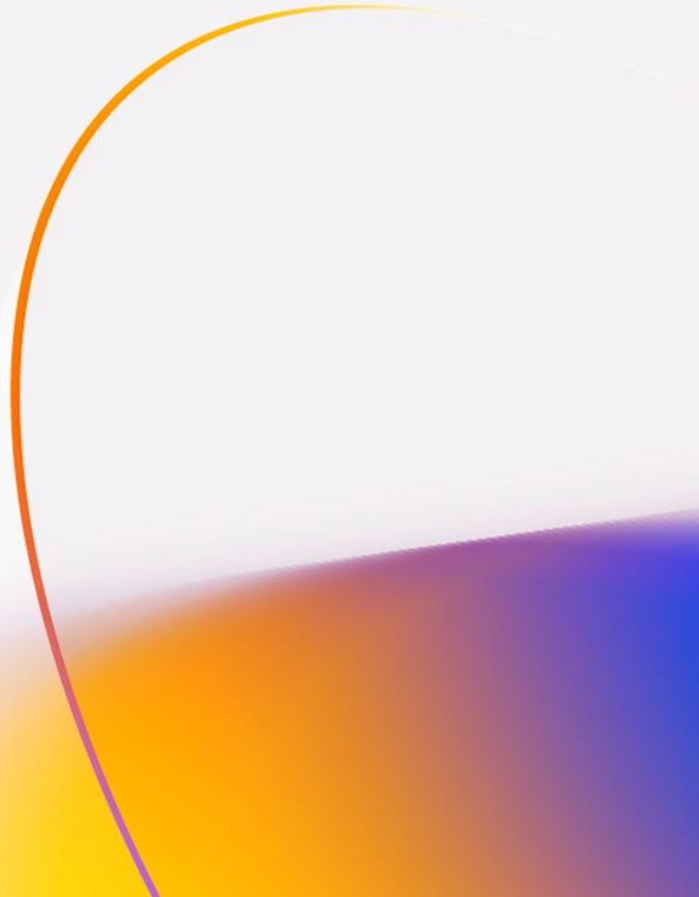
Microsoft

Azure Developer Day

Geovani de León
Carlos Lopez
Carlos Robles



Microsoft

Meet your speakers

Geovani

de León

Cloud Development Manager - PlusTI

me@yovadeleon.dev

Carlos

Lopez

Regional Data Architect - GBM

carlos.lopez@gbm.net

Carlos

Robles

Senior Product Manager - Microsoft

roblescarlos@microsoft.com

Agenda

- Introduction to Azure SQL
- Local Development with Azure SQL
- Cloud Development with Azure SQL
- Getting started with .NET Aspire

Azure SQL

Carlos Lopez

Data Architect

Microsoft MVP

caltls@gmail.com

linkedin.com/in/carlos-lopez-taks

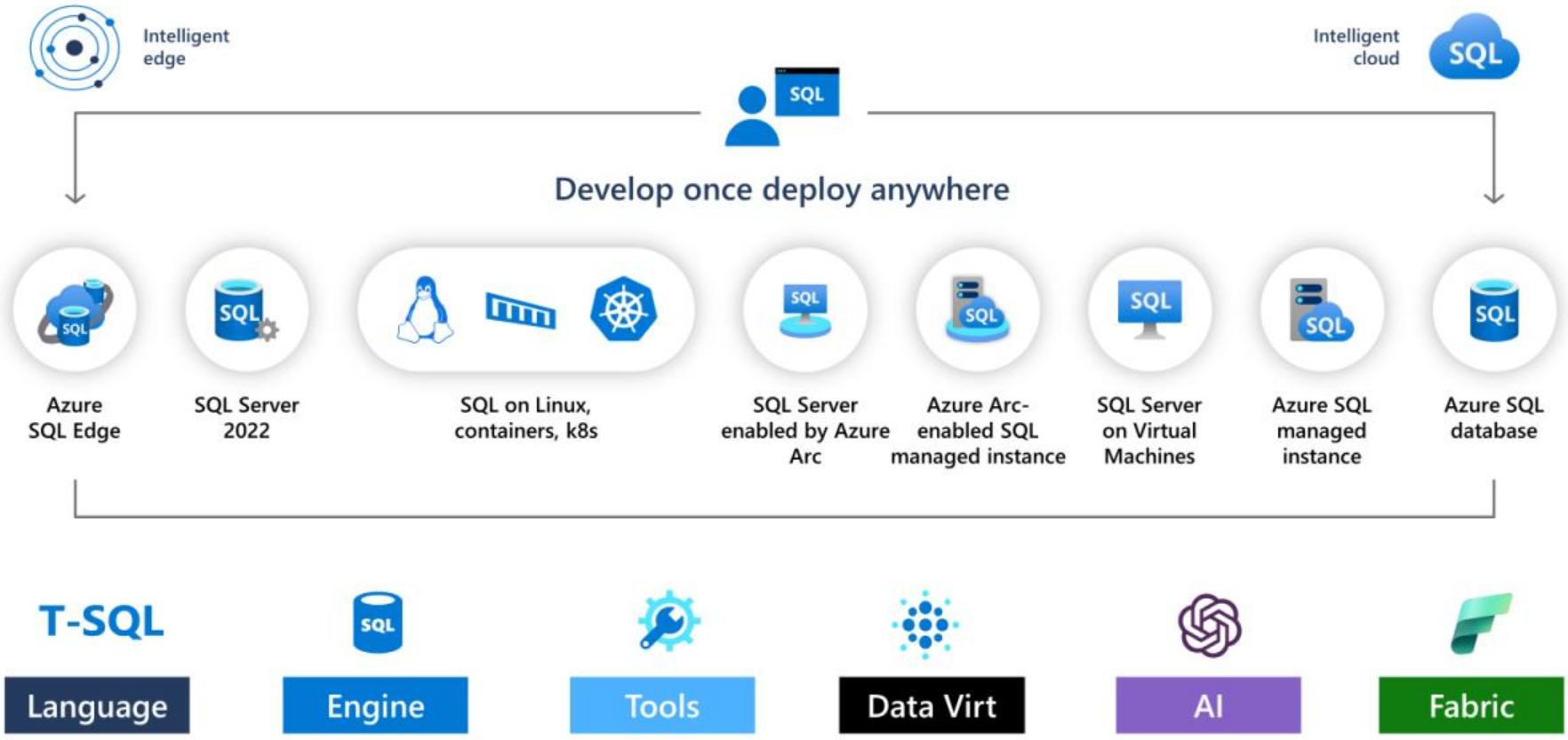
github.com/Muppity

Carlos López es Microsoft Data Platform MVP y Cloud Data Architect.

