by entwickler.de

Migrate a .NET Backend to .NET Aspire

Christian Nagel

https://www.cninnovation.com

Christian Nagel

- Veteran developer
- Book author
- Microsoft MVP

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

Content

- What’s .NET Aspire?
- Checking the existing solution
- Migrate to .NET Aspire
- Deploy to Azure & ..
- Add more integrations

.NET Aspire is evolving fast!

- **.NET release cycles**
  - **Long term support - LTS – 3 years (8, 10)**
  - **Standard term support - STS – 2 years (7, 9, 11)**
- .

- **NET Aspire release cycles**
  - **Microsoft Modern Lifecycle**
  - **Only the latest version is supported**
  - **8.0 – Nov-2023**
  - **9.0 – Nov-2024**
  - **9.1 – Feb-2025**
  - **9.2 – Apr-2025**
  - **9.3 – Jun-2025**
  - **9.4 – Jul-2025**
  - **9.5 – yesterday 8 PM CEST**

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

1393 8931

Questions during the day…

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

Hands-on

- You don’t need to do the hands-on at the same time I’m showing it to you
- You have lab files available in the repo
- Dedicated time for hands-on on your own pace
- I’m available for questions
- And the weekend is here as well ;-)

What do you need?

GitHub Account

(only required to create GitHub actions)

Visual Studio 2022 or Visual Studio 2026

with .NET Aspire

.NET 9

Docker Desktop

Microsoft Azure subscription

(only required to create Azure resources)

Azure Developer CLI (azd)

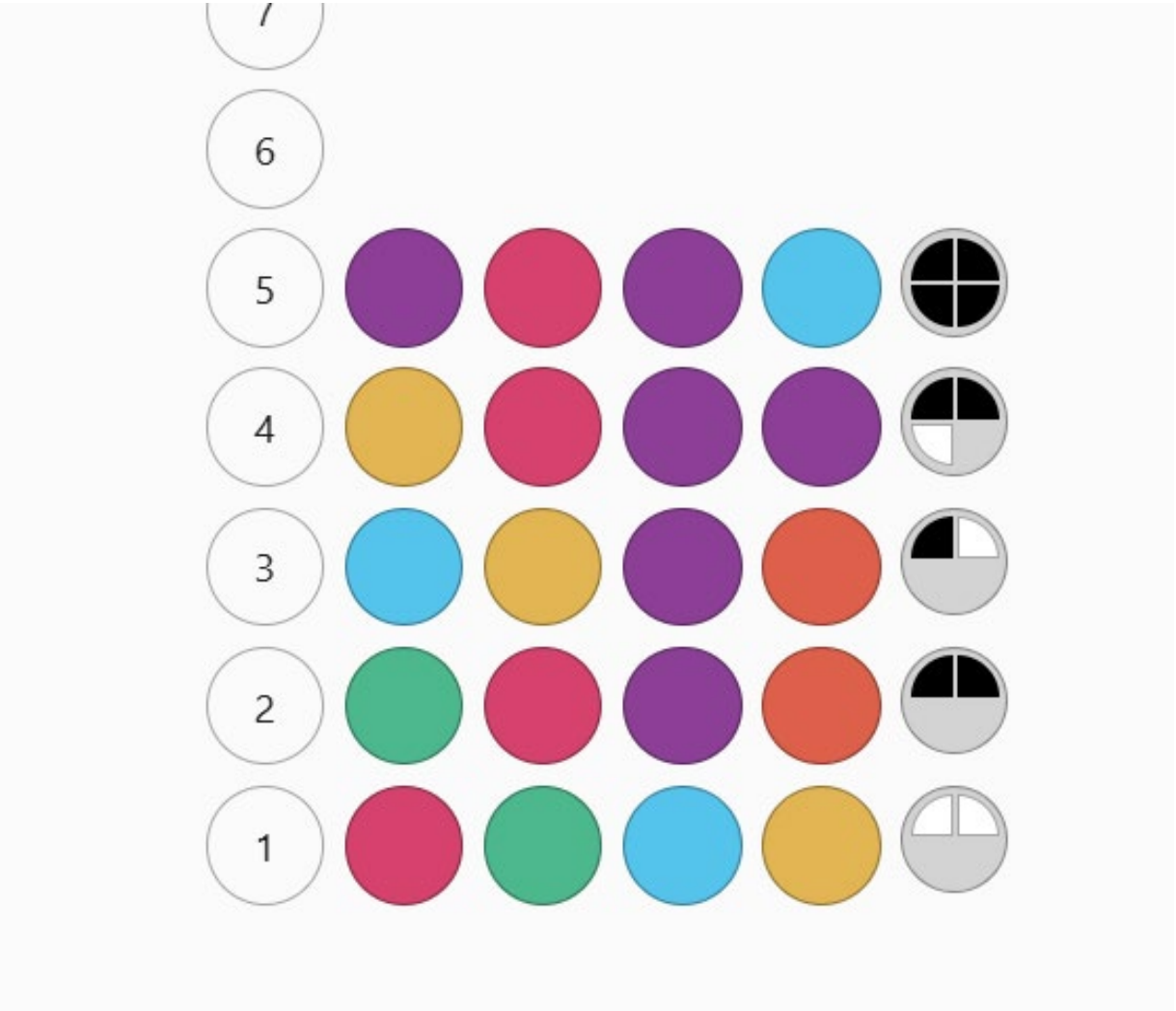
What did I install?

