Unlock the power of .NET in the cloud: Journey into the future with .NET Aspire

Orestis Meikopoulos

Head of Engineering @ Code Create

https://linkedin.com/in/ormikopo

Agenda

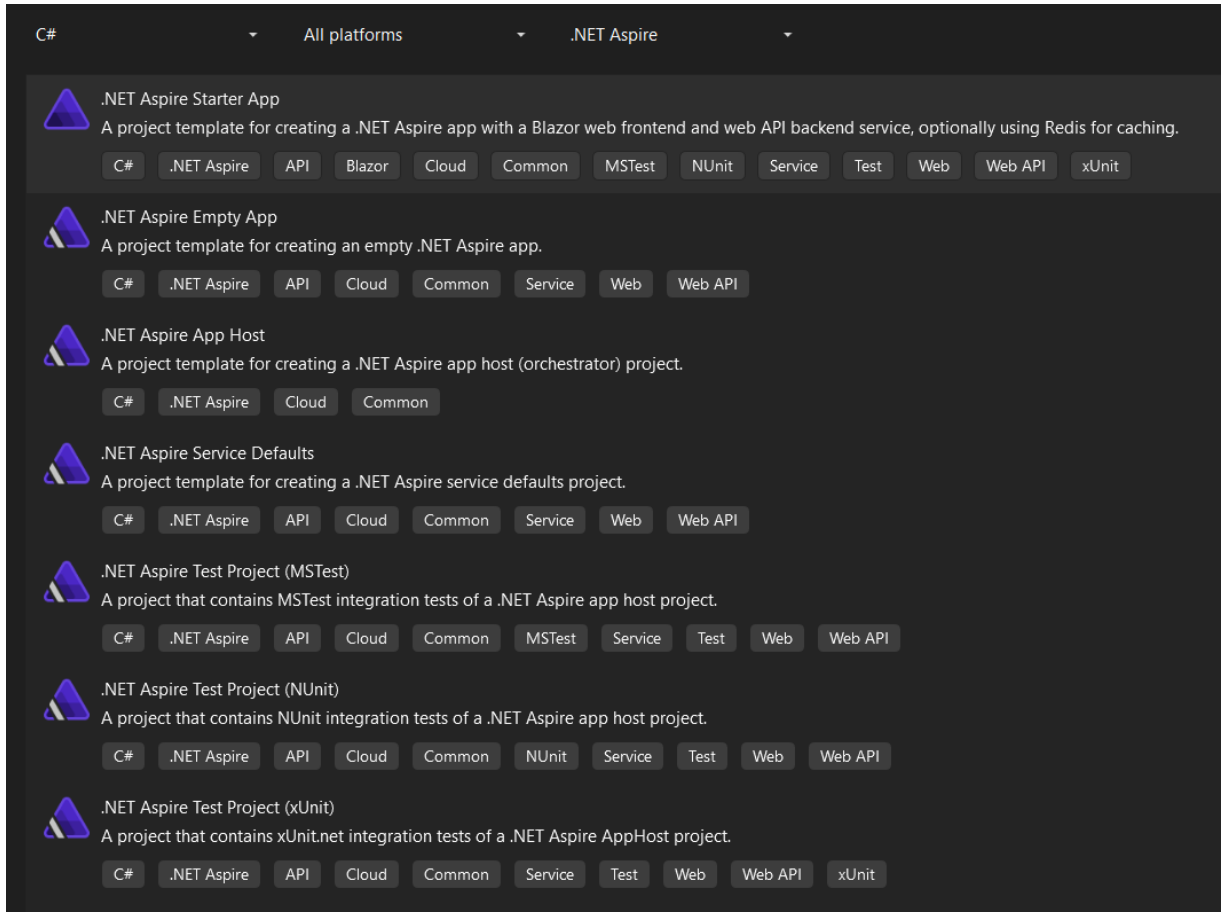
- Introduction, setup prerequisites and installation steps
- Key concepts
- Inner-loop networking
- Service discovery
- Service defaults
- Deploy to Azure

Introduction, setup prerequisites and installation steps

- **Purpose**: Build observable, production-ready, distributed applications.
- **Delivery**: Via a collection of NuGet packages.
- **Focus**: Cloud-native, distributed applications using microservices architecture.
- **Target**: Enhances building and managing .NET cloud-native apps.

