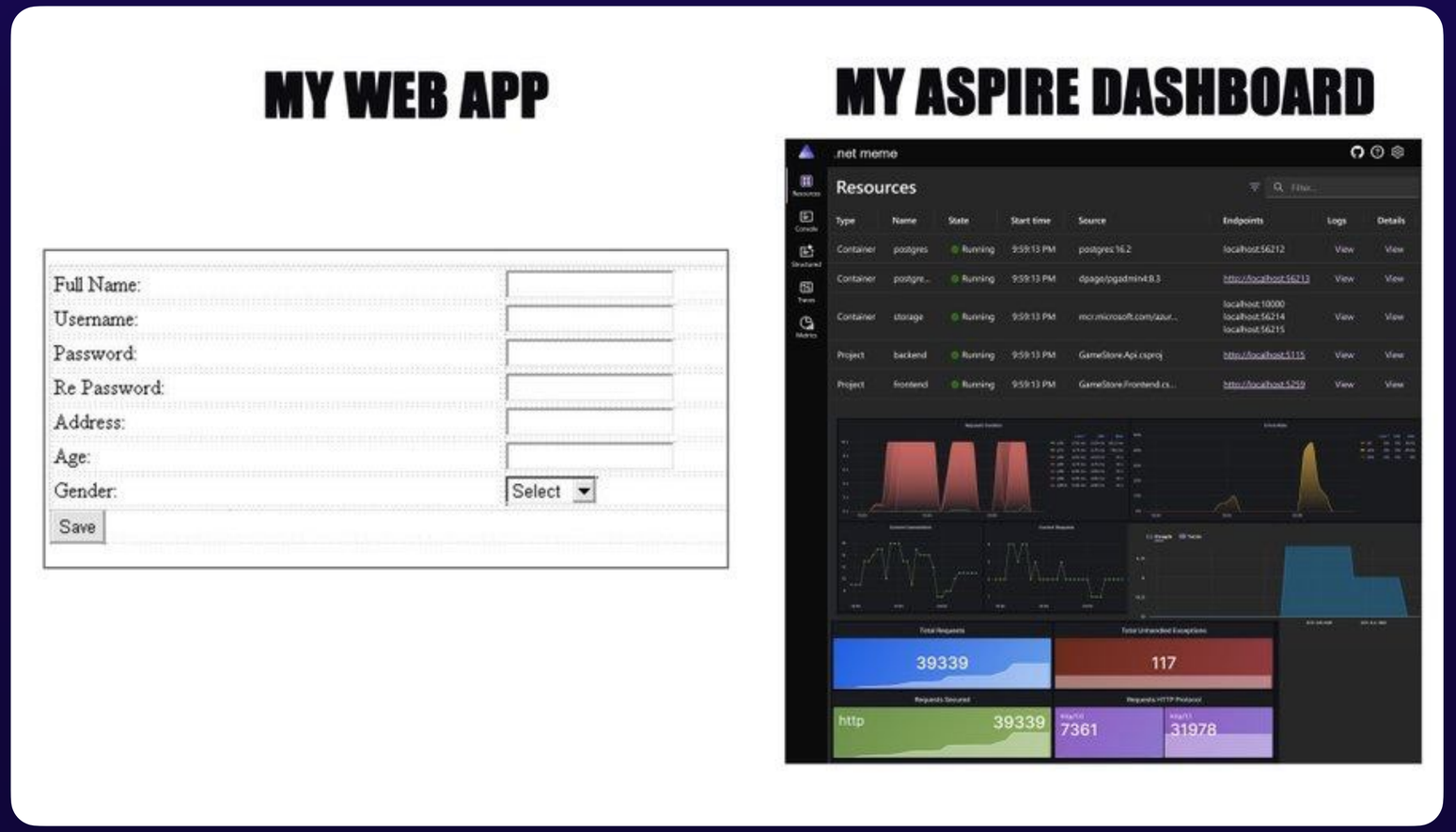
Итого

- Строго типизированная оркестрация (без YAML!)
- Единая точка входа
- Service Discovery
- Отказоустойчивость по-умолчанию
- Health Checks
- Сбор и экспорт телеметрии

Dashboard

Demo

- Resources
- Console Logs
- Structured Logs
- Tracing
- Metrics
- Health Checks



Dashboard

Итого

- Просмотр сервисов и их данных
- Подходит для всех типов сигналов
- Данные хранятся In-Memory
- Красивое...
- Существует как отдельный Telemetry Backend!

Dashboard Configuration

Variations

CLI

```
docker run --rm -it -p 18888:18888 -p 4317:18889 -d --name aspire-dashboard `
    -e DASHBOARD__TELEMETRYLIMITS__MAXLOGCOUNT='1000' `
    -e DASHBOARD__TELEMETRYLIMITS__MAXTRACECOUNT='1000' `
    -e DASHBOARD__TELEMETRYLIMITS__MAXMETRICSCOUNT='1000' `
    mcr.microsoft.com/dotnet/aspire-dashboard:8.0.0
```

DOTNET_DASHBOARD_CONFIG_FILE_PATH Value.json

```
{
  "Dashboard": {
    "TelemetryLimits": {
      "MaxLogCount": 1000,
      "MaxTraceCount": 1000,
      "MaxMetricsCount": 1000
    }
  }
}
```

Dashboard Configuration

Common

Dashboard Configuration

Common

Option	Default Value	Description

Dashboard Configuration

Common

Option	Default Value	Description
ASPNETCORE_URLS	http://localhost:18888	HTTP endpoints which the dashboard frontend is served

