Minimal APIs, Docker and Azure Container Apps with .NET Aspire

Christian Nagel, Sebastian Szvetecz

https://www.cninnovation.com

Christian Nagel

- Veteran developer
- Book author
- Microsoft MVP

- Training
- Coaching
- Development
- https://www.cninnovation.com

Wer bin ich?

- Sebastian Szvetecz
- Trainer, Developer
- sebastian@cninnovation.com
- Österreich
- CN innovation

.NET Aspire is early…

- Currently planned milestones (for .NET 8)

- Preview 3 – released Feb 13, 2024
- Preview 4 – March 5, 2024
- Preview 5 – March 28, 2024
- Preview 6 – TBD
- Preview TBD – TBD
- Stable release Q2 2024 - Microsoft Build??

A tour around .NET, Azure and .NET Aspire

- Backend with ASP.NET Core and EF Core
- .NET Aspire
- Docker
- Azure Container Apps
- Telemetry, App Insights

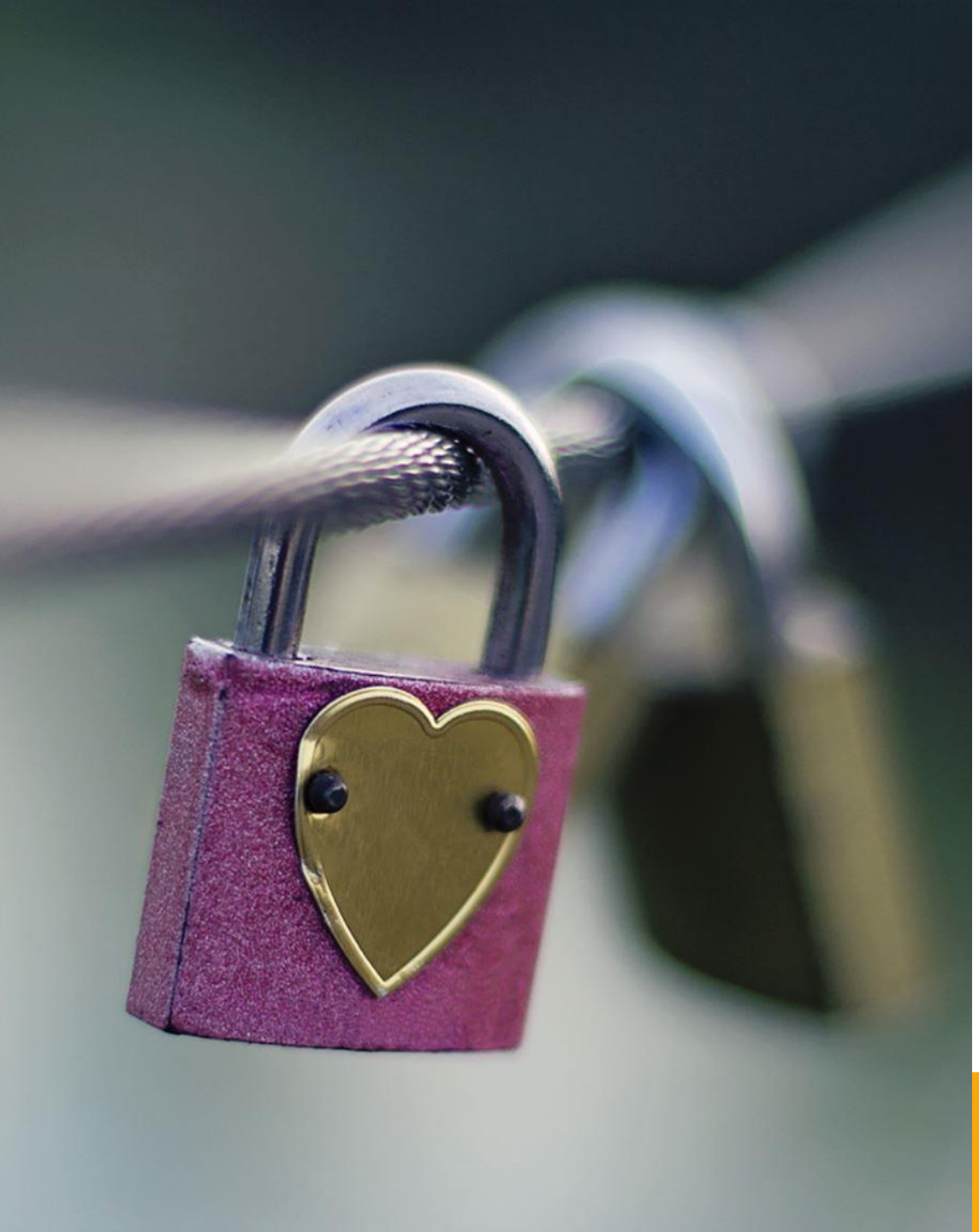
- Let’s play!