- Visual Studio 2026 Insider
  - https://visualstudio.microsoft.com/insiders/
- Docker Desktop
  - winget install Docker.DockerDesktop
- Azure CLI
  - winget install Microsoft.AzureCLI
- Azure Developer CLI
  - winget install Microsoft.azd

Codebreaker App
https://blazor.codebreaker.app

- Rules

- Start a game
- Set a move



.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

Integrations

Tooling

.NET Aspire CLI

- dotnet tool install aspire.cli –g

- new
- run
- add
- publish (preview)
- deploy (preview)

Orchestration

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

.NET Aspire Integrations

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

Integrations

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

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

Tooling

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

What you've seen

- .NET Aspire orchestration
- .NET Aspire tooling

Hands-On – .NET Aspire

- Part 1 – .NET Aspire

Migrate existing solution

Add Aspire Projects

- AppHost
  - Orchestration
- ServiceDefaults
  - OpenTelemetry
  - Service discovery

Orchestrator

- Defines all app components
- Managed dependencies, lifecyle, environment variables
- DistributedApplication builder API

Service Discovery

- Using logical names (https+http://gamesapi)
- Ideal for distributed systems
- Endpoint resolvers
  - Configuration-based
  - Pass-through
  - DNS
  - Custom

Summary

- Easy add for advantages during development!

Integrations

- Pre-built components for common infrastructure services
- Databases, caches, queues, cloud resources
- Service registration
- Environment variable injection
- Health checks
- Service discovery
- Telemetry

Adding Integrations
- Database

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 sqlGames = builder.AddSqlServer("gamesdatabase")
       .WithDataVolume("sqlgamesdata-volume")
       .WithLifetime(ContainerLifetime.Session)
       .AddDatabase("codebreaker-db");
```

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)
  .AddCosmosDatabase("codebreaker");

cosmos.AddContainer("GamesV3", "/PartitionKey“);
```

