.NET Aspire introduction

The Problem

- Building distributed apps can be difficult.
- This has been a truth in computer science since the very beginning.
- Today, we talk about using containers, Kubernetes, and creating microservices. But the nature of distributed computing is still very much the same as it was in the beginning.
- Creating applications that are distributed immediately require some basic requirements, including how the different services can reach each other, how to manage configuration across service boundaries, and how to monitor projects across the service boundaries.
- The community is trying to solve this with something called cloud native.

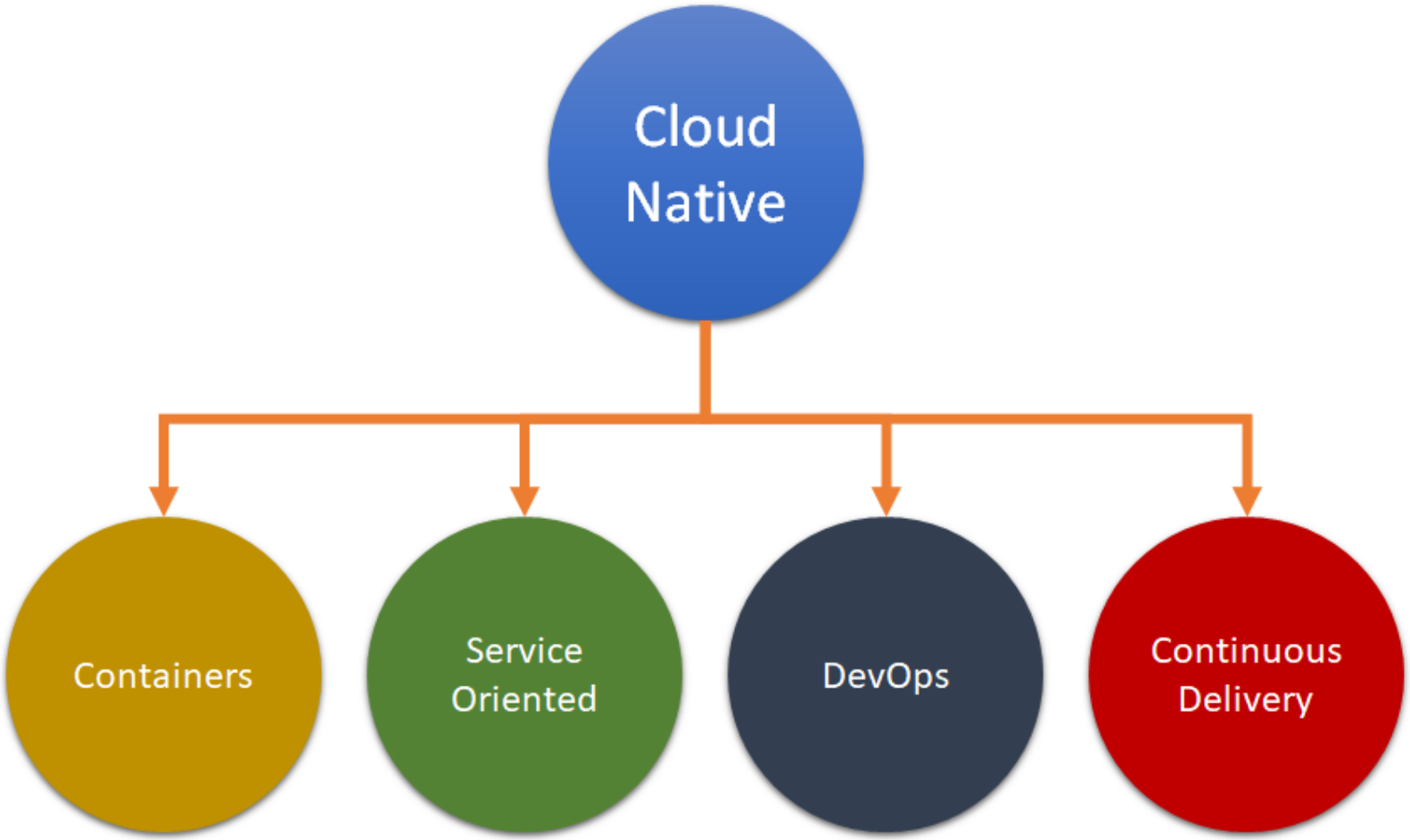
Cloud Native? What's That?

- The idea of cloud native comes from the basic desire to decouple services.
- Cloud native is an approach to run scalable applications in a variety of environments including public, private, and hybrid clouds. To achieve this, the architectural approach encourages:

    - Resiliency
    - Manageability
    - Observability

https://www.codemag.com/Article/2403071/Aspirational-.NET-What-Is-.NET-Aspire

Cloud native - four principles



Common services

What is .NET Aspire I

- .NET Aspire is a new stack that streamlines development of .NET **cloud-native services**

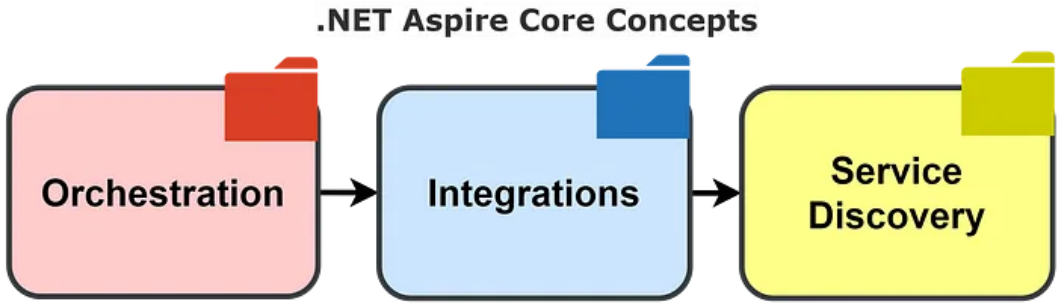
What is .NET Aspire II

- .NET Aspire is a set of tools, templates, and packages for building observable, production ready apps.