Questions for you!

https://menti.com

Questions during the day…

- If you have questions, don't hesitate to ask!
- If it‘s covered later → Whiteboard
- Sebastian monitors online questions!



Hands-on

- Try not to miss a thing during presentation
- Hands-on at your own pace
- Shortcuts are available!
- Focus on the parts important for you!

What do you need?

GitHub Account

(only required to create GitHub actions)

Visual Studio 2022 Preview with .NET Aspire workload

(not strictly required)

.NET 8 with .NET Aspire workload

Docker Desktop

Microsoft Azure subscription

(only required to create Azure resources)

Azure Developer CLI (azd)

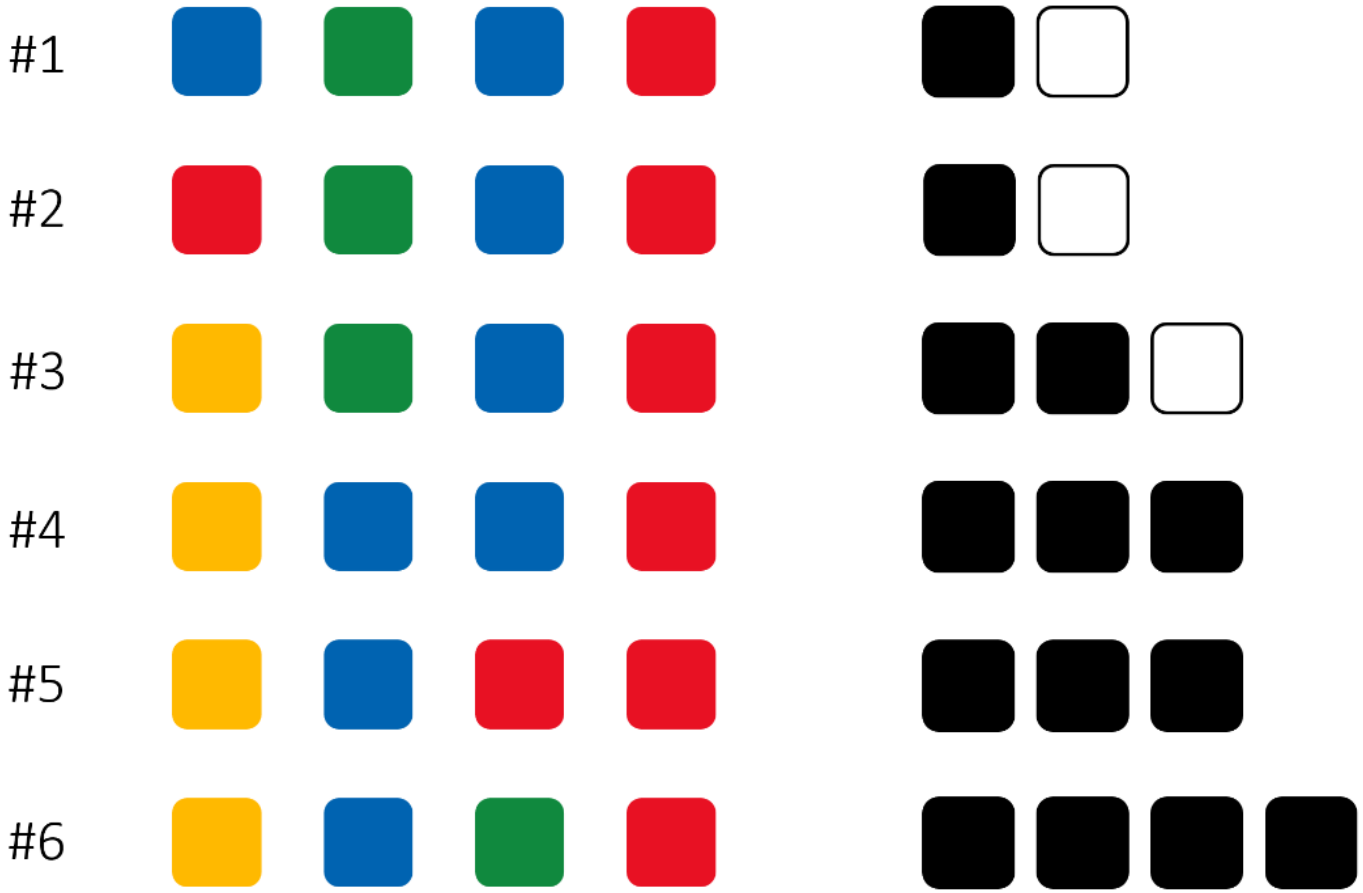
What did I install?