Azure Cosmos DB Emulator

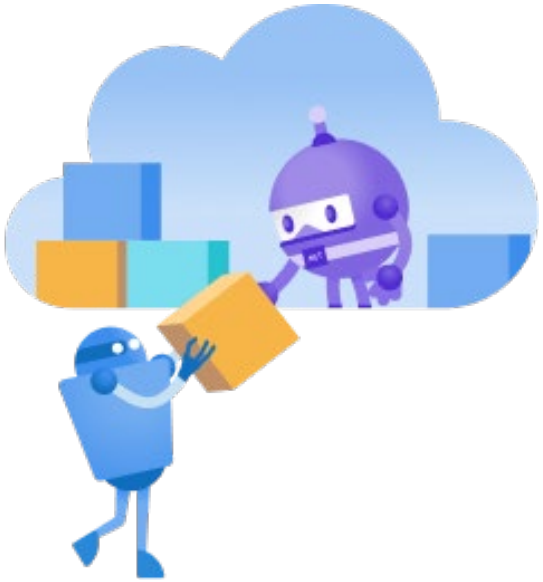
- Emulator on Windows
- Docker Images on Linux/Windows/Mac

```
var cosmosServer = builder.AddAzureCosmosDB("cosmos-db")
  .RunAsPreviewEmulator(p =>
    p.WithDataExplorer()
    .WithDataVolume("sqlcosmosdata-volume")
    .WithLifetime(ContainerLifetime.Session));
```

What you've seen...

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

Hands-On – Integrations



- Databases and Aspire Integrations

Azure resources during development

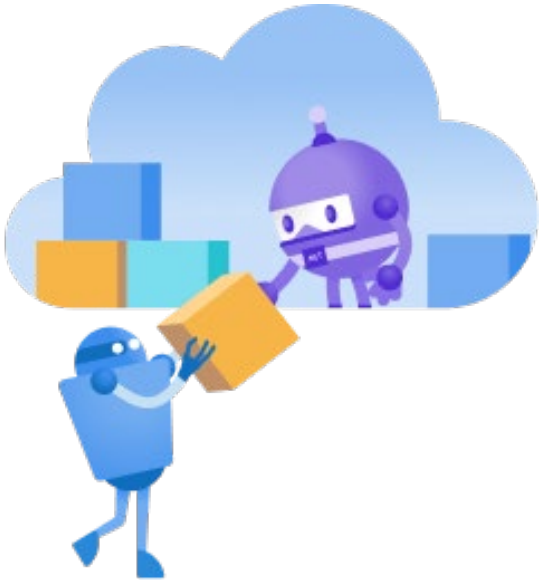
Connect the subscription

- Connected services configuration
- Parameters while starting the application

Azure resources

- Are automatically provisioned
- Configuration and resources stored with user secrets
- You are responsible to delete the resources when not needed!
- Configure Azure:CredentialSource (AzureCli) in case of authentication issues

Hands-On



- Using Azure during development time

Publish and deploy

App Host Manifest

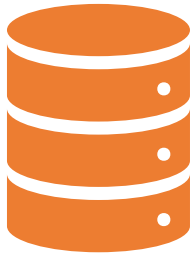
- JSON file to describe all resources with dependencies and configuration

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

Azure Developer CLI

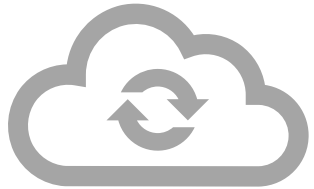
- azd init
- azd up
  - azd provision
  - azd deploy