Introduction, setup prerequisites and installation steps

- **Local requirements:**
  - .NET 8.0 or .NET 9.0
  - Docker Desktop or Podman for container support
  - IDE or code editor (e.g., Visual Studio 2022 version 17.9 or higher, Visual Studio Code, or JetBrains Rider with .NET Aspire plugin).
  - Alternatively, you can develop .NET Aspire solutions using GitHub Codespaces or Dev Containers.
- **Install the .NET Aspire templates**:
  - `dotnet new install Aspire.ProjectTemplates`



Introduction, setup prerequisites and installation steps

- **Creating projects**:
  - List templates: `dotnet new list aspire`
  - Create a basic project: `dotnet new aspire`
  - Create a project with UI and API: `dotnet new aspire-starter`

```
PS C:\Users\Orestis Meikopoulos\source\repos\CodeCreate.Template> dotnet new list aspire
These templates matched your input: 'aspire'

Template Name                     Short Name              Language  Tags
--------------------------------  ----------------------  --------  ------------------------------------------------------------------------------
.NET Aspire App Host              aspire-apphost          [C#]      Common/.NET Aspire/Cloud
.NET Aspire Empty App             aspire                  [C#]      Common/.NET Aspire/Cloud/Web/Web API/API/Service
.NET Aspire Service Defaults      aspire-servicedefaults  [C#]      Common/.NET Aspire/Cloud/Web/Web API/API/Service
.NET Aspire Starter App           aspire-starter          [C#]      Common/.NET Aspire/Blazor/Web/Web API/API/Service/Cloud/Test/MSTest/NUnit/xUnit
.NET Aspire Test Project (MSTest) aspire-mstest           [C#]      Common/.NET Aspire/Cloud/Web/Web API/API/Service/Test/MSTest
.NET Aspire Test Project (NUnit)  aspire-nunit            [C#]      Common/.NET Aspire/Cloud/Web/Web API/API/Service/Test/NUnit
.NET Aspire Test Project (xUnit)  aspire-xunit            [C#]      Common/.NET Aspire/Cloud/Web/Web API/API/Service/Test/xUnit
```

Key concepts - Terminology

- **App model**: A collection of interconnected resources making up your distributed application.
- **App host / Orchestrator project**: Orchestrates the app model, typically named with the *.AppHost suffix.
- **Resource**: Elements like projects, containers, executables, or services (e.g., databases, caches).
- **Integration**:
  - A NuGet package for the app host that models a resource.
  - A package that configures a client for use in a consuming app.
- **Reference**: Defines connections between resources as dependencies.

Key concepts - Defining the app model

- **Purpose**: Outline the resources in your app and their relationships.
- **Implementation**: Utilize `IDistributedApplicationBuilder` to configure resources and dependencies.
- **Example**: Use `AddProject` or `AddContainer` to include resources in your app model.