Dashboard Configuration

Common

Option	Default Value	Description
ASPNETCORE_URLS	http://localhost:18888	HTTP endpoints which the dashboard frontend is served
DOTNET_DASHBOARD_OTLP_ENDPOINT_URL	http://localhost:18889	The OTLP/gRPC endpoint

Dashboard Configuration

Common

Option	Default Value	Description
ASPNETCORE_URLS	http://localhost:18888	HTTP endpoints which the dashboard frontend is served
DOTNET_DASHBOARD_OTLP_ENDPOINT_URL	http://localhost:18889	The OTLP/gRPC endpoint
DOTNET_DASHBOARD_OTLP_HTTP_ENDPOINT_URL	http://localhost:18890	The OTLP/HTTP endpoint. This endpoint hosts an OTLP service and receives telemetry using Protobuf over HTTP

Dashboard Configuration

Common

Option	Default Value	Description
ASPNETCORE_URLS	http://localhost:18888	HTTP endpoints which the dashboard frontend is served
DOTNET_DASHBOARD_OTLP_ENDPOINT_URL	http://localhost:18889	The OTLP/gRPC endpoint
DOTNET_DASHBOARD_OTLP_HTTP_ENDPOINT_URL	http://localhost:18890	The OTLP/HTTP endpoint. This endpoint hosts an OTLP service and receives telemetry using Protobuf over HTTP
DOTNET_DASHBOARD_UNSECURED_ALLOW_ANONYMOUS	FALSE	Configures the dashboard to not use authentication and accepts anonymous access

Dashboard Configuration

Common

Option	Default Value	Description
ASPNETCORE_URLS	http://localhost:18888	HTTP endpoints which the dashboard frontend is served
DOTNET_DASHBOARD_OTLP_ENDPOINT_URL	http://localhost:18889	The OTLP/gRPC endpoint
DOTNET_DASHBOARD_OTLP_HTTP_ENDPOINT_URL	http://localhost:18890	The OTLP/HTTP endpoint. This endpoint hosts an OTLP service and receives telemetry using Protobuf over HTTP
DOTNET_DASHBOARD_UNSECURED_ALLOW_ANONYMOUS	FALSE	Configures the dashboard to not use authentication and accepts anonymous access
DOTNET_DASHBOARD_CONFIG_FILE_PATH	null	The path for a JSON configuration file

Dashboard Configuration

Auth & More

Dashboard Configuration

Auth & More

Option	Default Value	Available Values

Dashboard Configuration

Auth & More

Option	Default Value	Available Values
Dashboard:Frontend:AuthMode	BrowserToken	BrowserToken | OpenIdConnect | Unsecured

Dashboard Configuration

Auth & More

Option	Default Value	Available Values
Dashboard:Frontend:AuthMode	BrowserToken	BrowserToken | OpenIdConnect | Unsecured
Dashboard:Otlp:AuthMode	Unsecured	ApiKey | Certificate | Unsecured

Dashboard Configuration

Auth & More

Option	Default Value	Available Values
Dashboard:Frontend:AuthMode	BrowserToken	BrowserToken | OpenIdConnect | Unsecured
Dashboard:Otlp:AuthMode	Unsecured	ApiKey | Certificate | Unsecured
Dashboard:ResourceServiceClient:Url	null	string

Dashboard Configuration

Auth & More

Option	Default Value	Available Values
Dashboard:Frontend:AuthMode	BrowserToken	BrowserToken | OpenIdConnect | Unsecured
Dashboard:Otlp:AuthMode	Unsecured	ApiKey | Certificate | Unsecured
Dashboard:ResourceServiceClient:Url	null	string
Dashboard:ResourceServiceClient:AuthMode	null	ApiKey | Certificate | Unsecured

Custom Resource

Custom Resource

1. A custom resource type that implements IResource;

Custom Resource

1. A custom resource type that implements IResource;
2. An extension method for IDistributedApplicationBuilder named Add{CustomResource}.

Custom Resource

Preparation

Custom Resource

Preparation

```
dotnet new classlib -o MailDev.Hosting
dotnet add ./MailDev.Hosting/MailDev.Hosting.csproj package Aspire.Hosting --version 8.0.0
dotnet add ./MailDevResource.AppHost/MailDevResource.AppHost.csproj reference ./MailDev.Hosting/MailDev.Hosting.csproj
dotnet sln ./MailDevResource.sln add ./MailDev.Hosting/MailDev.Hosting.csproj
```

Custom Resource

Creation

MailDev.Hosting/MailDevResource.cs

```
public sealed class MailDevResource(string name) : ContainerResource(name), IResourceWithConnectionString
{
    internal const string SmtpEndpointName = "smtp";
    internal const string HttpEndpointName = "http";

    private EndpointReference? _smtpReference;

    public EndpointReference SmtpEndpoint ⇒
        _smtpReference ??= new(this, SmtpEndpointName);

    public ReferenceExpression ConnectionStringExpression ⇒
        ReferenceExpression.Create(
            $"smtp://{SmtpEndpoint.Property(EndpointProperty.Host)}:{SmtpEndpoint.Property(EndpointProperty.Port)}"
        );
}
```

Custom Resource

Injection

MailDev.Hosting/MailDevResourceBuilderExtensions.cs

```
public static class MailDevResourceBuilderExtensions
{
    public static IResourceBuilder<MailDevResource> AddMailDev(
        this IDistributedApplicationBuilder builder,
        string name,
        int? httpPort = null,
        int? smtpPort = null)
    {
        var resource = new MailDevResource(name);

        return builder.AddResource(resource)
                      .WithImage(MailDevContainerImageTags.Image)
                      .WithImageRegistry(MailDevContainerImageTags.Registry)
                      .WithImageTag(MailDevContainerImageTags.Tag)
                      .WithHttpEndpoint(
                          targetPort: 1080,
                          port: httpPort,
                          name: MailDevResource.HttpEndpointName)
                      .WithEndpoint(
                          targetPort: 1025,
                          port: smtpPort,
                          name: MailDevResource.SmtpEndpointName);
    }
}
```