- Visual Studio 2022 Preview 17.10.0 Preview 1
  - winget install Microsoft.VisualStudio.2022.Community.Preview
- Docker Desktop
  - winget install Docker.DockerDesktop
- .NET Aspire Workload
  - dotnet workload install aspire --include-previews
- Azure CLI
  - winget install Microsoft.AzureCLI
- Azure Developer CLI
  - winget install Microsoft.azd
- Azure Cosmos DB Emulator
  - winget install Microsoft.Azure.CosmosEmulator

Master Mind!

- Rules

- Start a game
- Set a move



Part 1
Minimal APIs and Docker

Minimal API Project

- Top-level statements
- C# enhancements with lambda expressions

- dotnet new webapi -minimal --no-https -o GamesApi

Model for the API

- Game analyzer package
  - *CNinnovation.Codebreaker.Analyzers* Package
  - *IGame* interface
  - Game guess analyzers
- Models
  - *CNinnovation.Codebreaker.BackendModels* Package
  - *IGamesRepository*
  - *Game*, *Move*

In-Memory Repository

- Implement *IGamesRepository*
- In-memory state of games and their state

Games Service

- Functionality of the game invoked by the API
- Uses a **games factory** to work with different game types
- Uses a **game analyzer** library to calculate a move

Minimal API

- Less code to write compared to controllers
- Method parameters are injected
- OpenAPI (Swagger) API description
- .NET 8: Native AOT Compilation

Minimal API Enhancements with .NET 7 and 8

- Grouping
- Typed Results
- Filters
- Source generators

HTTP Files

- Visual Studio Endpoints Explorer
  - Check endpoints
  - Generate requests
- HTTP files
  - Interact with APIs