Con mas de 14 años de experiencia en manejo de datos a gran escala en diferentes plataformas.

Lider de comunidad tecnica de Azure Data Tech group, blogger, speaker.

Microsoft SQL – all edges and clouds



Azure SQL Database

- Start **free** and **pay as** you go and grow
- Develop **locally** and leverage **built-in** DevOps and CI/CD
- Develop on **your** OS, use **your favorite popular** language
- Performance you **need** with **built-in** security, scale, availability
- Build apps **globally** that **last** while you grow and integrate

Azure SQL DB

Azure SQL Database is the world's database

249K+

active customers every month

45T+

queries run every month

13M+

active databases every month

60+

regions globally supported

Beyond RDBMS

- JSON
- Spatial
- Graph
- Columnstore
- Memory-optimized
- Ledger
- Query Intelligence
- REST API integrated
- Data API Builder, GraphQL
- Azure Functions
- Hyperscale, Serverless, Pools
- Containers, DevOps, GitHub
- AI
- Fabric

Azure SQL Database free of charge

Get 100,000 vCore seconds of serverless compute and 32 GB of storage every month!

Azure SQL Database with serverless compute

Flexible compute automatically scales to meet demand.

No time limits

Apply this free offer for the life of your subscription.

Need more? No problem.

Stick with the default auto-pause option or continue usage for additional charges.

What's included:

One Azure SQL Database with serverless compute per Azure Subscription with 100,000 vCore seconds every month.

32 GB data storage +

32 GB backup storage.

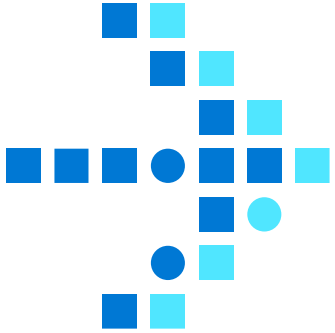
Learn More: aka.ms/sqlfreeoffer

Demo

Crear una Db Az SQL

Nuevos lanzamientos

- T-SQL support RegEx Azure SQL DB (Private preview)
- Copilot in Azure SQL Database is a new self-assistance AI companion (Private Preview)
- New Natural Language to SQL (NL2SQL) (Preview)
- Mirroring in Microsoft Fabric for Azure SQL Database (Public Preview)



Copilot en Azure SQL

Azure portal location	Experiences
Microsoft Copilot for Azure (preview)	**Integración de Azure Copilot:** Esta experiencia agrega conocimientos de Azure SQL a Microsoft Copilot para Azure, lo que proporciona a los clientes asistencia autoguiada, lo que les permite administrar sus bases de datos y resolver problemas de forma independiente.
Azure portal Query editor (preview)	**De lenguaje natural a SQL:** Esta experiencia en el editor de consultas de Azure Portal para Azure SQL Database traduce las consultas de lenguaje natural a SQL, lo que hace que las interacciones de la base de datos sean más intuitivas.

Ver aka.ms/sqlcopilot-de-interview para más información

T-SQL RegEx support en Azure SQL DB

Ventajas

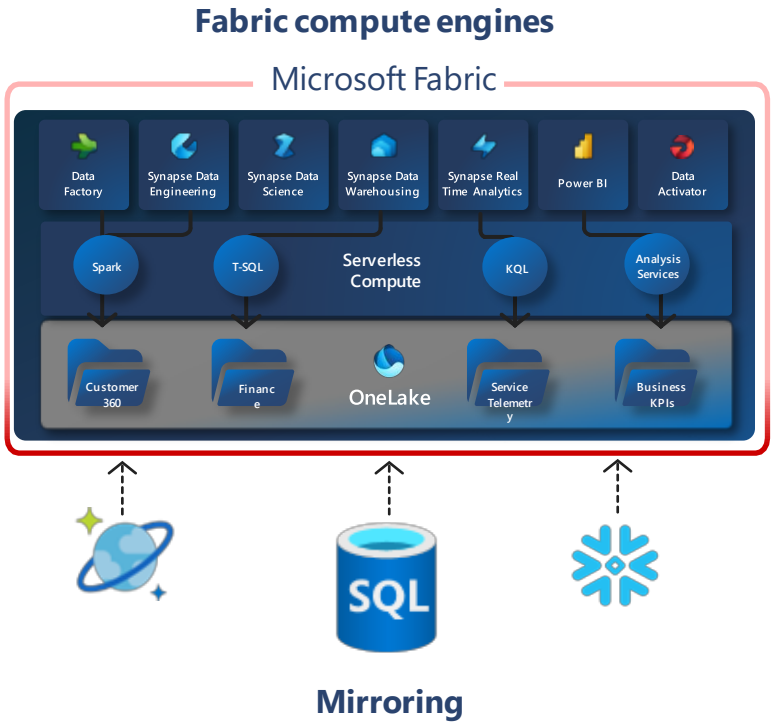
- Flexibilidad de busqueda de patrones
- Eficiencia en manipulacion y consultas

Soporte actual

- POSIX standard
- Soporta CS, agrupamientos, simbolos de RegEx y patrones de busqueda

Private Preview

- Disponible en todos los tiers



Fabric Mirroring allows replicating databases and data warehouses to Fabric without any ETL

Data is replicated into OneLake in Delta format and kept up-to-date in near real-time

All the Fabric experiences instantly work with the OneLake replica, including data warehousing experiences

Analysts and Data Scientists can work with bear real-time data

The replica protects operational databases from analytical queries

Storage and replication for mirroring is included with your capacity for no additional charge

Demo

Copilot en Azure SQL

Mas lanzamientos

- Database Watcher (Preview)
- SQL Server enabled by Arc
- SQL Server on Azure VMs
  - Soporte para Premium v2 SSDs

Local development

Carlos Robles

Senior Product Manager

Microsoft

roblescarlos@microsoft.com

linkedin.com/in/croblesm

github.com/croblesm

With 14+ years of experience working in tech, his experience spans a breadth of roles, such as Software Developer, DBA / Manager, Solutions Architect, and Consultant.