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

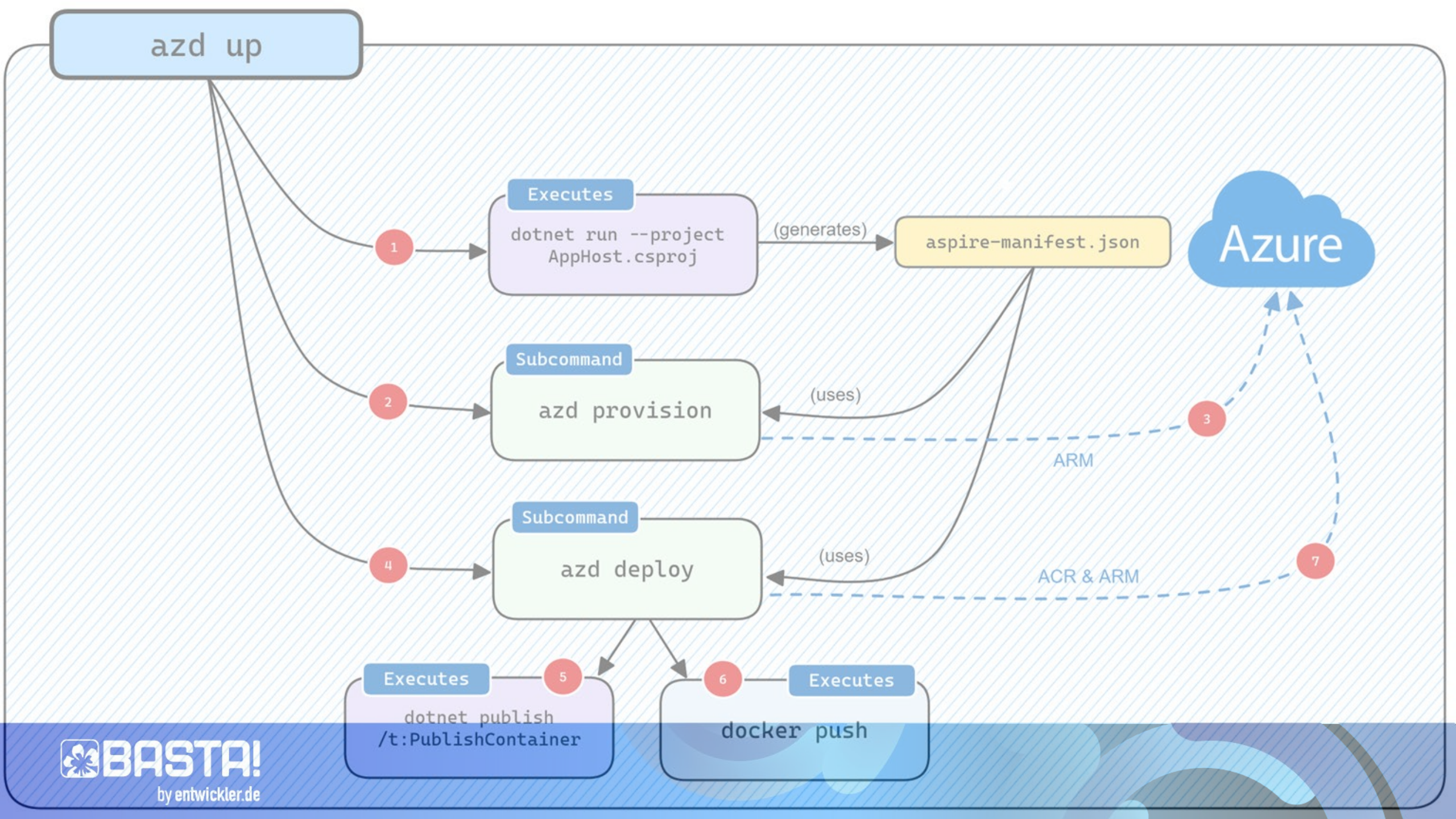
Bicep 💪

- 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 (azd provision | azd deploy)

- 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 Scripts

- **azd infra gen**

Next: configure infrastructure with C#!

- Aspire using Azure Cloud Development Kit

```
var storage = builder.AddAzureStorage("storage")
  .ConfigureInfrastructure(infrastructure ⇒
  {
    var account = infrastructure.GetProvisionableResources()
      .OfType<StorageAccount>().Single();
    account.Sku = new StorageSku()
    { Name = sku.AsProvisioningParameter(infrastructure) };
    account.Location =
      locationOverride.AsProvisioningParameter(infrastructure);
  });
```

Aspire CLI with host environments

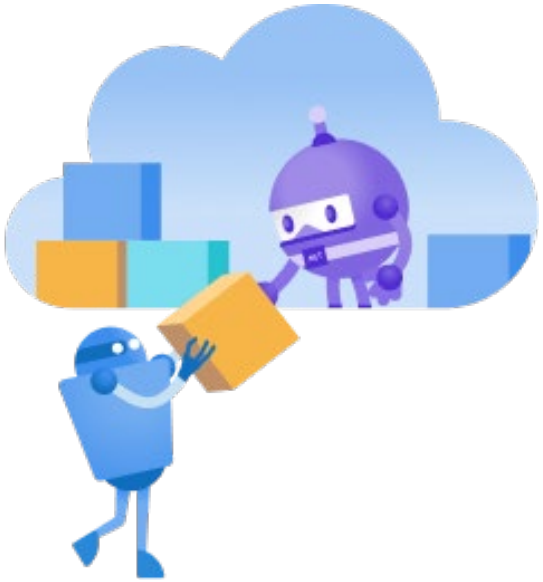
- aspire publish
- aspire deploy (coming!)