- Today's apps generally consume a large number of services, such as databases, messaging, and caching, many of which are supported via .NET Aspire Integrations.

What is .NET Aspire III

- .NET Aspire improves the experience of building apps that have a variety of projects and resources.
  - Dev-time orchestration
    - In .NET Aspire, "orchestration" primarily focuses on enhancing the local development experience by simplifying the management of your app's configuration and interconnections.
  - .NET Aspire integrations
    - .NET Aspire integrations are NuGet packages designed to simplify connections to popular services and platforms, such as Redis or PostgreSQL.
  - Project templates and tooling
    - OpenTelemetry
    - Default health checks
    - Service discovery



Orchestration in .NET Aspire I

- Orchestration in .NET Aspire is all about automating how your microservices are started, scaled, and managed.
- If you’ve ever struggled with manually writing Dockerfiles or hooking up multiple services in a docker-compose file, this is where .NET Aspire shines.

Docker Compose

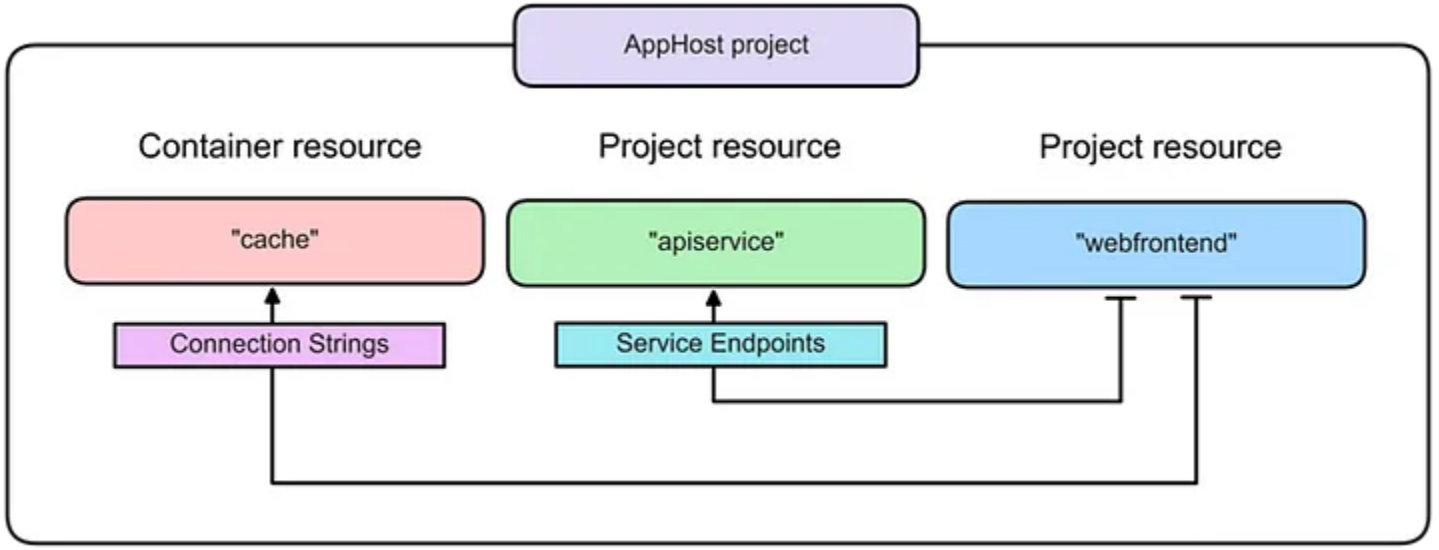
```yaml
services:
  contentplatform-api:
    image: ${DOCKER_REGISTRY-}contentplatform-api
    container_name: ContentPlatform.Api
    build:
      context: .
      dockerfile: ContentPlatform.Api/Dockerfile
    ports:
      - 5000:8080
      - 5001:8081

  contentplatform-reporting-api:
    image: ${DOCKER_REGISTRY-}contentplatform-reporting-api
    container_name: ContentPlatform.Reporting.Api
    build:
      context: .
      dockerfile: ContentPlatform.Reporting.Api/Dockerfile
    ports:
      - 6000:8080
      - 6001:8081

  contentplatform-presentation:
    image: contentplatform-ui:latest
    container_name: ContentPlatform.Presentation
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
    ports:
```

.NET Aspire Orchestration

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

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

var rabbitMq = builder.AddRabbitMQ("contentplatform-mq")
    .WithManagementPlugin();

builder.AddProject<Projects.ContentPlatform_Api>("contentplatform-api")
    .WithReference(postgres)
    .WithReference(rabbitMq);

builder.AddProject<Projects.ContentPlatform_Reporting_Api>("contentplatform-reporting-api")
    .WithReference(postgres)
    .WithReference(rabbitMq);

builder.AddProject<Projects.ContentPlatform_Presentation>("contentplatform-presentation");

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

Service Discovery in .NET Aspire I

- Service Discovery solves the question of “Where is my service?” in a world where microservices can spin up, scale out, or shut down at a moment’s notice. It ensures each microservice can dynamically locate and communicate with the others.



Service Discovery in .NET Aspire II

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

builder.Services.AddHttpClient<BasketServiceDashboardClient>(
    client => client.BaseAddress = new("https+http://_dashboard.basket"));
```


DEMO