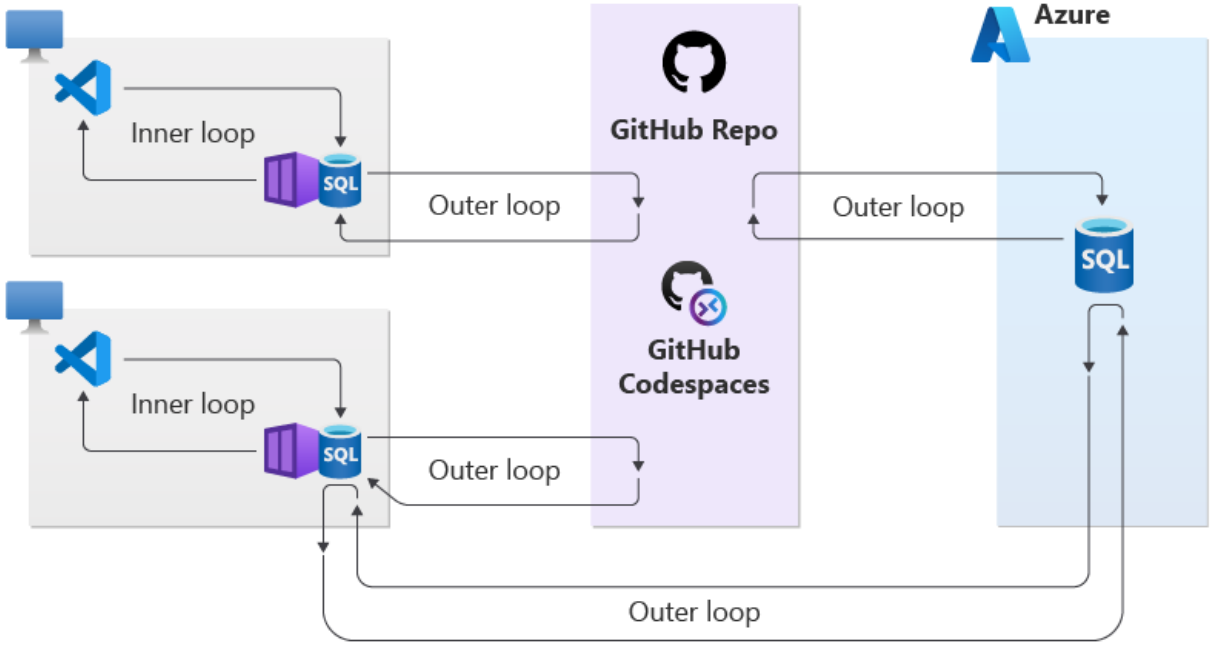
Former Amazon Web Services employee and Microsoft MVP, part of Redgate 100 – DevOps.

International speaker, author, mentor, community leader, geek, and gamer.

Inner and outer loop

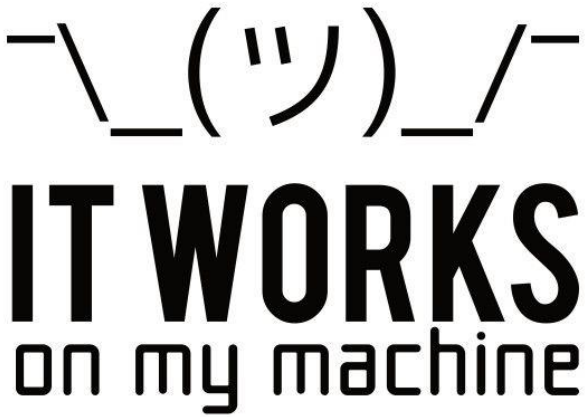
(Go) SQLCMD
Dev Containers
SQL Database Projects

GitHub Actions
Static Web Apps



Local development

- Where
  - Local machine
  - Remote / cloud
  - Container
- What
  - Docker / Podman
  - VS Code
  - Other IDEs
- How?



SQLCMD

- Command line utility
  - Administration, automation, and development
  - New version built on Golang
- Cross platform
  - winget, choco → Windows
  - apt, rpm, yum → Linux
  - brew → macOS
- Create SQL Containers
  - One-and-done experience
  - Docker /Podman