C#

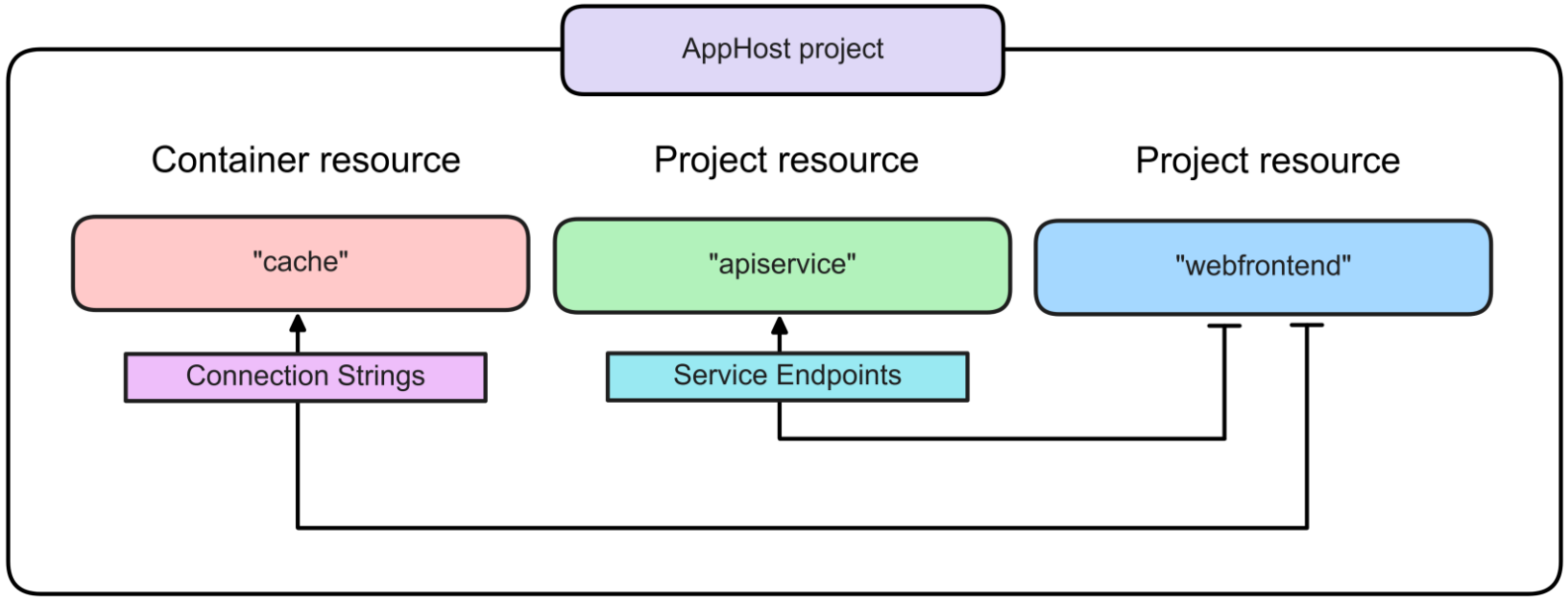
```
// Create a new app model builder
var builder = DistributedApplication.CreateBuilder(args);

// TODO:
//   Add resources to the app model
//   Express dependencies between resources

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

Key concepts – App host project

- **Purpose**: Handles running all the projects that are part of the .NET Aspire application.
- **Example**: The current image describes an application with two projects and a Redis cache.

XML

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

    <Sdk Name="Aspire.AppHost.Sdk" Version="9.1.0" />

    <PropertyGroup>
        <OutputType>Exe</OutputType>
        <TargetFramework>net9.0</TargetFramework>
        <IsAspireHost>true</IsAspireHost>
        <!-- Omitted for brevity -->
    </PropertyGroup>
```



Key concepts – App host project

C#

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

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

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

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

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

Key concepts – Resource types

- **Resource Management**: .NET Aspire apps are made up of a set of resources:
  - `AddProject`: A .NET project, for example an ASP.NET Core web app. Project resources are .NET projects that are part of the app model.
  - `AddContainer`: A container image, such as a Docker image.
  - `AddExecutable`: An executable file, such as a Node.js app.
- **Example**: To add a project to the app model:
  - `var aspireDemoApp = builder.AddProject<Projects.GlobalAzure_NetAspire_Server>("aspiredemoapp")`

```
C#

var builder = DistributedApplication.CreateBuilder(args);

// Adds the project "apiservice" of type "Projects.AspireApp_ApiService".
var apiservice = builder.AddProject<Projects.AspireApp_ApiService>("apiservice")
                        .WithReplicas(3);
```

Key concepts – Reference management

- **Define dependencies**:
  - Use **WithReference** to establish dependencies among resources:
    - `ConnectionStrings__ aspiredemosqlserver="localhost:1433“`

```
var customerDb = builder
    .AddSqlServer("aspiredemosqlserver");

builder
    .AddProject<Projects.GlobalAzure_NetAspire_Server>("aspiredemoapp")
    .WithReference(customerDb);
```