Custom Resource

Injection

MailDev.Hosting/MailDevResourceBuilderExtensions.cs

```
internal static class MailDevContainerImageTags
{
    internal const string Registry = "docker.io";

    internal const string Image = "maildev/maildev";

    internal const string Tag = "2.0.2";
}
```

Custom Resource

Orchestration

AppHost/Program.cs

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

var cache = builder.AddRedis("cache");
var mailDev = builder.AddMailDev("maildev");
var apiService = builder.AddProject<Projects.AspireStarter_ApiService>("apiservice");

builder.AddProject<Projects.AspireStarter_Web>("webfrontend")
    .WithExternalHttpEndpoints()
    .WithReference(cache)
    .WithReference(apiService)
    .WithReference(mailDev);

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

Custom Resource

Usage

Web/Program.cs

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

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

/* … */

builder.Services.AddSingleton<SmtpClient>(sp =>
{
    var smtpUri = new Uri(builder.Configuration.GetConnectionString("maildev")!);

    var smtpClient = new SmtpClient(smtpUri.Host, smtpUri.Port);

    return smtpClient;
});
```

Custom Component

Preparation

Custom Integration

Preparation

Custom Integration

Preparation

```
dotnet new classlib -o MailDev.Client.MailKit
dotnet sln ./MailDevResource.sln add ./MailDev.Client.MailKit/MailDev.Client.MailKit.csproj
```

Custom Integration

Preparation

```
dotnet new classlib -o MailDev.Client.MailKit

dotnet sln ./MailDevResource.sln add ./MailDev.Client.MailKit/MailDev.Client.MailKit.csproj
```

MailDev.Client.MailKit.csproj

```
<ItemGroup>
  <PackageReference Include="MailKit" Version="4.7.0" />
  <PackageReference Include="Microsoft.Extensions.Configuration.Binder" Version="8.0.2" />
  <PackageReference Include="Microsoft.Extensions.Resilience" Version="8.7.0" />
  <PackageReference Include="Microsoft.Extensions.Hosting.Abstractions" Version="8.0.0" />
  <PackageReference Include="Microsoft.Extensions.Diagnostics.HealthChecks" Version="8.0.7" />
  <PackageReference Include="OpenTelemetry.Extensions.Hosting" Version="1.9.0" />
</ItemGroup>
```

Custom Integration

Configuration

```
public sealed class MailKitClientSettings
{
    internal const string DefaultConfigSectionName = "MailKit:Client";

    public Uri? Endpoint { get; set; }
    public bool DisableHealthChecks { get; set; }
    public bool DisableTracing { get; set; }
    public bool DisableMetrics { get; set; }

    internal void ParseConnectionString(string? connectionString)
    {
        /* */

        Endpoint = uri;
    }
}
```

Custom Integration

Creation

MailKitClientFactory.cs

```
public sealed class MailKitClientFactory(MailKitClientSettings settings) : IDisposable
{
    private readonly SemaphoreSlim _semaphore = new(1, 1);

    private SmtpClient? _client;

    public async Task<ISmtpClient> GetSmtpClientAsync(CancellationToken cancellationToken = default)
    {
        await _semaphore.WaitAsync(cancellationToken);

        try
            if (_client is null)
            {
                _client = new SmtpClient();

                await _client.ConnectAsync(settings.Endpoint, cancellationToken)
                    .ConfigureAwait(false);
            }
        finally
            _semaphore.Release();

        return _client;
    }
```

Custom Integration

Creation

MailKitClientFactory.cs

```
    public async Task<ISmtpClient> GetSmtpClientAsync(CancellationToken cancellationToken = default)
    {
        await _semaphore.WaitAsync(cancellationToken);

        try
            if (_client is null)
            {
                _client = new SmtpClient();

                await _client.ConnectAsync(settings.Endpoint, cancellationToken)
                    .ConfigureAwait(false);
            }
        finally
            _semaphore.Release();

        return _client;
    }


    public void Dispose()
    {
        _client?.Dispose();
        _semaphore.Dispose();
    }
}
```

Custom Integration

Injection

MailKitExtensions.cs

```
public static class MailKitExtensions
{
    public static void AddMailKitClient(this IHostApplicationBuilder builder, string connectionName,
        Action<MailKitClientSettings>? configureSettings = null)
    {
        AddMailKitClient(builder, MailKitClientSettings.DefaultConfigSectionName, configureSettings,
            connectionName, serviceKey: null);
    }

    public static void AddKeyedMailKitClient(this IHostApplicationBuilder builder, string name,
        Action<MailKitClientSettings>? configureSettings = null)
    {
        ArgumentNullException.ThrowIfNull(name);
        AddMailKitClient(builder, $"{MailKitClientSettings.DefaultConfigSectionName}:{name}",
            configureSettings, connectionName: name, serviceKey: name);
    }

    private static void AddMailKitClient(/* parameters */);
}
```

Custom Integration

Injection

MailKitExtensions.cs

```
private static void AddMailKitClient(
    this IHostApplicationBuilder builder,
    string configurationSectionName,
    Action<MailKitClientSettings>? configureSettings,
    string connectionName,
    object? serviceKey)
{
    /* Inject */
    /* Configure */
}
```

Custom Integration

Injection

MailKitExtensions.cs