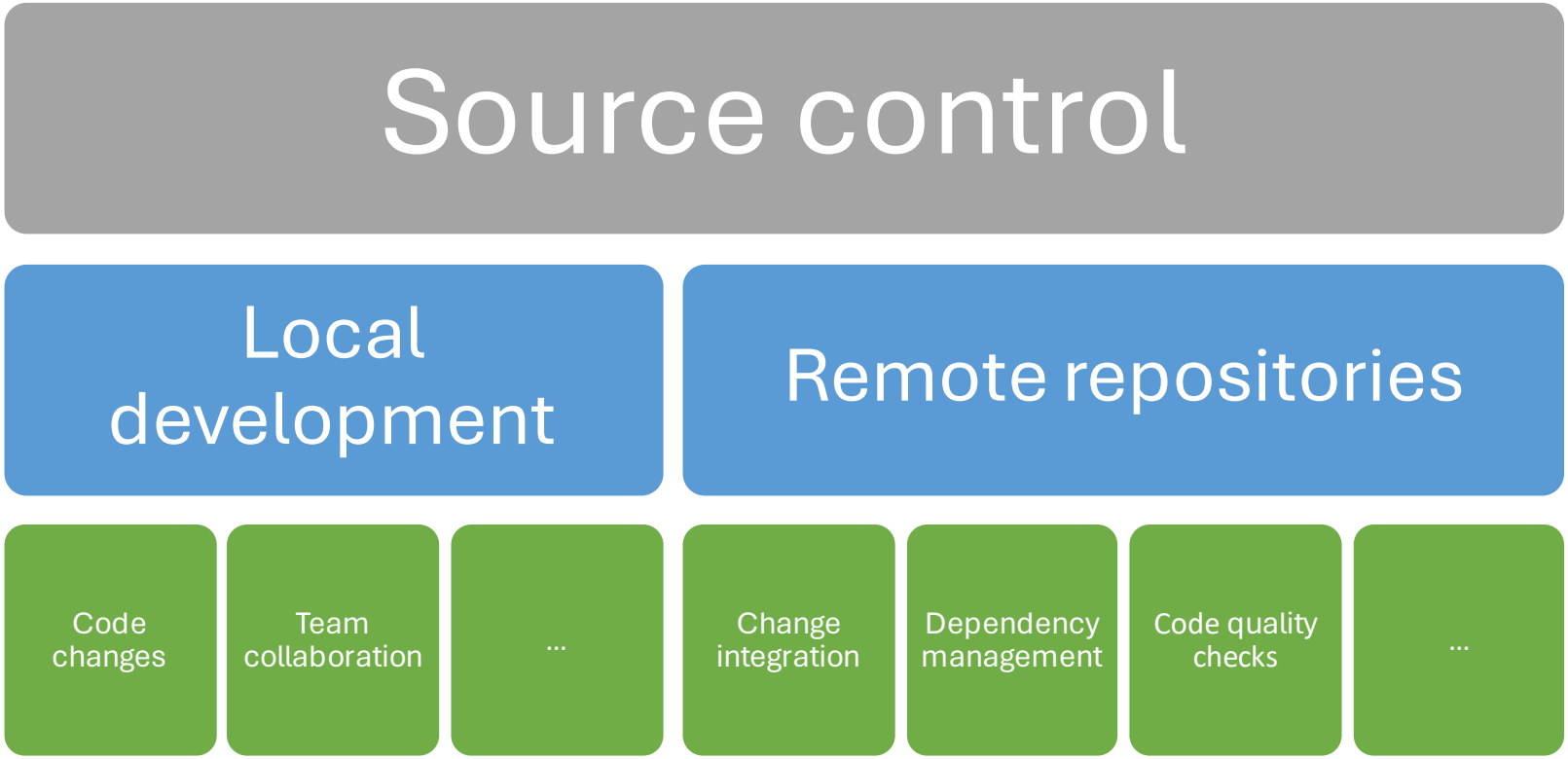
Create SQL container with SQLCMD

- `sqlcmd create mssql [flags]`
- `sqlcmd create mssql --accept-eula`

```
@croblesm → /workspaces/go-sqlcmd-examples (main) $ sqlcmd create mssql --accept-eula
Downloading mcr.microsoft.com/mssql/server:latest
Starting mcr.microsoft.com/mssql/server:latest
Created context "mssql" in "/home/codespace/.sqlcmd/sqlconfig", configuring user account...
Disabled "sa" account (and rotated "sa" password). Creating user "codespace"
Now ready for client connections on port 1,433

HINT:
  1. Run a query:                sqlcmd query "SELECT @@version"
  2. Start interactive session: sqlcmd query
  3. View sqlcmd configuration: sqlcmd config view
  4. See connection strings:     sqlcmd config connection-strings
  5. Remove:                     sqlcmd delete
```