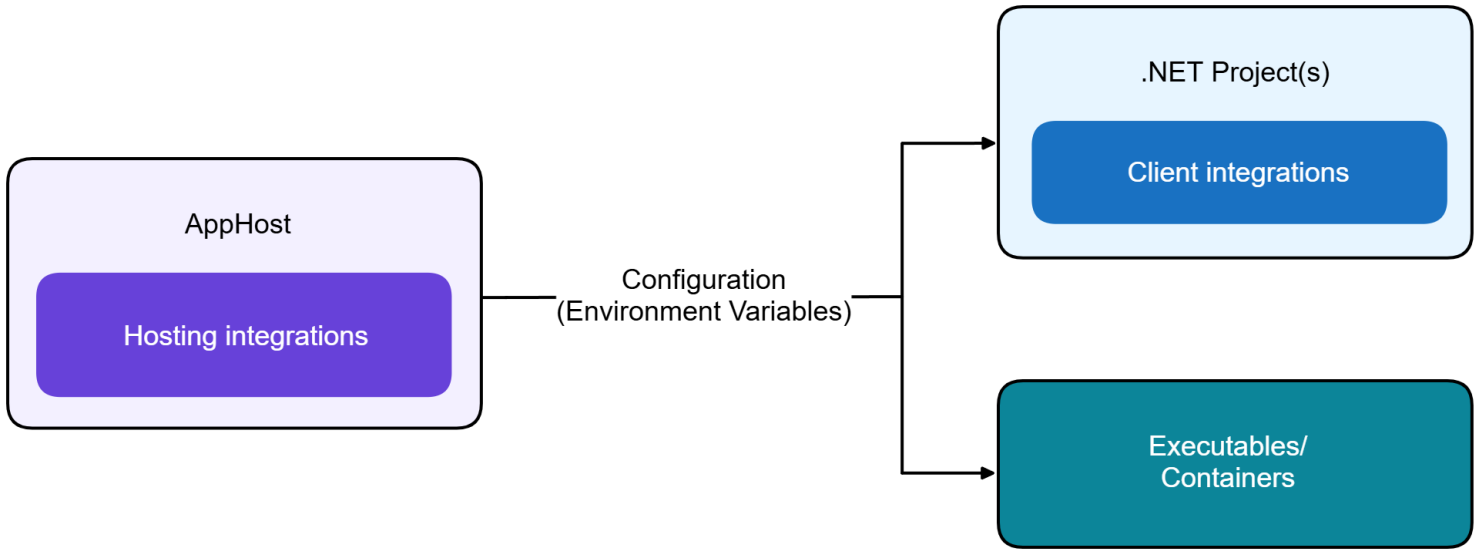
- **Connection strings and service discovery**:
  - Inject environment variables for dependencies and service discovery:

services__aspiredemoapi__http__0	http://localhost:5036
services__aspiredemoapi__https__0	https://localhost:7143

```
var aspireDemoApi = builder
    .AddProject<Projects.GlobalAzure_NetAspire_Api>("aspiredemoapi");

var aspireDemoApp = builder
    .AddProject<Projects.GlobalAzure_NetAspire_Server>("aspiredemoapp")
    .WithReference(aspireDemoApi)
    .WithExternalHttpEndpoints();
```

Key concepts – Integration

- **Purpose**: Enhance integration with services like Redis and PostgreSQL through curated NuGet packages.
- **Setup**: Automatically integrated features such as connection retries and timeouts to maintain functionality during failures.
- **Configuration**: Through JSON configuration files / directly via code.