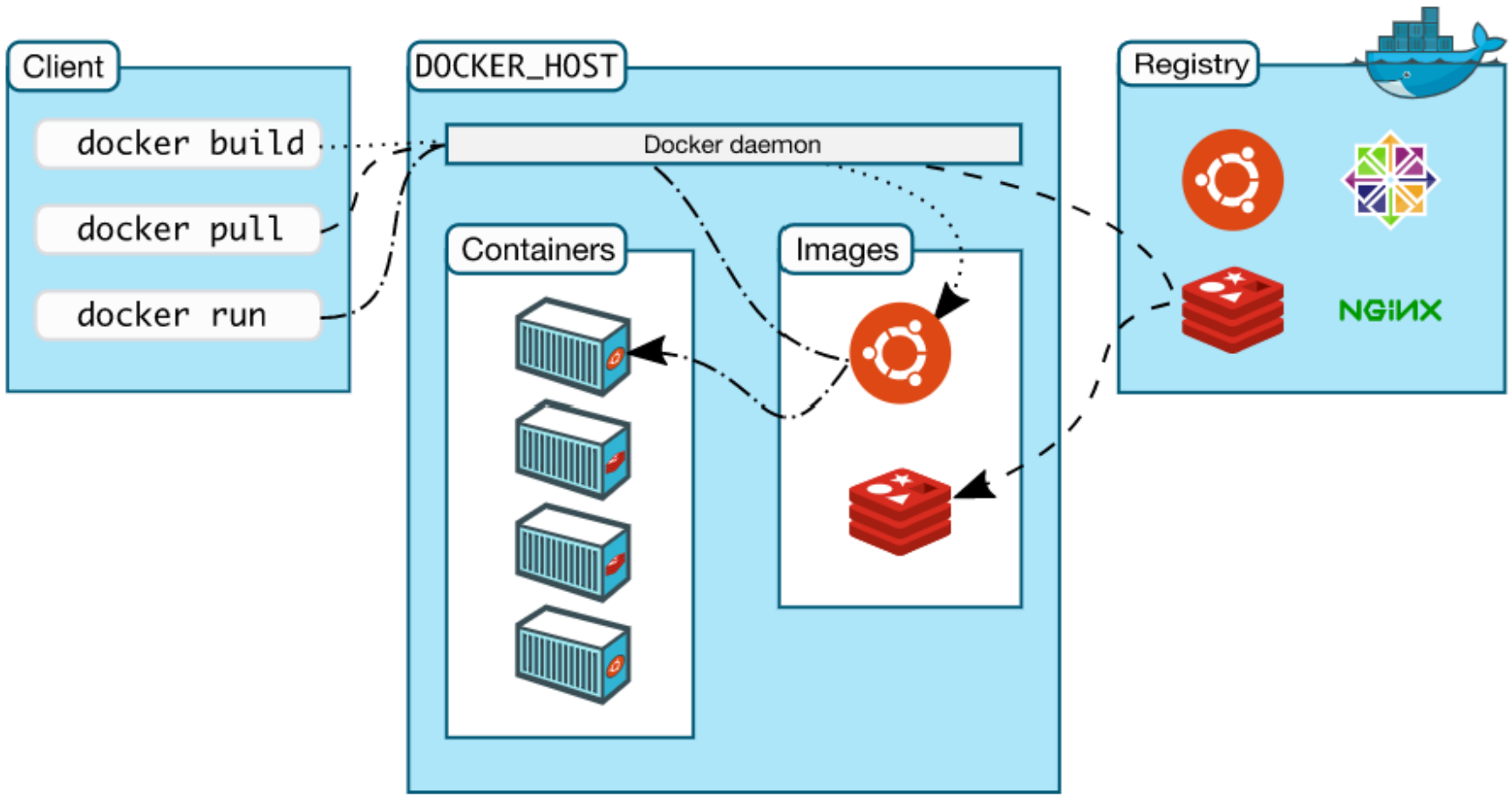
Docker

- Why? What are the issues?
- It runs on my machine!!
- Images – Containers – Registries - Docker CLI - Dockerfiles

Docker

- The Company
- The Technology
  - CLI
  - Images
  - Containers
  - Engine
  - Repository

Docker


Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

What is a Dockerfile?

A Dockerfile contains commands to build a Docker image

What is the Registry?

- Repository for docker images
- pull images for use
- push images that you've built

What is a Container?

- Encapsulated environment that runs applications
- Managed using the Docker API or CLI
- Docker containers running on a single machine *share the machines resources*

What is a Volume?

- Persist data with Docker containers
- Manage volumes with Docker CLI
- Share volumes across containers
- Access storage outside of the container with bind mounts