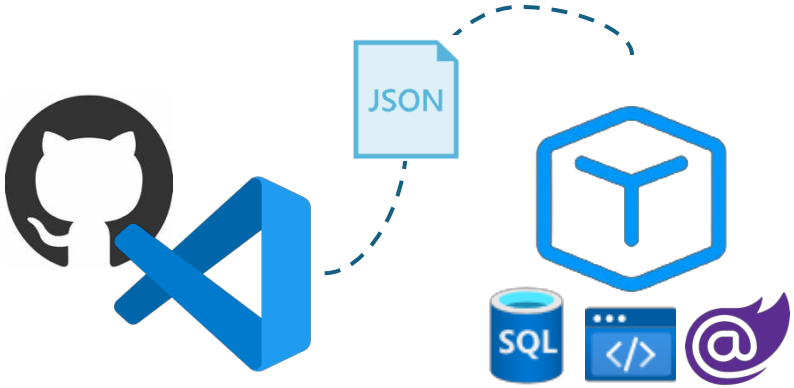
Demo

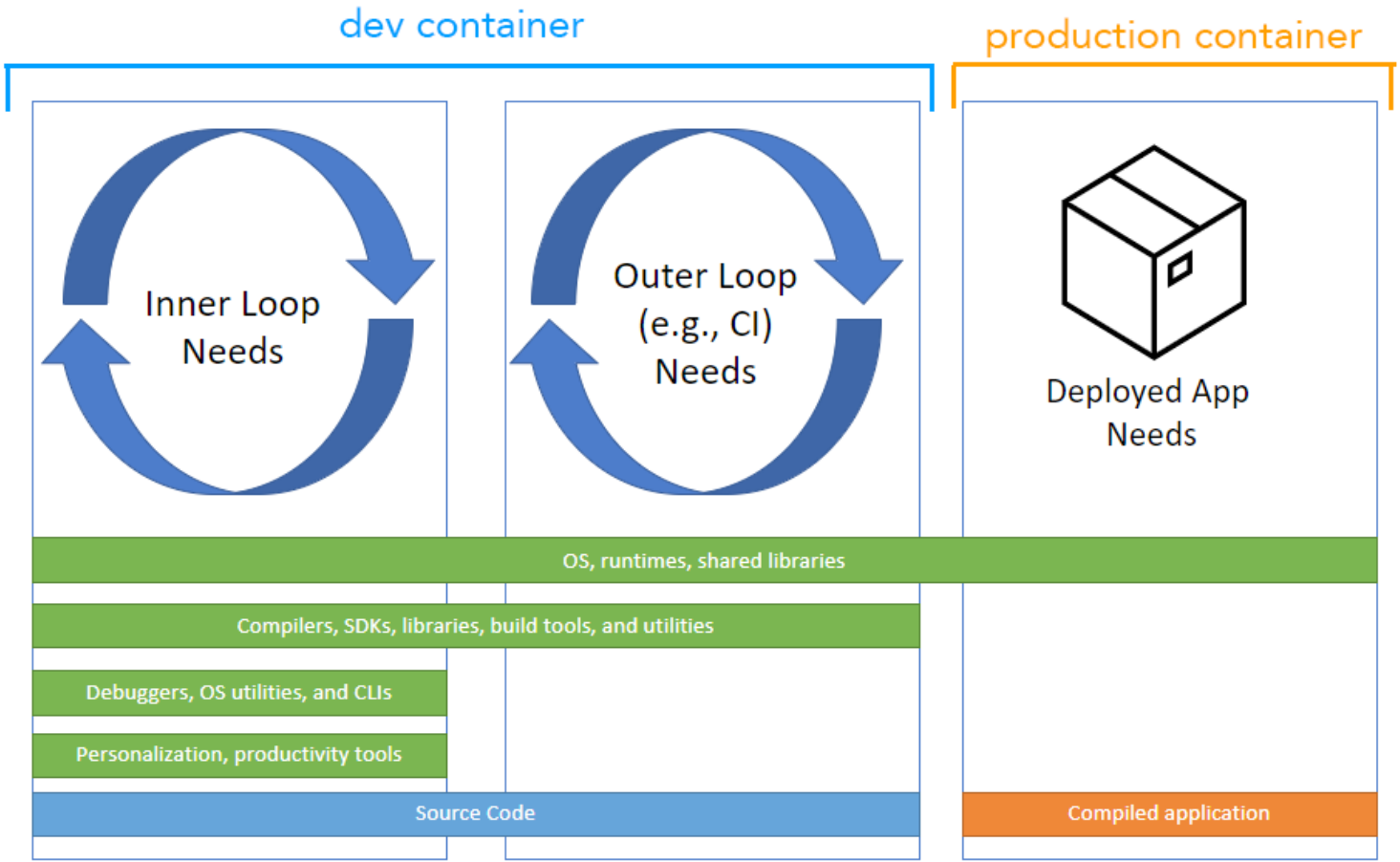
~~Source control~~ Source of truth



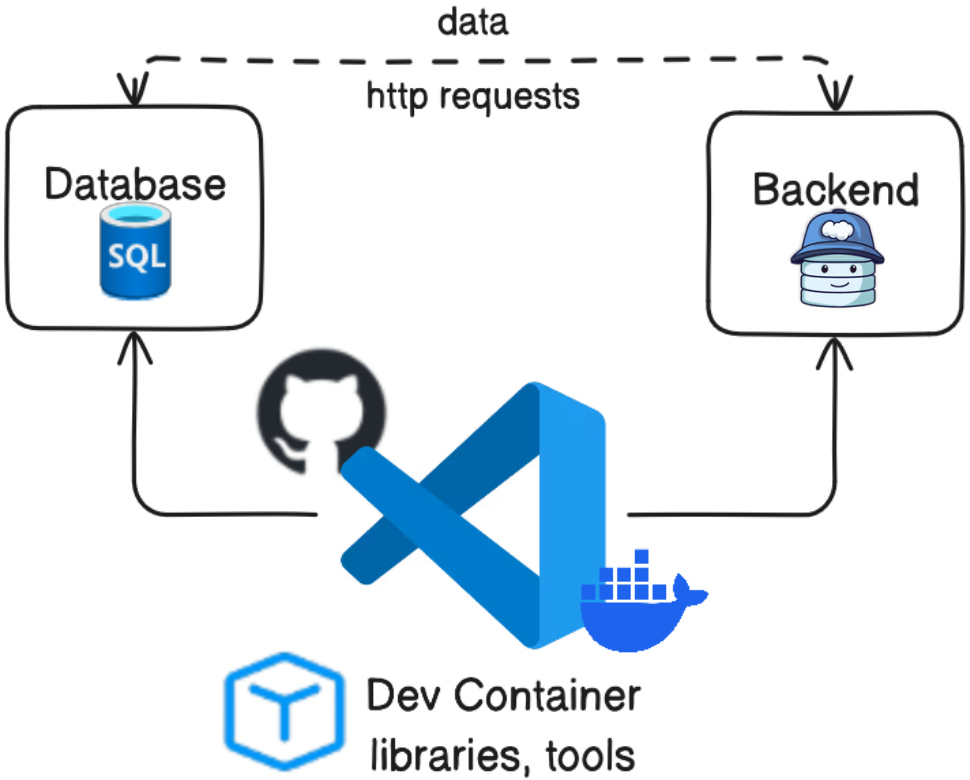
Dev Containers

- Full-featured development environment
- Run apps, specific tools, libraries, codebases
- Definition (metadata) → `devcontainer.json`
- Build your own templates




dev container
production container
Inner Loop Needs
Outer Loop (e.g., CI) Needs
Deployed App Needs
OS, runtimes, shared libraries
Compilers, SDKs, libraries, build tools, and utilities
Debuggers, OS utilities, and CLIs
Personalization, productivity tools
Source Code
Compiled application

Example



aka.ms/try-dab

```
@croblesm → /workspace (main) $ tree
├── .git
├── .devcontainer
│   ├── devcontainer.json
│   ├── docker-compose.yml
│   ├── Dockerfile
│   └── sql
│       ├── installSQLtools.sh
│       ├── library.azure-sql.sql
│       └── postCreateCommand.sh
├── DAB-Config
│   ├── dab.config.json
│   └── .env
└── README.md
```

Demo

Database Development Approaches

Migration-Based

- Hand crafted scripts
- Flyway
- Liquibase
- dbUp

State-Based

- SQL Database projects
- Microsoft.Build.Sql – SDK-style
- MSBuild.Sdk.SqlProj

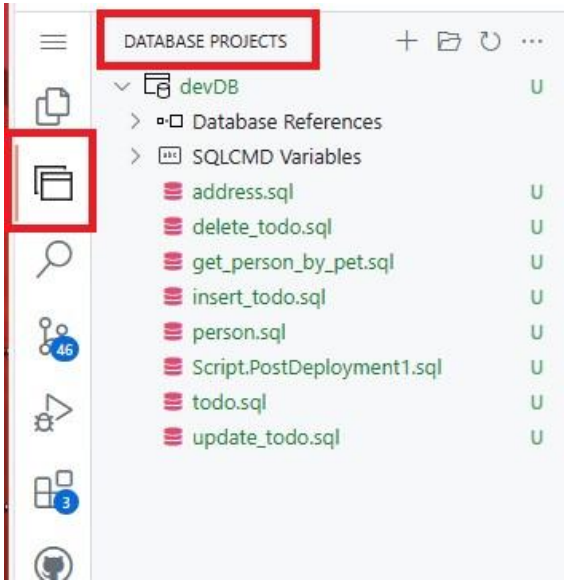
Automation platforms:
Azure DevOps, GitHub Actions, Jenkins, GitLab, etc.