```
private static void AddMailKitClient(/* parameters */)
{
    ArgumentNullException.ThrowIfNull(builder);

    var settings = new MailKitClientSettings();

    builder.Configuration
           .GetSection(configurationSectionName)
           .Bind(settings);

    if (builder.Configuration.GetConnectionString(connectionName) is string connectionString)
        settings.ParseConnectionString(connectionString);

    configureSettings?.Invoke(settings);

    if (serviceKey is null)
        builder.Services.AddScoped(_ => new MailKitClientFactory(settings));
    else
        builder.Services.AddKeyedScoped(serviceKey, (sp, key) => new MailKitClientFactory(settings));

    /* Configure */
}
```

Custom Integration

Injection

MailKitExtensions.cs

```
private static void AddMailKitClient(/* parameters */)
{
    /* Inject */
    if (settings.DisableHealthChecks is false)
        builder.Services.AddHealthChecks()
            .AddCheck<MailKitHealthCheck>(
                name: serviceKey is null ? "MailKit" : $"MailKit_{connectionName}",
                failureStatus: default,
                tags: []);

    if (settings.DisableTracing is false)
        builder.Services.AddOpenTelemetry()
            .WithTracing(traceBuilder => traceBuilder.AddSource(Telemetry.SmtpClient.ActivitySourceName));

    if (settings.DisableMetrics is false)
    {
        Telemetry.SmtpClient.Configure(); // Required by MailKit to enable metrics
        builder.Services.AddOpenTelemetry()
            .WithMetrics(metricsBuilder => metricsBuilder.AddMeter(Telemetry.SmtpClient.MeterName));
    }
}
```

Custom Integration

Injection

```
tionName}",


.SmtpClient.ActivitySourceName));


e metrics

etry.SmtpClient.MeterName));
```

MailKitHealthCheck.cs

```
internal sealed class MailKitHealthCheck(MailKitClientFactory factory) : IHealthCheck
{
    public async Task<HealthCheckResult> CheckHealthAsync(
        HealthCheckContext context,
        CancellationToken cancellationToken = default)
    {
        try
        {
            _ = await factory.GetSmtpClientAsync(cancellationToken);

            return HealthCheckResult.Healthy();
        }
        catch (Exception ex)
        {
            return HealthCheckResult.Unhealthy(exception: ex);
        }
    }
}
```

Custom Integration

Injection

MailKitExtensions.cs

```
private static void AddMailKitClient(/* parameters */)
{
    /* Inject */
    if (settings.DisableHealthChecks is false)
        builder.Services.AddHealthChecks()
            .AddCheck<MailKitHealthCheck>(
                name: serviceKey is null ? "MailKit" : $"MailKit_{connectionName}",
                failureStatus: default,
                tags: []);

    if (settings.DisableTracing is false)
        builder.Services.AddOpenTelemetry()
            .WithTracing(traceBuilder => traceBuilder.AddSource(Telemetry.SmtpClient.ActivitySourceName));

    if (settings.DisableMetrics is false)
    {
        Telemetry.SmtpClient.Configure(); // Required by MailKit to enable metrics
        builder.Services.AddOpenTelemetry()
            .WithMetrics(metricsBuilder => metricsBuilder.AddMeter(Telemetry.SmtpClient.MeterName));
    }
}
```

Custom Integration

Usage

Web/Program.cs

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

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

/* … */

builder.AddMailKitClient("maildev");
```