- AddDockerComposeEnvironment
- AddAzureAppServiceEnvironment
- AddAzureContainerAppEnvironment

What you've seen

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

Hands-On



- Deployment


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

Hosting elsewhere

- Kubernetes
- Docker Compose
- AWS
- IIS

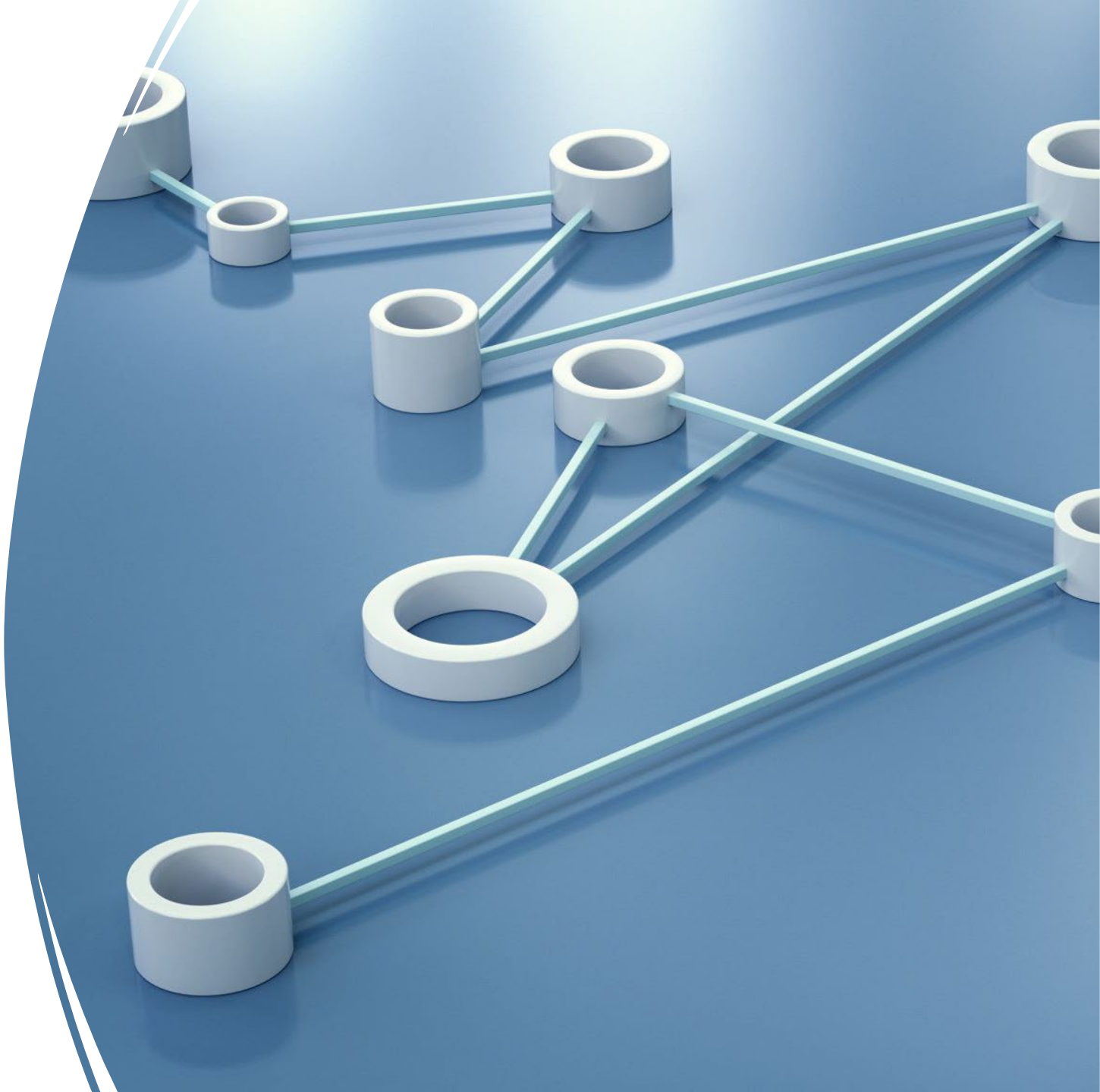
- https://github.com/aws/integrations-on-dotnet-aspire-for-aws
- https://github.com/prom3theu5/aspirational-manifests
- https://github.com/CommunityToolkit/Aspire