SQL Database Projects

- Manage, design, deploy Azure SQL databases
  - VS Code, GitHub Codespaces and ADS extension
  - SDK-style SQL projects, declarative model
  - Database validation (locally and against target)
- Other capabilities
  - Version control
  - Compare schemas
  - CI/CD integration (dacpac)
  - Import, edit, build, publish

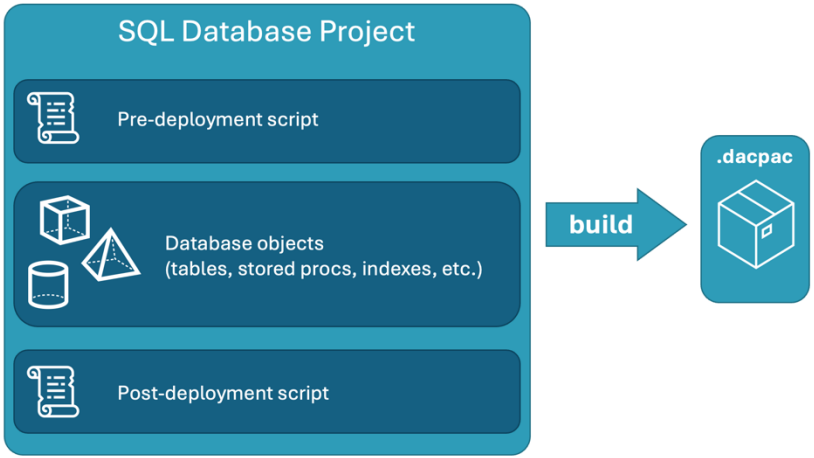
SQL Database Projects

- `dotnet new sqlproj -n [name] -tp [Azure SQL offering]`
- `dotnet build`
- `dotnet publish`



Advanced capabilities

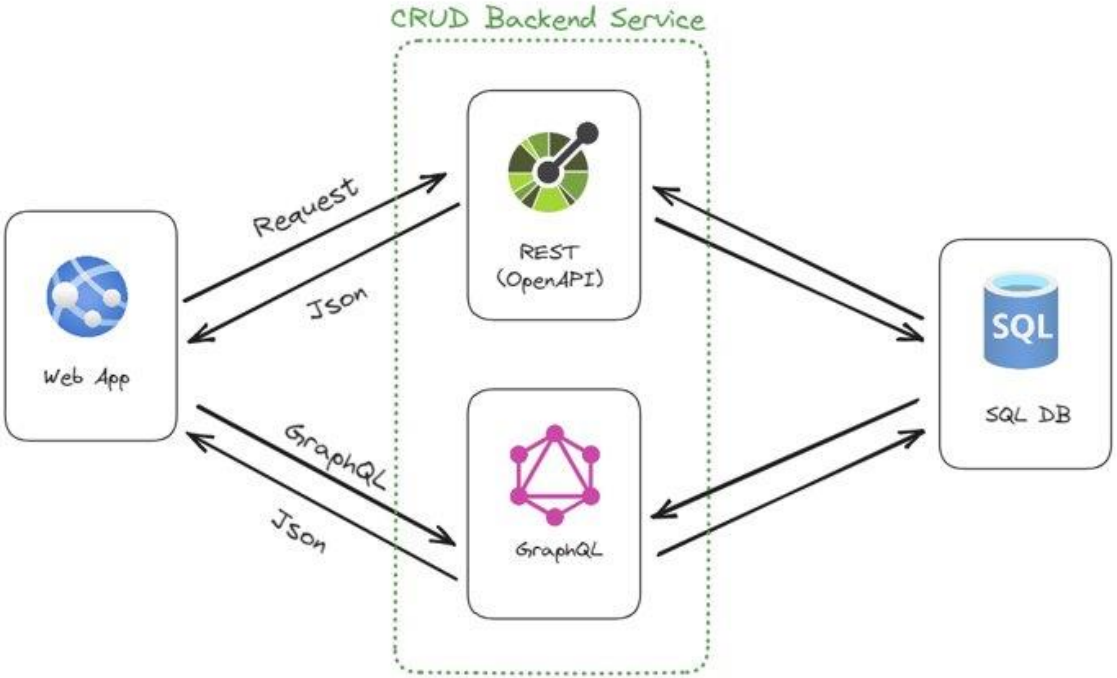
- Pre/Post deployment scripts
  - Initialize database (static data)
  - Add users
- Database References
  - Variable evaluation (SQLCMD)
  - Dependent/nested projects



Demo

Cloud development

(Basic) Modern Cloud Architecture



Azure SQL bindings for Azure Functions

- Azure SQL
  - Encompasses all Azure SQL products
  - SQL Server with cloud connectivity
- Azure Functions
  - Serverless runtime for standalone use
  - Integrated with Azure Static Web Apps
- Popular Languages
  - C#, JavaScript, Python, Java, and PowerShell

Bindings

Input (read data into function from database)

object in function

SQL query results

Azure SQL Input bindings take a SQL query or stored procedure to run and returns the output to the function.

Output (save data from function into database)

object in function

SQL table

Azure SQL Output bindings take a list of rows and update / inserts them to the user table.

SQL Trigger

Fires a function after a data change

data change

starts function

The Azure SQL trigger uses SQL change tracking functionality to monitor a SQL table for changes.

Use cases

- Data enrichment
  - OpenAI, geocoding, call REST/GraphQL from Azure SQL
- Start complex processing
  - Call another function
- Update websites
  - Broadcast SignalR messages
- Integrate with event-based architectures
  - Send data to EventHubs
- Create a change data stream
  - Stream analytics

SQL Input binding

Input (read data into function from database)

- Query text or stored procedure name
- Parameters, optional
- Connection string

object in function

SQL query results

Sample API: product list

API response


```
[
 {
  "ProductId” : 779,
  "Name” : "Mountain-200 Silver, 38”
 },
 {
  "ProductId” : 780,
  "Name” : "Mountain-200 Silver, 42”
 },
 {
  "ProductId” : 781,
  "Name” : "Mountain-200 Silver, 46”
 },
 ...
]
```

Sample C# Input Binding

SampleInputBinding.cs

```
[FunctionName("SampleInputBinding")]
public static IActionResult Run(
    [HttpTrigger(AuthorizationLevel.Function, "get", Route = getProducts)]
        HttpRequest req,
    [Sql("SELECT ProductId, Name FROM dbo.Products",
        CommandType = System.Data.CommandType.Text,
        ConnectionStringSetting = "SqlConnectionString")]
        IEnumerable<Object> sqlResult)
{
      return new OkObjectResult(sqlResult);
}
```