Custom Integration

Usage

Web/Program.cs

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

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

/* … */

builder.AddMailKitClient("maildev");
```

Integrations

Packages

Integrations

Packages

- Hosting Package:
  - 📦 Aspire.Hosting.Redis

Integrations

Packages

- Hosting Package:
  - 📦 Aspire.Hosting.Redis
- Integration Packages:
  - 📦 Aspire.StackExchange.Redis
  - 📦 Aspire.StackExchange.Redis.DistributedCaching
  - 📦 Aspire.StackExchange.Redis.OutputCaching

Integrations

Available

https://learn.microsoft.com/en-us/dotnet/aspire/fundamentals/integrations-overview?tabs=dotnet-cli#available-components

Integrations

Available

https://learn.microsoft.com/en-us/dotnet/aspire/fundamentals/integrations-overview?tabs=dotnet-cli#available-components

Integrations

Requirements

https://github.com/dotnet/aspire/blob/f38b6cba86942ad1c45fc04fe7170f0fd4ba7c0b/src/Components/Aspire_Components_Progress.md#net-aspire-component-requirements

Integrations

Requirements

- README

https://github.com/dotnet/aspire/blob/f38b6cba86942ad1c45fc04fe7170f0fd4ba7c0b/src/Components/Aspire_Components_Progress.md#net-aspire-component-requirements

Integrations

Requirements

- README
- Public API

https://github.com/dotnet/aspire/blob/f38b6cba86942ad1c45fc04fe7170f0fd4ba7c0b/src/Components/Aspire_Components_Progress.md#net-aspire-component-requirements

Integrations

Requirements

- README
- Public API
- Json Schema/Configuration

https://github.com/dotnet/aspire/blob/f38b6cba86942ad1c45fc04fe7170f0fd4ba7c0b/src/Components/Aspire_Components_Progress.md#net-aspire-component-requirements

Integrations

Requirements

- README
- Public API
- Json Schema/Configuration
- DI Services (Example)

https://github.com/dotnet/aspire/blob/f38b6cba86942ad1c45fc04fe7170f0fd4ba7c0b/src/Components/Aspire_Components_Progress.md#net-aspire-component-requirements

Integrations

Requirements

- README
- Public API
- Json Schema/Configuration
- DI Services (Example)
- Health Checks

https://github.com/dotnet/aspire/blob/f38b6cba86942ad1c45fc04fe7170f0fd4ba7c0b/src/Components/Aspire_Components_Progress.md#net-aspire-component-requirements

Integrations

Requirements

- README
- Public API
- Json Schema/Configuration
- DI Services (Example)
- Health Checks
- Telemetry (Logging, Tracing, Metrics)

https://github.com/dotnet/aspire/blob/f38b6cba86942ad1c45fc04fe7170f0fd4ba7c0b/src/Components/Aspire_Components_Progress.md#net-aspire-component-requirements

Integrations

Итого

https://youtu.be/istmgTDtSSQ?si=CCIqtp_TUO72AfQy

Integrations

Итого

★ Подключают сервисы

https://youtu.be/istmgTDtSSQ?si=CCIqtp_TUO72AfQy

Integrations

Итого

★ Подключают сервисы

★ Внедряют привычные библиотеки

- Не меняют API взаимодействия

https://youtu.be/istmgTDtSSQ?si=CCIqtp_TUO72AfQy

Integrations

Итого

★ Подключают сервисы

★ Внедряют привычные библиотеки

- Не меняют API взаимодействия

★ Автоинструментируют

https://youtu.be/istmgTDtSSQ?si=CCIqtp_TUO72AfQy

Integrations

Итого

★ Подключают сервисы

★ Внедряют привычные библиотеки

- Не меняют API взаимодействия

★ Автоинструментируют

★ Настраивают отказоустойчивость

https://youtu.be/istmgTDtSSQ?si=CCIqtp_TUO72AfQy

Integrations

Итого

★ Подключают сервисы

★ Внедряют привычные библиотеки

- Не меняют API взаимодействия

★ Автоинструментируют

★ Настраивают отказоустойчивость

★ Добавляют Health Checks

https://youtu.be/istmgTDtSSQ?si=CCIqtp_TUO72AfQy

Integrations

Итого

★ Подключают сервисы

★ Внедряют привычные библиотеки

- Не меняют API взаимодействия

★ Автоинструментируют

★ Настраивают отказоустойчивость

★ Добавляют Health Checks

★ Предоставляют гибкость

https://youtu.be/istmgTDtSSQ?si=CCIqtp_TUO72AfQy

Integrations

Итого

Делают рутину за нас

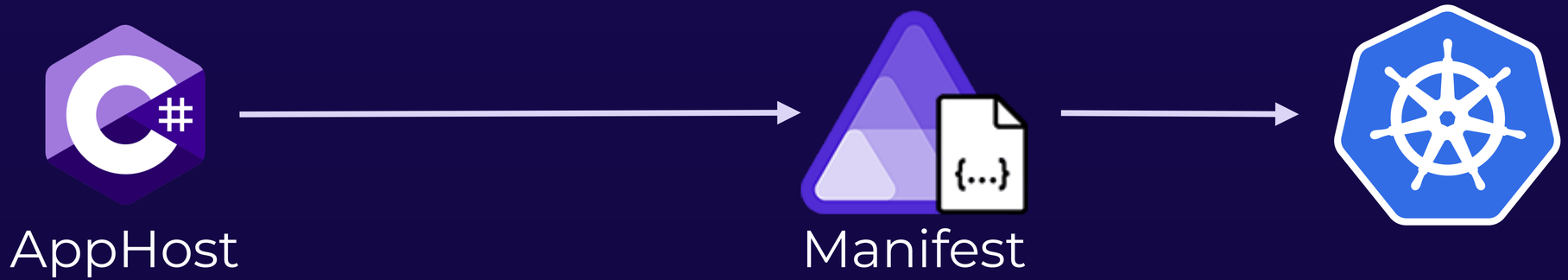
https://youtu.be/istmgTDtSSQ?si=CCIqtp_TUO72AfQy

Deployment

Deployment

Generate a Manifest

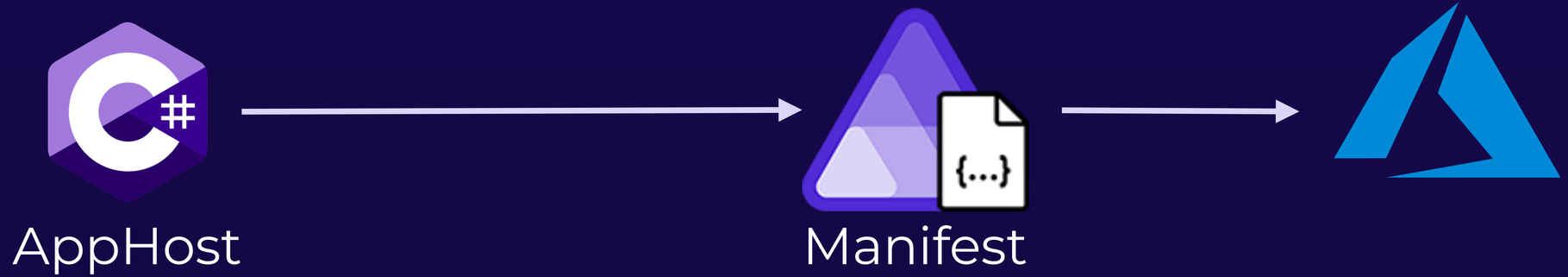
```
dotnet run --publisher manifest --output-path ../aspire-manifest.json
```



Deployment

Generate a Manifest

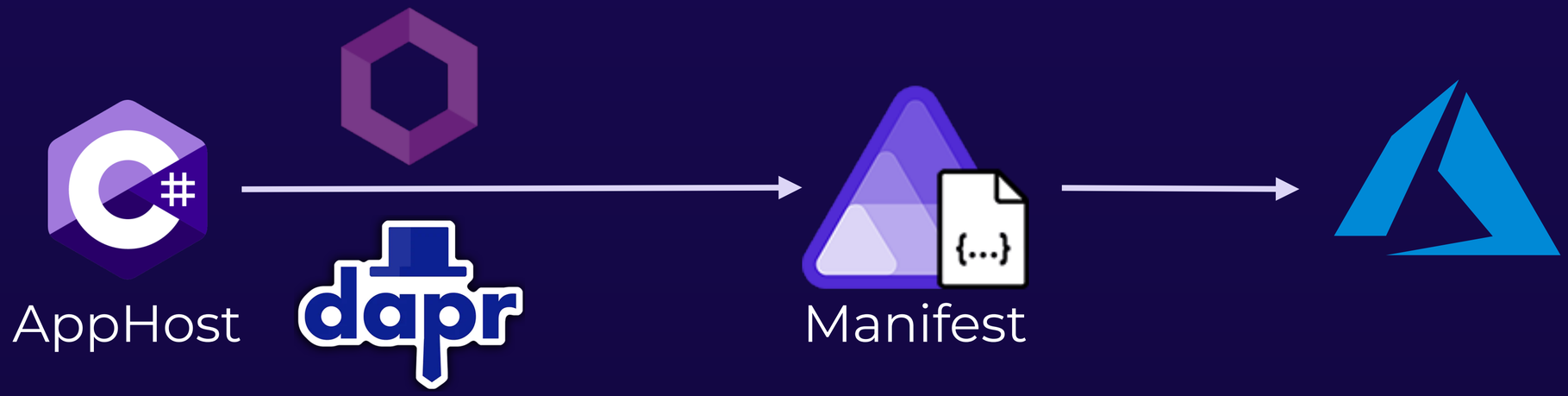
```
dotnet run --publisher manifest --output-path ../aspire-manifest.json
```



Deployment

Generate a Manifest