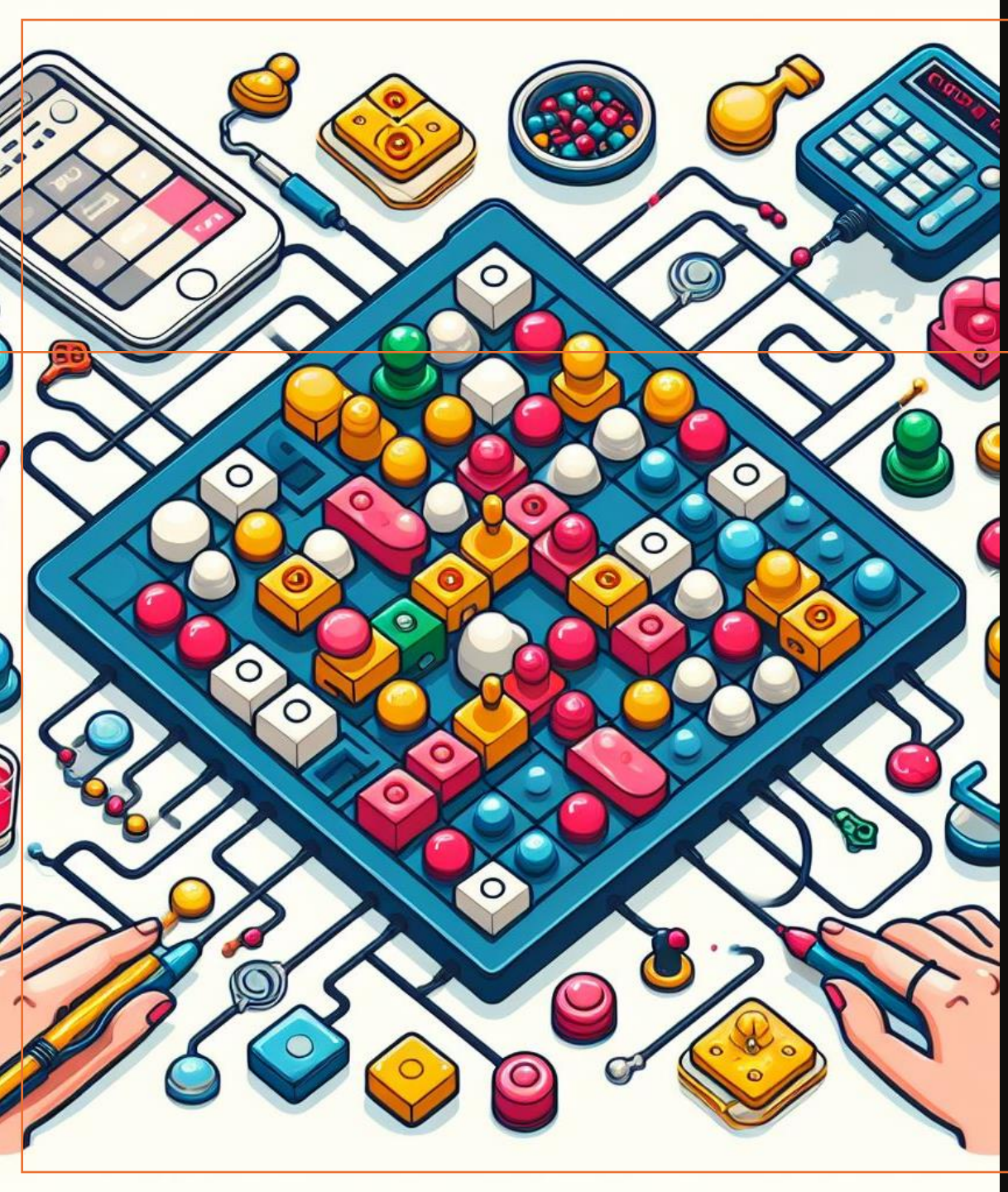
What is Orchestration?

- Tooling to help automate the maintenance of applications
- Replacement of failed containers
- Manage rollout of updates

- Manage, scale, maintain containerized applications

dotnet publish (.NET 7+)

- .NET CLI knows needed base images
- .NET CLI knows about build process
- *dotnet publish* allows creating images without specifying a Dockerfile
- Push images to a registry



What you've seen

- Creating a service with the minimal API
- Building Docker images

Hands-On – Create minimal APIs

- Part 1 – Minimal API

  https://github.com/CodebreakerApp/codebreakerlight/01-minimalAPI.md

.NET Aspire

What is .NET Aspire?

“.NET Aspire is an opinionated, cloud ready stack for building observable, production ready, distributed applications.”

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

.NET Aspire

Orchestration

Components

Tooling

Orchestration

- App composition
  - .NET projects
  - Containers
  - Executables
  - Cloud resources
- Service discovery and string management

.NET Aspire Components

- NuGet packages
- Simplify connections to popular services and platforms
- Component works with Aspire orchestration

Component

- Registers with the DI container
- Applies Azure ServiceBus configurations
- Enables health checks, logging, telemetry

```
builder.AddAzureServiceBus("servicebus");
```

Tooling

- .NET Workload
- Templates
  - aspire
  - aspire-starter
  - aspire-apphost
  - aspire-servicedefaults
- Visual Studio integration
- Create manifest for deployment

What you've seen

- .NET Aspire tooling and orchestration features

Hands-On – .NET Aspire

- Part 2 – .NET Aspire

  https://github.com/CodebreakerApp/codebreakerlight/02-aspire.md


Part 3 – EF Core, SQL Server & Cosmos with Aspire in Action

Pass configuration to services (Aspire)