Other languages

- Java
- Python
- Golang
- Node
- Rust
- Bun

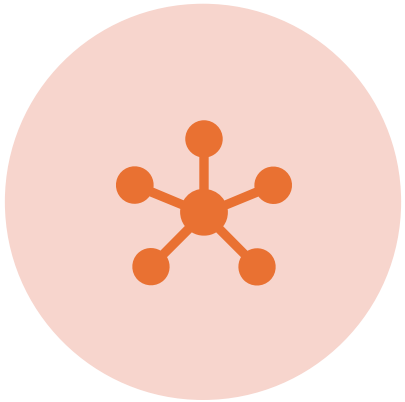
Summary

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



Take away - .NET Aspire

ORCHESTRATION



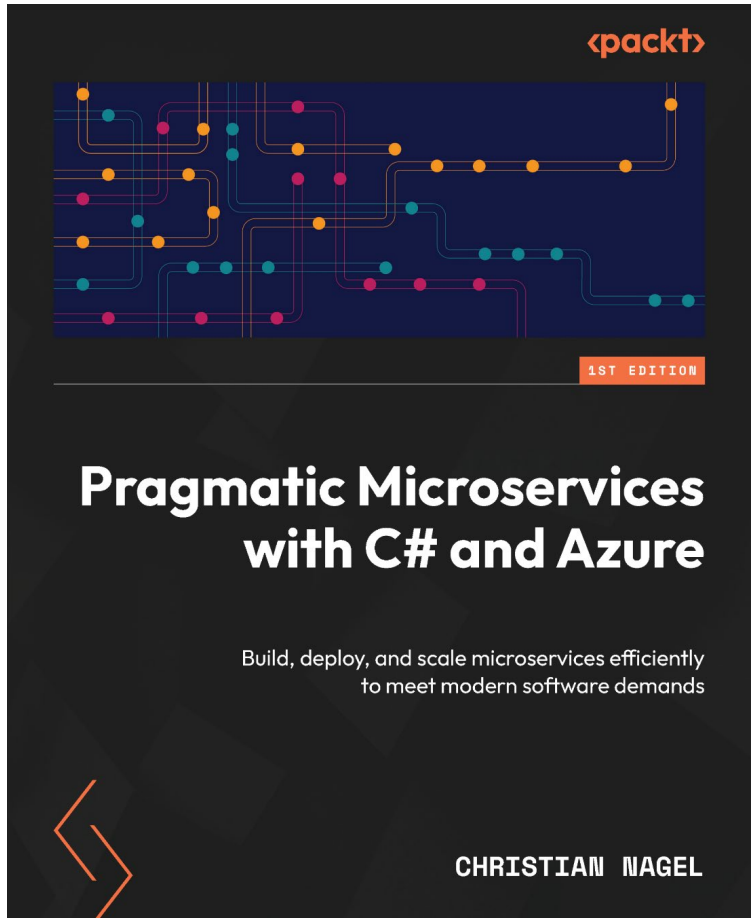
INTEGRATION

TOOLING

Thank you for joining!

Questions?

- https://blazor.codebreaker.app
- https://github.com/cnilearn/bastamainz2025
- https://github.com/codebreakerapp
- https://csharp.christiannagel.com
- https://www.cninnovation.com



Please fill out the BASTA! questionnaire!