```
dotnet run --publisher manifest --output-path ../aspire-manifest.json
```



Deployment

Azure Developer CLI (azd)

Deployment

Azure Developer CLI (azd)

azd init

Deployment

Azure Developer CLI (azd)

azd init

How do you want to initialize your app?
1. Use code in the current directory
2. Select a template

Deployment

Azure Developer CLI (azd)

azd init

How do you want to initialize your app?
1. Use code in the current directory
2. Select a template

Use code in the current directory

Deployment

Azure Developer CLI (azd)

Use code in the current directory

Deployment

Azure Developer CLI (azd)

Use code in the current directory

Detected services:
.NET (Aspire)
Detected in: /Path/To/Project/AspireStarter.AppHost.csproj

Deployment

Azure Developer CLI (azd)

Use code in the current directory

Detected services:
.NET (Aspire)
Detected in: /Path/To/Project/AspireStarter.AppHost.csproj

Confirm and continue initializing my app

Deployment

Azure Developer CLI (azd)

Use code in the current directory

Detected services:
.NET (Aspire)
Detected in: /Path/To/Project/AspireStarter.AppHost.csproj

Confirm and continue initializing my app

Enter a new environment name:

Deployment

Azure Developer CLI (azd)

Enter a new environment name:

Deployment

Azure Developer CLI (azd)

Enter a new environment name:

prod

Deployment

Azure Developer CLI (azd)

prod

Enter a new environment name:

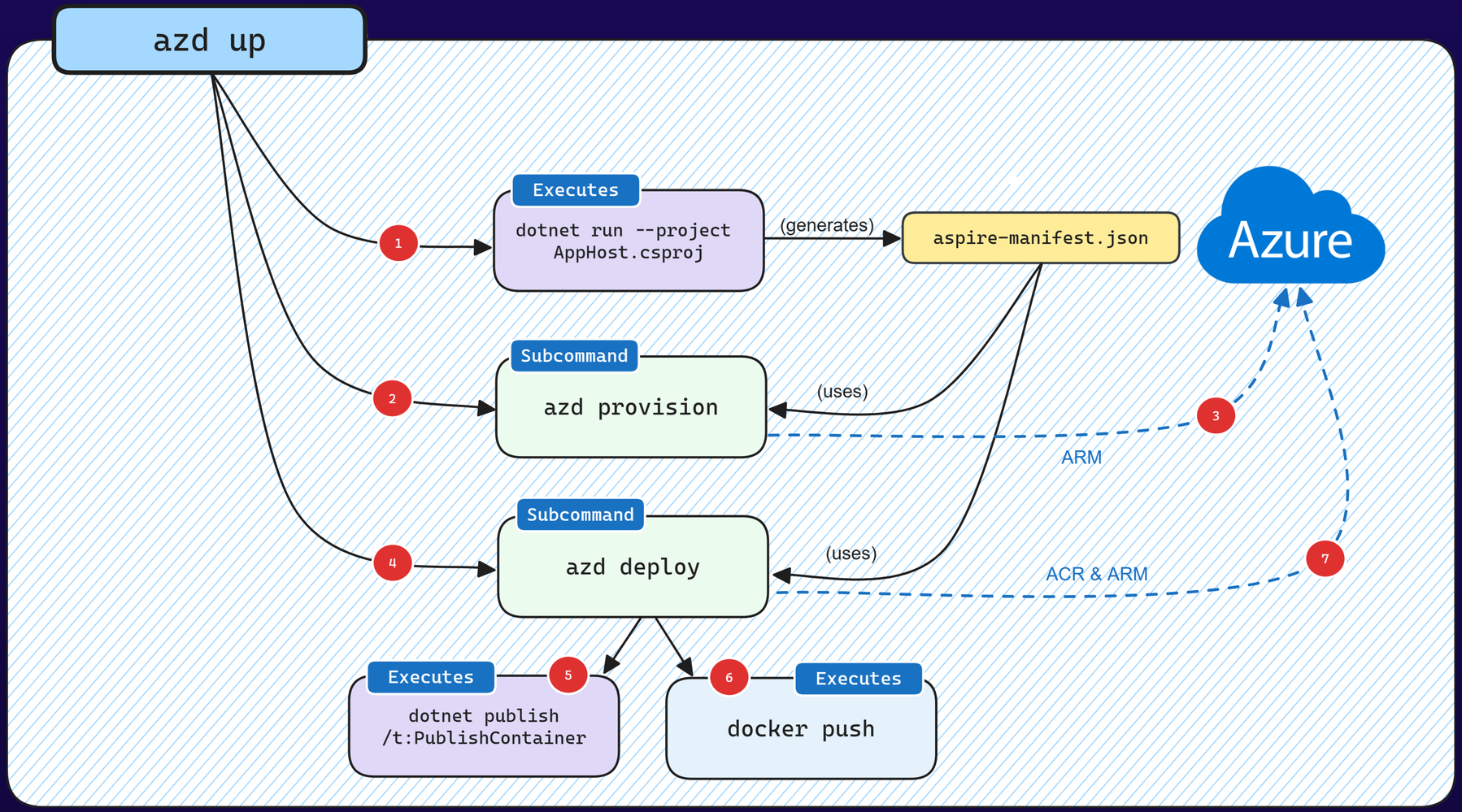
Generating files to run your app on Azure:

(✓) Done: Generating ./azure.yaml
(✓) Done: Generating ./next-steps.md

SUCCESS: Your app is ready for the cloud!

Deployment

Azure Developer CLI (azd)



Deployment

Aspir8

Deployment

Aspir8

1. Установка инструмента глобально