- Read a configuration with the AppHost

```
string dataStore = builder.Configuration["DataStore"] ?? "InMemory";
```

- Set an environment with the service

```
var gameAPIs =
 builder.AddProject<Projects.Codebreaker_GameAPIs>("gameapis")
   .WithEnvironment("DataStore", dataStore)
```

- And use it

```
builder.Configuration["DataStore"]
```

EF Core

- Object-relational mapper (O/RM)
- Enables working with a database using .NET objects
- Eliminates (most) data-access code
- Many database engines (MSSQL, PostgreSQL, SQLite, Cosmos, …)

Using EF Core

- Define a model
- Context for Dependency Injection
- Use SQL Server
- Use the Azure Cosmos provider

SQL Server running with Docker

- Use a Docker image
- Define a volume for persistence
- Use *WithReference* from a project

```
var sqlServer = builder.AddSqlServerContainer("sql", sqlPassword)
  .WithVolumeMount(
    "volume.codebreaker.sql",
    "/var/opt/mssql",
    VolumeMountType.Named)
  .AddDatabase("CodebreakerSql");
```

SQL Server EF Core Component

- Package *Aspire.Microsoft.EntityFrameworkCore.SqlServer*

```
builder.AddSqlServerDbContext<YourDbContext>("sql");
```

Azure Cosmos DB

- Distributed NoSQL and relational database
- APIS
  - NoSQL (DocumentDb)
  - PostgreSQL
  - MongoDB
  - Cassandra
  - Table
  - Apache Gremlin

Configure Cosmos - AppHost

- Get a connection string

```
string cosmosConnectionString = builder.Configuration["CosmosConnectionString"]
 ?? throw new InvalidOperationException("Missing connection string");
```

- Add Cosmos Database

```
var cosmos = builder.AddAzureCosmosDB(
 "GamesCosmosConnection", cosmosConnectionString)
 .AddDatabase("codebreaker");
```

Azure Cosmos DB Emulator

- Emulator on Windows
- Docker Images on Linux/Windows/Mac

What you've seen...

- Azure Cosmos DB
- Using EF Core with the Cosmos Provider
- Using a Docker container with a volume
- Using Aspire components

Hands-On – Create minimal APIs

- Part 3 – Databases and Aspire Components

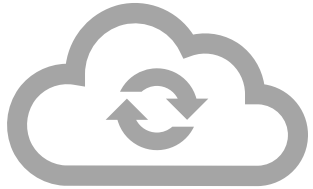
  https://github.com/CodebreakerApp/codebreakerlight/03-components.md

Part 4 - Hosting with Microsoft Azure

Azure Container Services

Azure Container Registry

Registry for your images



Azure Container Apps

Abstraction of Kubernetes

Simple Ingress configuration

Simple Scalability (KEDA)