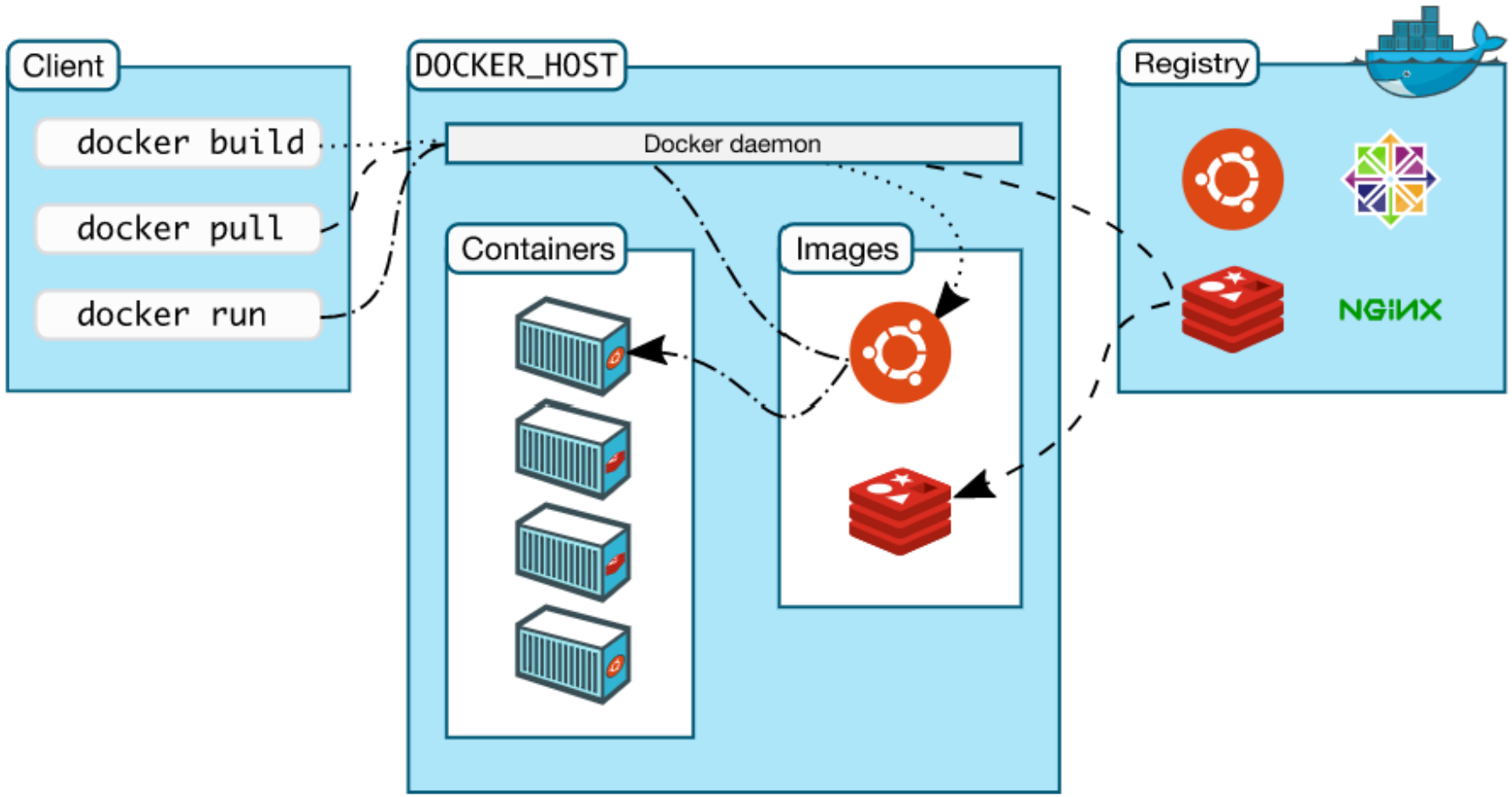
Docker

- 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

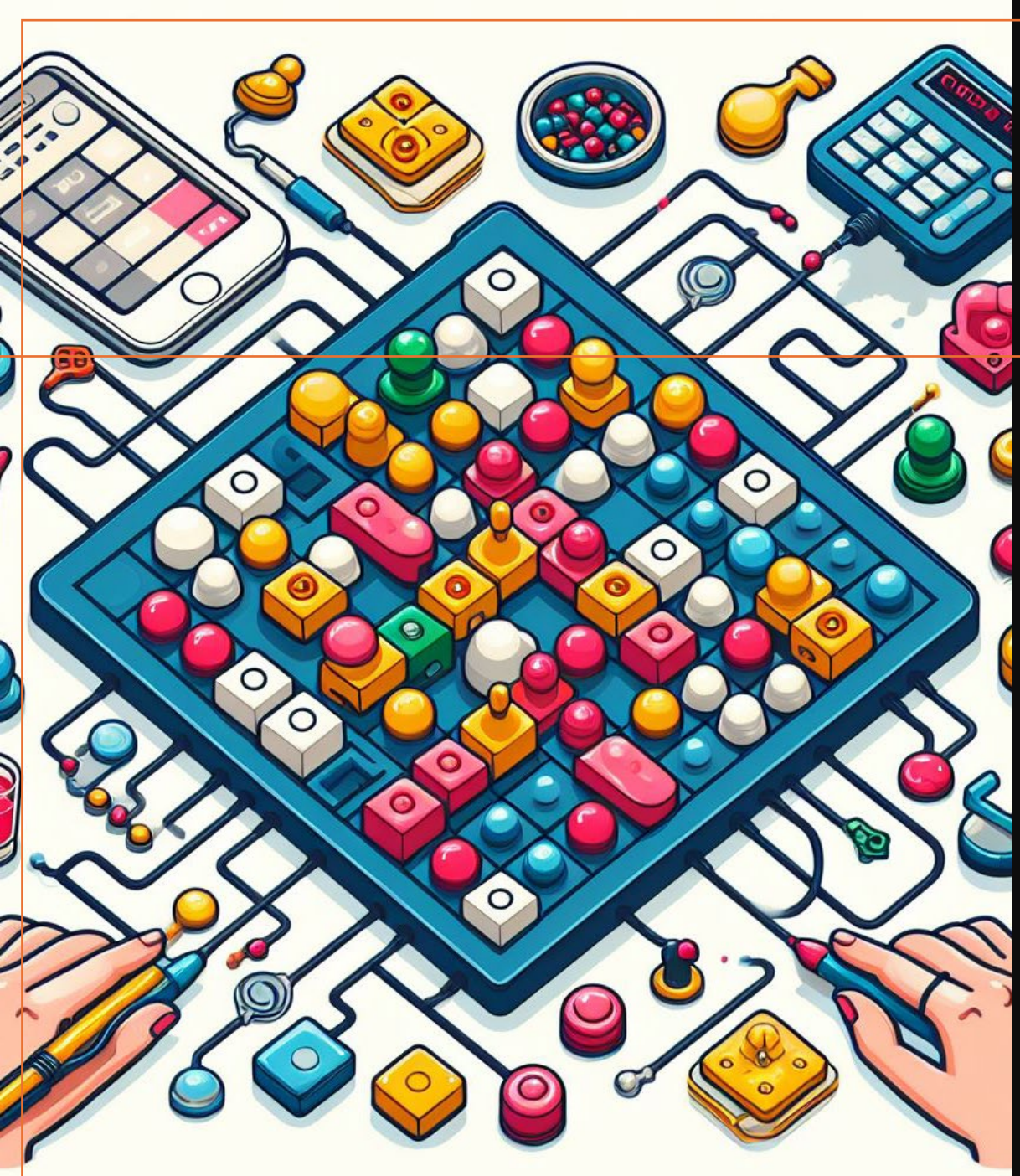
What is Orchestration?

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

- Manage, scale, maintain containerized applications

dotnet publish (since .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