```
dotnet tool install -g aspirate --prerelease
```

Deployment

Aspir8

1. Установка инструмента глобально

```
dotnet tool install -g aspirate --prerelease
```

2. Инициализация и настройка

```
aspirate init
```

Deployment

Aspir8

1. Установка инструмента глобально

```
dotnet tool install -g aspirate --prerelease
```

2. Инициализация и настройка

```
aspirate init
```

3. Сборка проектов и подготовка контейнеров в *manifest.json*

```
aspirate build
```

Deployment

Aspir8

1. Установка инструмента глобально
2. Инициализация и настройка
3. Сборка проектов и подготовка контейнеров в *manifest.json*
4. Генерация compose/k8s манифеста или helm чарта

```
dotnet tool install -g aspirate --prerelease
```

```
aspirate init
```

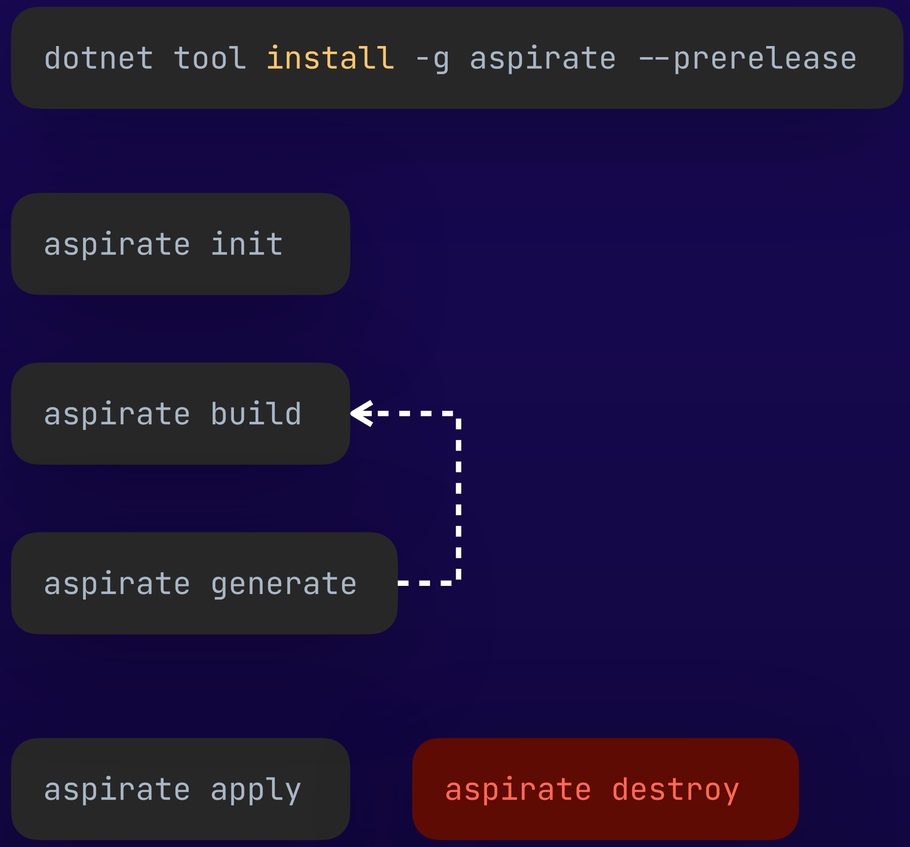
```
aspirate build
```

```
aspirate generate
```

Deployment

Aspir8

1. Установка инструмента глобально
2. Инициализация и настройка
3. Сборка проектов и подготовка контейнеров в *manifest.json*
4. Генерация compose/k8s манифеста или helm чарта
5. Заполнение переменных и применение манифеста



Deployment

Aspir8

Deployment

Aspir8

Запуск непосредственно
из директории в кластере
из KUBERCONFIG