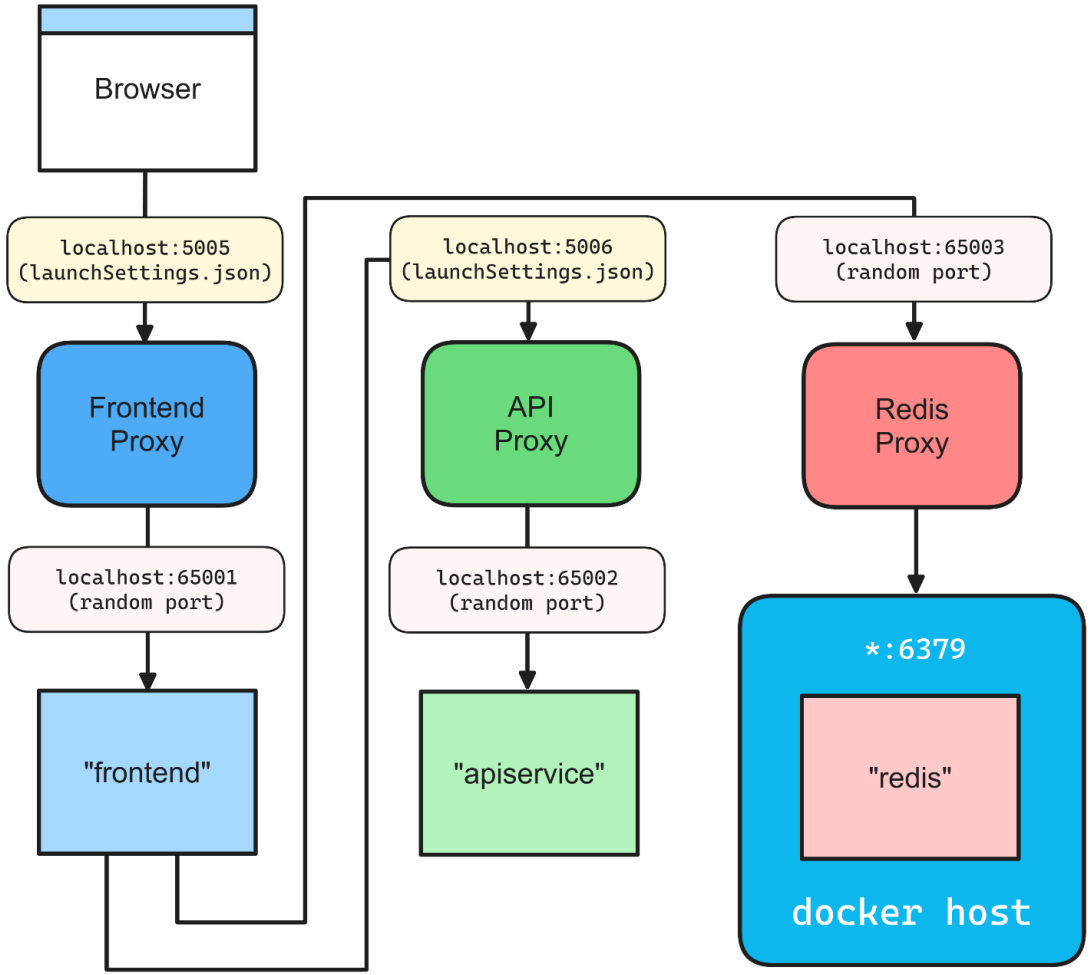

```
"Aspire": {
  "Npgsql": {
    "HealthChecks": false,
    "Tracing": false
  }
}
```

Audience Question

Is .NET Aspire reminding you of some other technology?