Pay per use

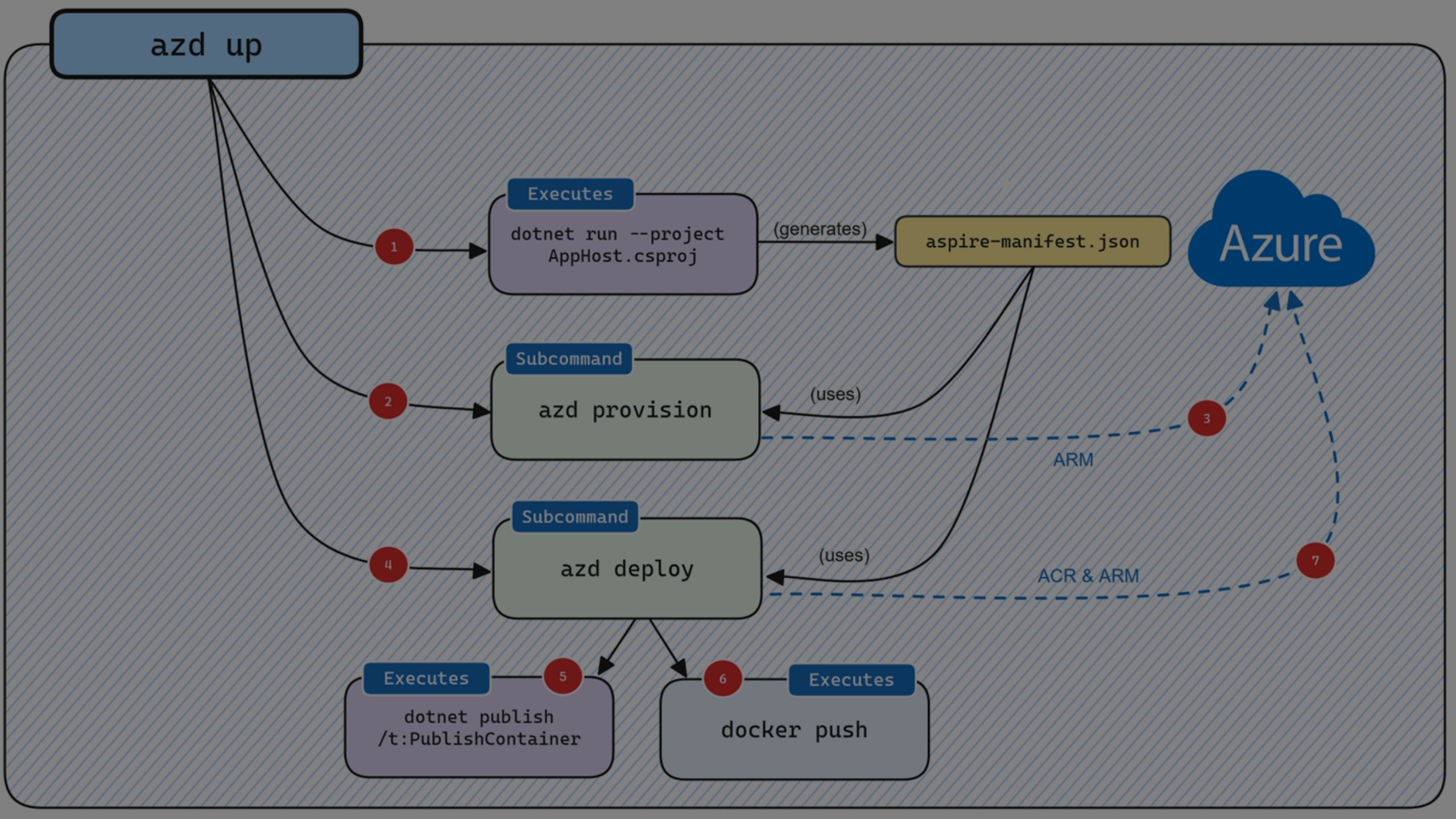
Bicep 💪

- DSL – Domain Specific Language
- Declarative syntax to deploy Azure resources
- All Azure resources supported
- Simple syntax
- Integration

Aspire & Azure Developer CLI

- Initialize the project
  - azd init
- Publish to Azure
  - azd up

- Create environments
  - azd env new codebreaker-dev


azd up
1
Executes
dotnet run --project AppHost.csproj
(generates)
aspire-manifest.json
Azure
2
Subcommand
azd provision
(uses)
3
ARM
4
Subcommand
azd deploy
(uses)
7
ACR & ARM
5
Executes
dotnet publish /t:PublishContainer
6
Executes
docker push

Generate Bicep

- azd config set alpha.infraSynth on
- azd infra synth

What you've seen

- Azure Cosmos DB
- Azure Container Registry
- Azure Container Apps
- Azure Developer CLI with Aspire

Hands-On

- Part 4 – Hosting

  https://github.com/CodebreakerApp/codebreakerlight/04-hosting.md


Next Steps...

Logging, Monitoring, Metrics, Distributed Tracing

- Logging
  - ILogger
  - Strongly typed logging
- Metrics
  - System.Diagnostics.Metrics
  - IMetricsFactory (.NET 8)
- Distributed Tracing
  - .NET Activity
- OpenTelemetry
  - Pre-configured with .NET Aspire