Sample JavaScript Input Binding

index.js

```
module.exports = async function
  (context, req, products)
{
  context.res = {
    body: products
  };
}
```

function.json

```
{
  "bindings": [
  …
  {
    "direction": "in",
    "type": "sql",
    "name": "products",
    "commandText": "SELECT ProductId, Name FROM
    Products",
    "commandType": "Text",
    "connectionStringSetting":
    "SqlConnectionString"
  },
  …
  ]
}
```

SQL output binding

Output (save data from function into database)

- Destination table
- Connection string

object in function

SQL table

Sample API: new product

POST body to API

```
{
    "Name": "Test Product",
    "ProductNumber": "10",
    "Color": "Red",
    "StandardCost": 10.00,
    "ListPrice": 20.00,
    "ProductCategoryID": 30,
    "ProductModelID": 1,
    "Weight": 12
}
```

Sample C# Output Binding

SampleOutputBinding.cs

```
[FunctionName("SampleOutputBinding")]
public static IActionResult Run(
    [HttpTrigger(AuthorizationLevel.Function, "post", Route = AddProducts)]
        HttpRequest req,
    [Sql("dbo.Products", ConnectionStringSetting = "SqlConnectionString")]
        out Object newProduct)
{
      // set newProduct attributes
      return new OkObjectResult(newProduct);
}
```

Sample JavaScript Output Binding

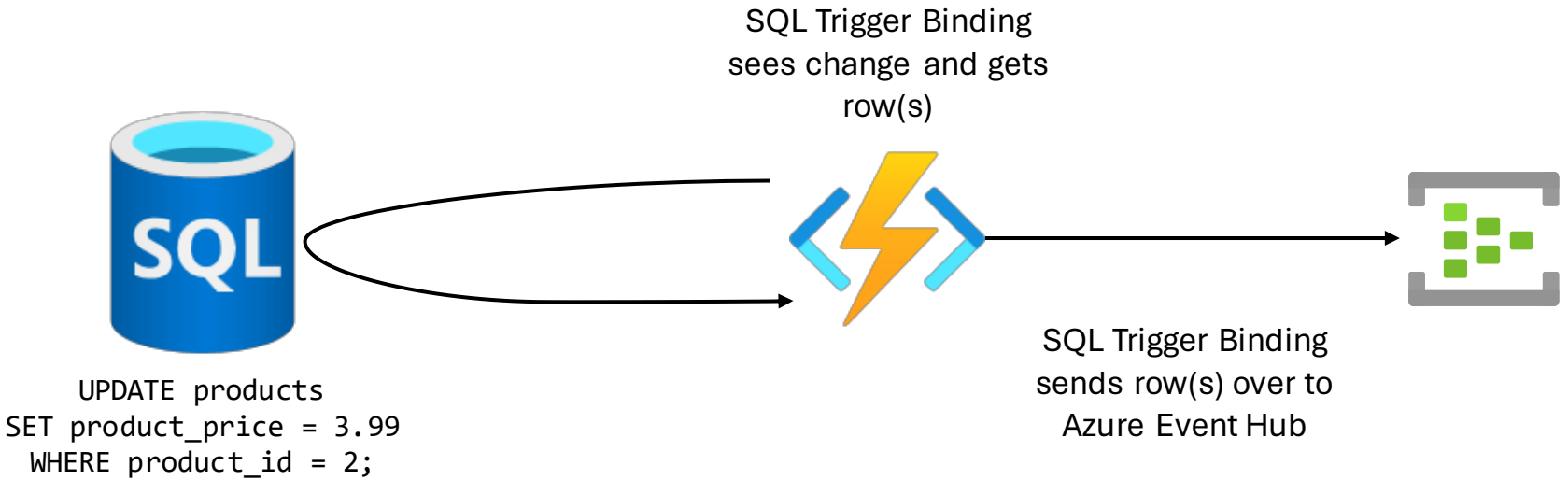
index.js

```
module.exports = async function
  (context, req)
{
  let newproduct = "";
  if (req.body) {
    newproduct = req.body;
    context.bindings.newproduct =
newproduct;
  }

  context.res = {
    body: newproduct
  };
}
```

function.json

```
{
  "bindings": [
  …
  {
    "direction": "out",
    "type": "sql",
    "name": "newproduct",
    "commandText": "dbo.Products",
    "connectionStringSetting":
    "SqlConnectionString"
  },
  …
  ]
}
```

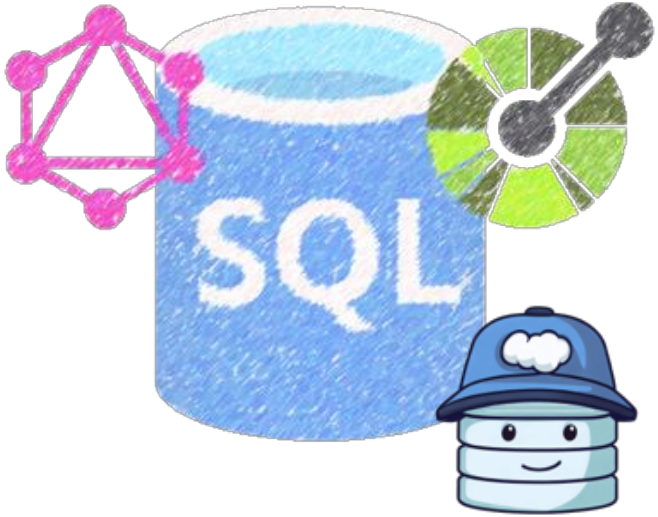
Example


SQL Trigger Binding
sees change and gets
row(s)
SQL
UPDATE products
SET product_price = 3.99
WHERE product_id = 2;
SQL Trigger Binding
sends row(s) over to
Azure Event Hub