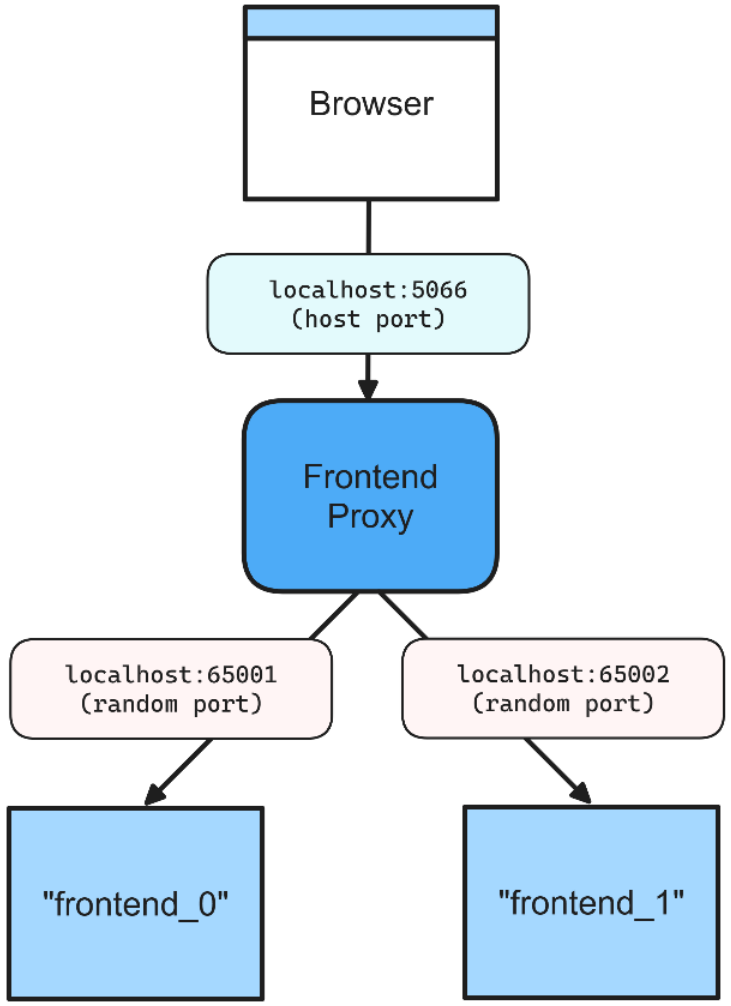
Inner-loop networking – Service bindings

- **Role**: Connect your app to external services required (databases, queues, APIs).
- **Types**:
  - Implicit: Automatically created from launch profiles.
  - Explicit: Manually created using WithEndpoint.
- **Proxy function**: Handles routing and load balancing, launched for each service binding.



Inner-loop networking – Service bindings

- **Configuration**: Host port is assigned to a proxy, which manages connections to services.
- **Example**: Using `AddProject` with `WithHttpEndpoint` and `WithReplicas`:
  - Creates multiple service replicas, each listening on a unique port.
  - Proxies route traffic to appropriate service replica based on the configuration.



Service discovery

- **Purpose**: Facilitate configuration of service discovery for development and testing environments.
- **Functionality**: Allows apps within the .NET Aspire framework to automatically discover and connect with each other.
- **Implementation**: Service discovery settings are provided to individual services within the application model based on their references.
- **Example**:

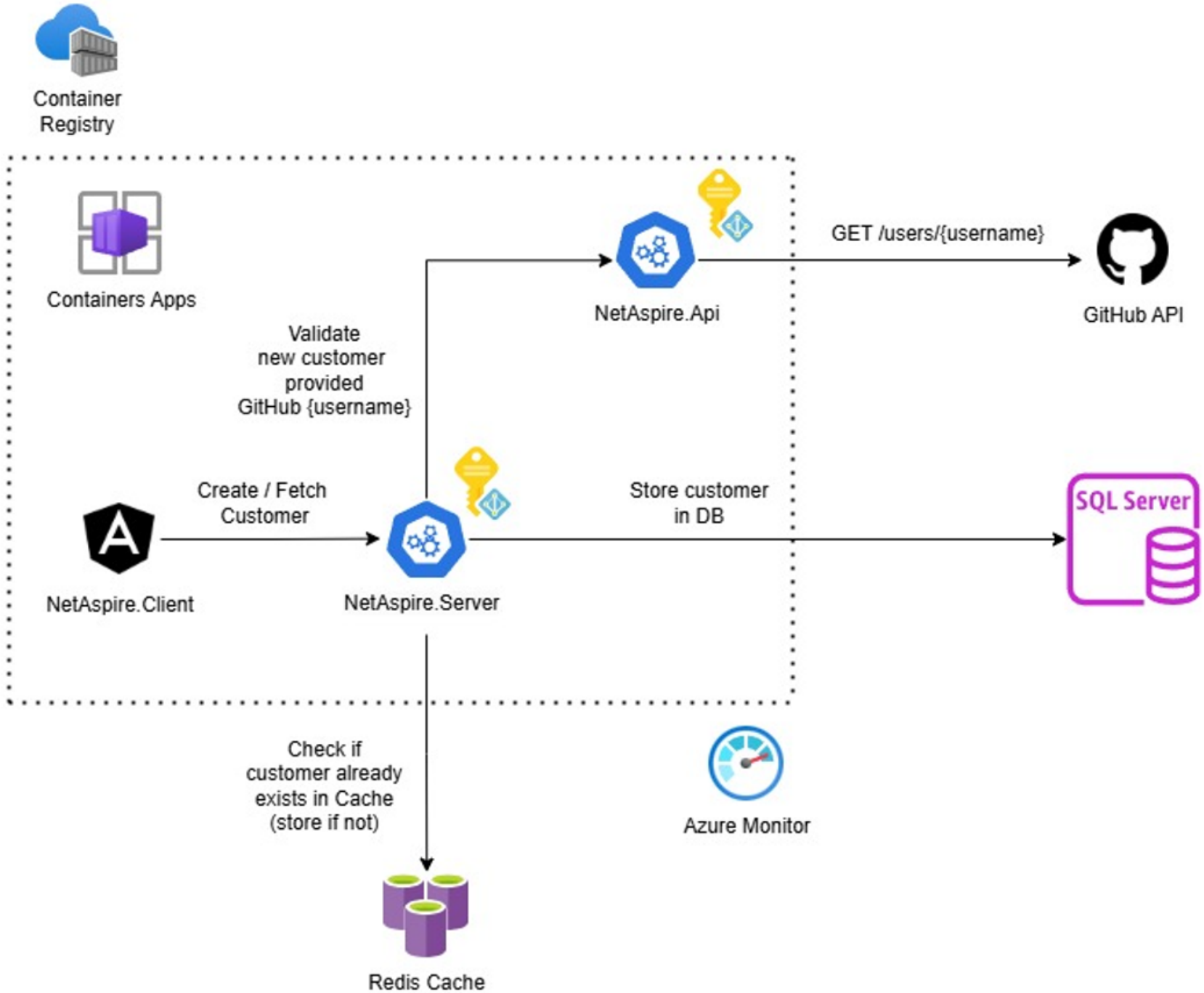
```
`var builder = DistributedApplication.CreateBuilder(args);
var catalog = builder.AddProject<Projects.CatalogService>("catalog");
var basket = builder.AddProject<Projects.BasketService>("basket");
var frontend =
builder.AddProject<Projects.MyFrontend>("frontend").WithReference(basket).WithReference(catalog)`
```

Service defaults

- **Purpose**: Manage extensive configurations for cloud-native applications across various environments.
- **Key methods:**
  - `ConfigureOpenTelemetry`: Sets up metrics and tracing.
  - `AddDefaultHealthChecks`: Incorporates default health check endpoints.
  - `AddServiceDiscovery`: Adds service discovery functionality.
  - `ConfigureHttpClientDefaults`: Sets up HttpClient defaults

```
2 references | Orestis Meikopoulos, 2 days ago | 1 author, 2 changes
public static IHostApplicationBuilder AddServiceDefaults(this IHostApplicationBuilder builder)
{
    builder.ConfigureOpenTelemetry();

    builder.AddDefaultHealthChecks();

    builder.Services.AddServiceDiscovery();

    builder.Services.ConfigureHttpClientDefaults(http =>
    {
        // Turn on resilience by default
        http.AddStandardResilienceHandler();

        // Turn on service discovery by default
        http.AddServiceDiscovery();
    });

    return builder;
}
```

Demo


Container Registry
Containers Apps
NetAspire.Api
GET /users/{username}
GitHub API
Validate new customer provided GitHub {username}
Create / Fetch Customer
NetAspire.Client
NetAspire.Server
Store customer in DB
SQL Server
Check if customer already exists in Cache (store if not)
Azure Monitor
Redis Cache

Thank you

- For the opportunity
- For participating
- For listening