Configuration

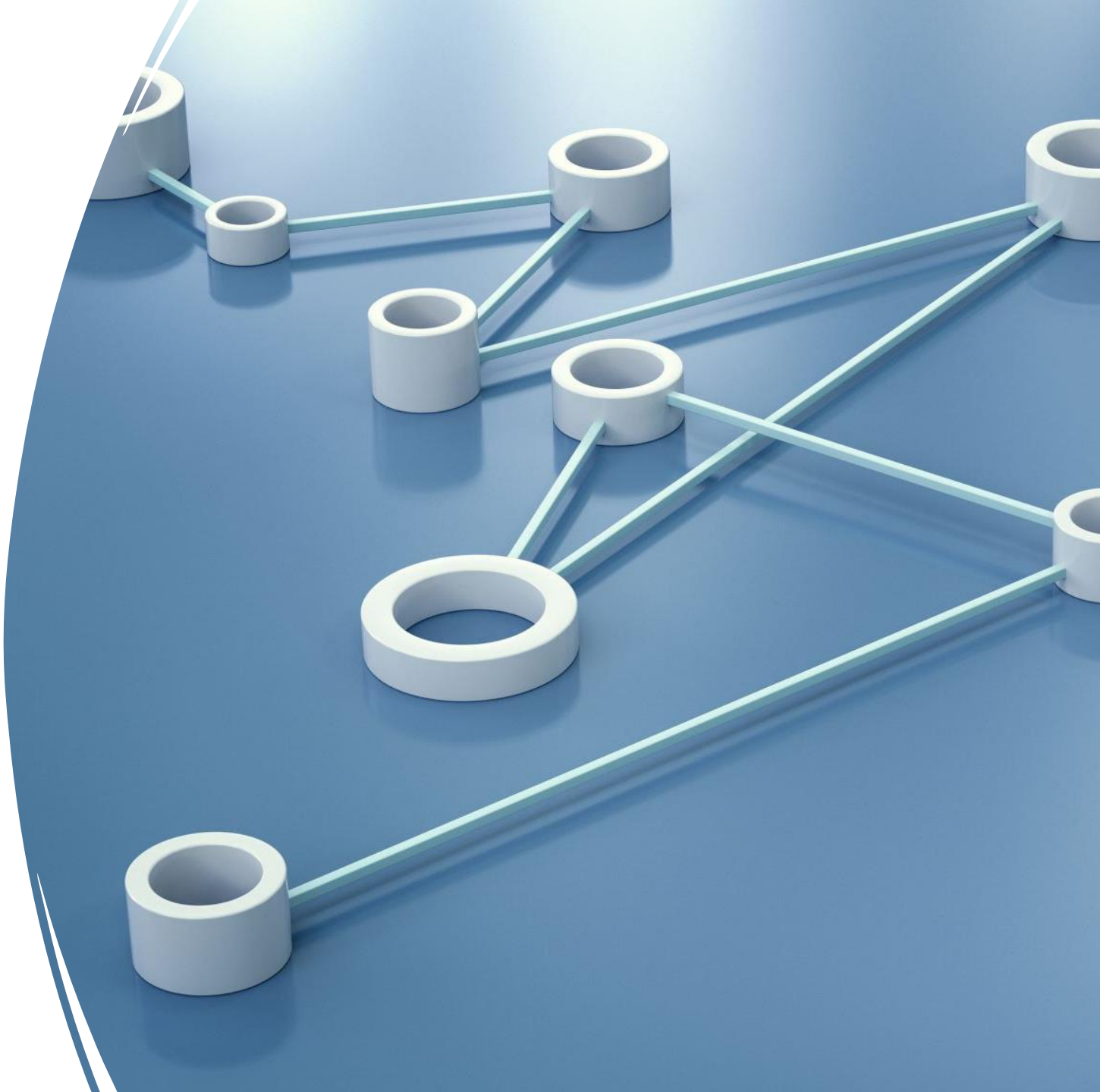
- Azure App Configuration
  - Configuration for multiple services
  - Can be used to differ staging/production environments
  - Supports feature flags
- Azure Key Vault
  - Secrets
  - Certificates

Managed Identities

- Reduce the need to store secrets
- Configure user-assigned or system-assigned managed identities
- Role access with Azure resources

Summary

- Application Insights
- Azure App Configuration
- Azure Key Vault
- Managed Identities



Take away

- ASP.NET Core minimal APIs
- EF Core with Azure Cosmos DB
- .NET Aspire
- Microsoft Azure Services
- Azure Container Apps

Thank you for joining!

Questions?

- https://github.com/cnilearn/bastaspring2024
- https://github.com/codebreakerapp/codebreakerlight
- https://csharp.christiannagel.com
- https://www.cninnovation.com