```
aspirate run
```

Deployment

Aspir8

Запуск непосредственно
из директории в кластере
из KUBERCONFIG

```
aspirate run
```

Очистка пространства имён

```
aspirate stop
```

Deep Dive

dotnet run

- Строит модель приложения по коду
- Запускает экземпляр Developer Control Plane
- Просит DCP запустить ресурсы из модели
- Слушает события DCP и получает телеметрию ресурсов
- Запускает Dashboard

Deep Dive

dotnet run

- Строит модель приложения по коду
- Запускает экземпляр Developer Control Plane
- Просит DCP запустить ресурсы из модели
- Слушает события DCP и получает телеметрию ресурсов
- Запускает Dashboard

Running Processes
dcp
dcpctrl
dcpproc

Deep Dive

Microsoft Developer Control Plane (DCP)

- ★ Взаимодействовать с Docker и запускать контейнеры
- ★ Взаимодействовать с операционной системой и запускать произвольные процессы
- ★ Открывать порты запущенных ресурсов для хост-машины
- ★ Управлять жизненным циклом запущенных ресурсов
- ★ Работать с несколькими репликами одного и того же ресурса

https://dev.to/asimmon/exploring-the-microsoft-developer-control-plane-at-the-heart-of-the-new-net-aspire-123

Orchestrate NPM Apps

Project(.csproj)	Node.js project (package.json)
Solution	Monorepo
NuGet	NPM (Node Package Manager)
NuGet Package	NPM Package
C# (created by Anders Hejlsberg)	TypeScript (created by Anders Hejlsberg)

Orchestrate NPM Apps

Demo

- JavaScript in Aspire
- Console logs in Dashboard by default
- Conditional Resources (by execution context)

Выводы

Выводы

Плюсы:

- Упрощённая оркестрация
- Набор абстракций
- Общая точка конфигурации
- Компоненты
- Dashboard
- Отличная документация и примеры
- Оркеструет .NET 6 ⩾
- Тестируемый (xUnit, NUnit, MSTest)

Выводы

Плюсы:

- Упрощённая оркестрация
- Набор абстракций
- Общая точка конфигурации
- Компоненты
- Dashboard
- Отличная документация и примеры
- Оркеструет .NET 6 ⩾
- Тестируемый (xUnit, NUnit, MSTest)

Минусы:

- Пока не востребован на рынке
- Непонятная поддержка компонентов

Выводы

Плюсы:

- Упрощённая оркестрация
- Набор абстракций
- Общая точка конфигурации
- Компоненты
- Dashboard
- Отличная документация и примеры
- Оркеструет .NET 6 ⩾
- Тестируемый (xUnit, NUnit, MSTest)

Минусы:

- Пока не востребован на рынке
- Непонятная поддержка компонентов

Подводные камни:

Benefits for all

Benefits for all

📈 Standalone Dashboard

Benefits for all

📈 Standalone Dashboard

🔭 Best Practices in Integrations

Полезные ссылки

Core

https://medium.com/@josephsims1/aspire-aspi8-deploy-microservices-effortlessly-with-cli-no-docker-or-yaml-needed-f30b58443107

https://itnext.io/understanding-net-aspire-orchestration-460b50b7da46

https://dev.to/asimmon/exploring-the-microsoft-developer-control-plane-at-the-heart-of-the-new-net-aspire-123

https://devblogs.microsoft.com/dotnet/adding-dotnet-aspire-to-your-existing-dotnet-apps/

https://habr.com/ru/articles/818907/

https://habr.com/ru/articles/820435/

Aspir8

https://prom3theu5.github.io/aspirational-manifests/getting-started.html

https://github.com/prom3theu5/aspirational-manifests

Other

https://dev.to/asimmon/net-aspire-dashboard-is-the-best-tool-to-visualize-your-opentelemetry-data-during-local-development-9dl

https://www.youtube.com/watch?v=ALR9NnlmL_c

https://dev.to/asimmon/must-have-resources-for-new-net-aspire-developers-3jc2

https://www.youtube.com/watch?v=89tWVaG00TM

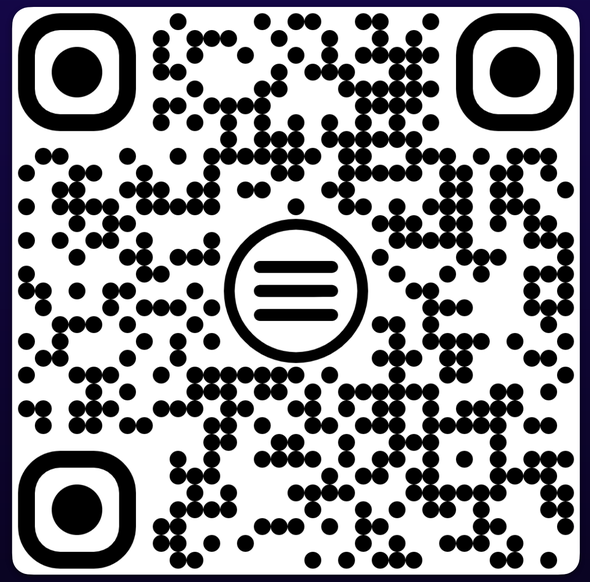
Examples

https://github.com/dotnet/aspire-samples/tree/main

https://github.com/bradygaster/AugmentR/tree/01-start

https://github.com/Alexanderbtw/AspireStarterStand

Чуть менее полезные ссылки



Alexander Goldebaev

@bornToWhine

sgoldebaev@gmail.com

Alexanderbtw