Demo

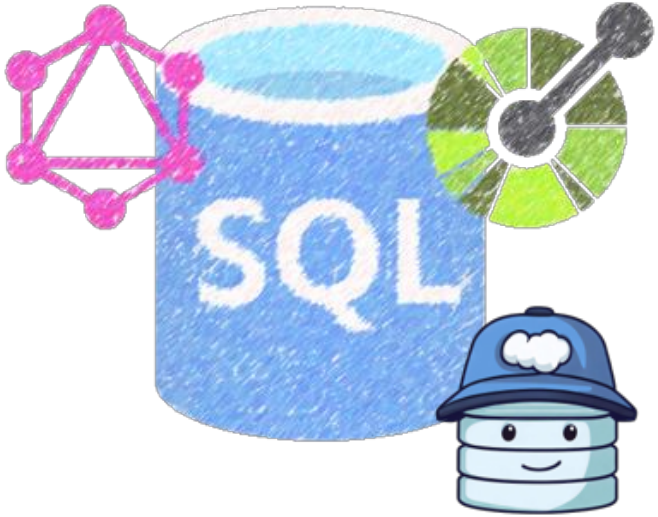
Data API builder

- Provides REST and GraphQL endpoints to your Azure Databases
  - Tables, views and stored procedures
- With full support for
  - Pagination
  - Sorting
  - Filtering
  - Selection
  - Relationships (GraphQL)

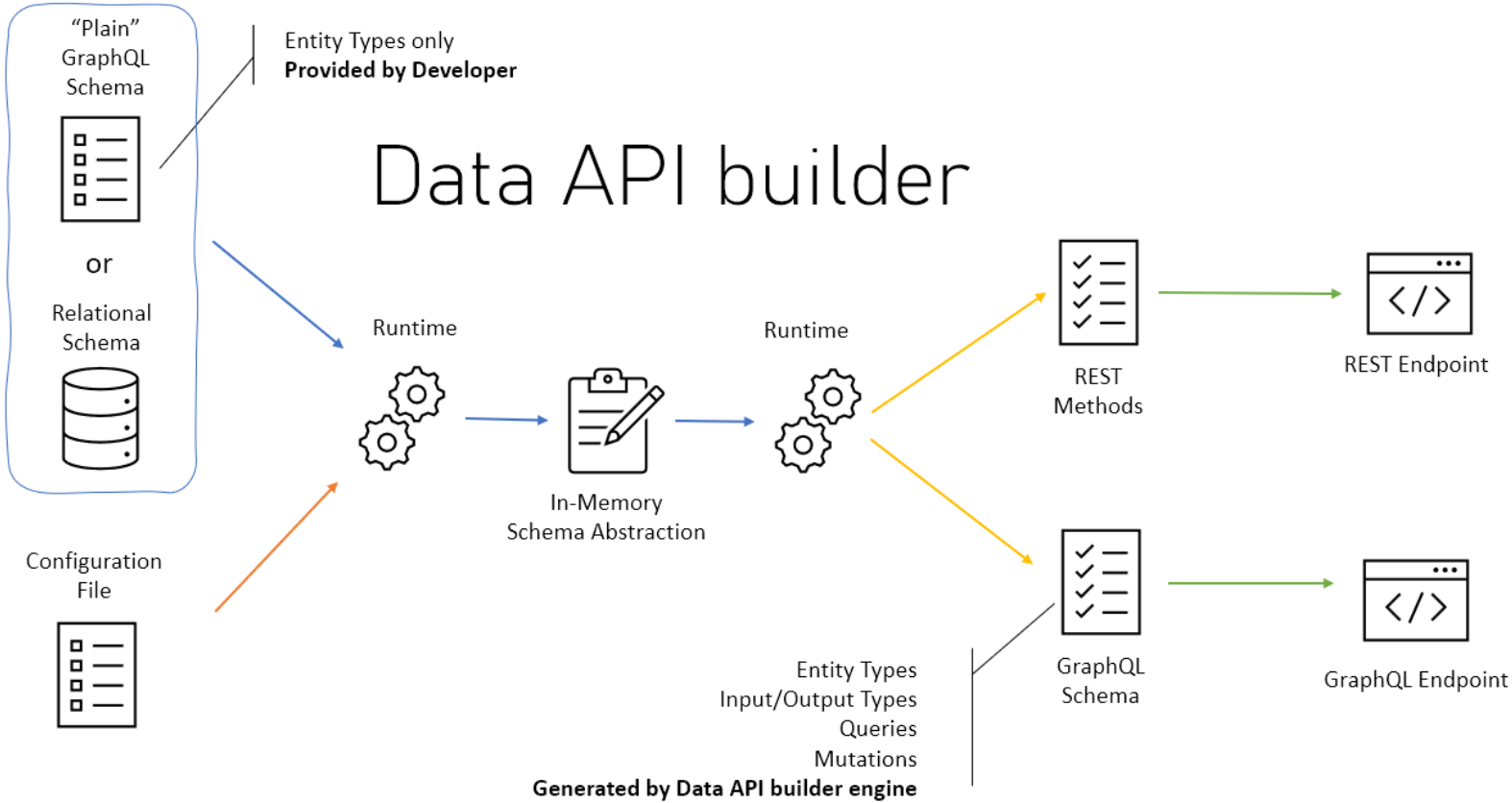


Data API builder

- Granular security
  - EasyAuth, Entra ID, JWT
  - Roles-based authentication
  - Permissions at entity level
- Multiple deployment options
  - Local
  - Container
- Simplifies development
  - Azure Static Web App integration

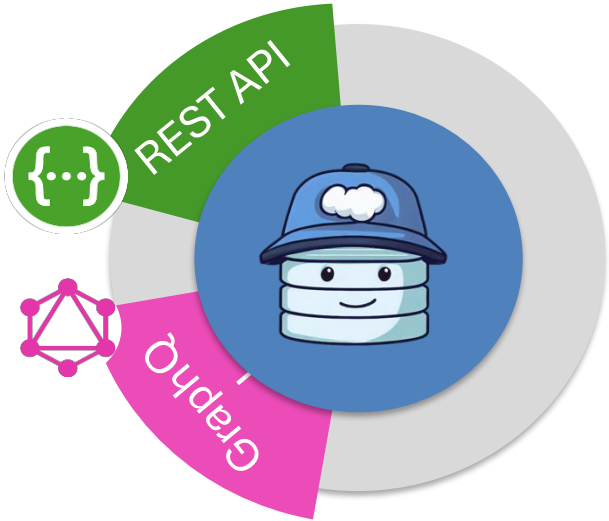


Data API builder



Initialization / configuration

```
dab init
    --config [file name] \
    --database-type [db engine] \
    --connection-string [db url]

dab add [entity name] \
    --source [db object] \
    --permissions [role:actions]

dab start
```



Demo

Microsoft

Data API builder & Azure SQL Dev Container

Carlos Robles
Senior Product Manager

SQL Developer Experience Team

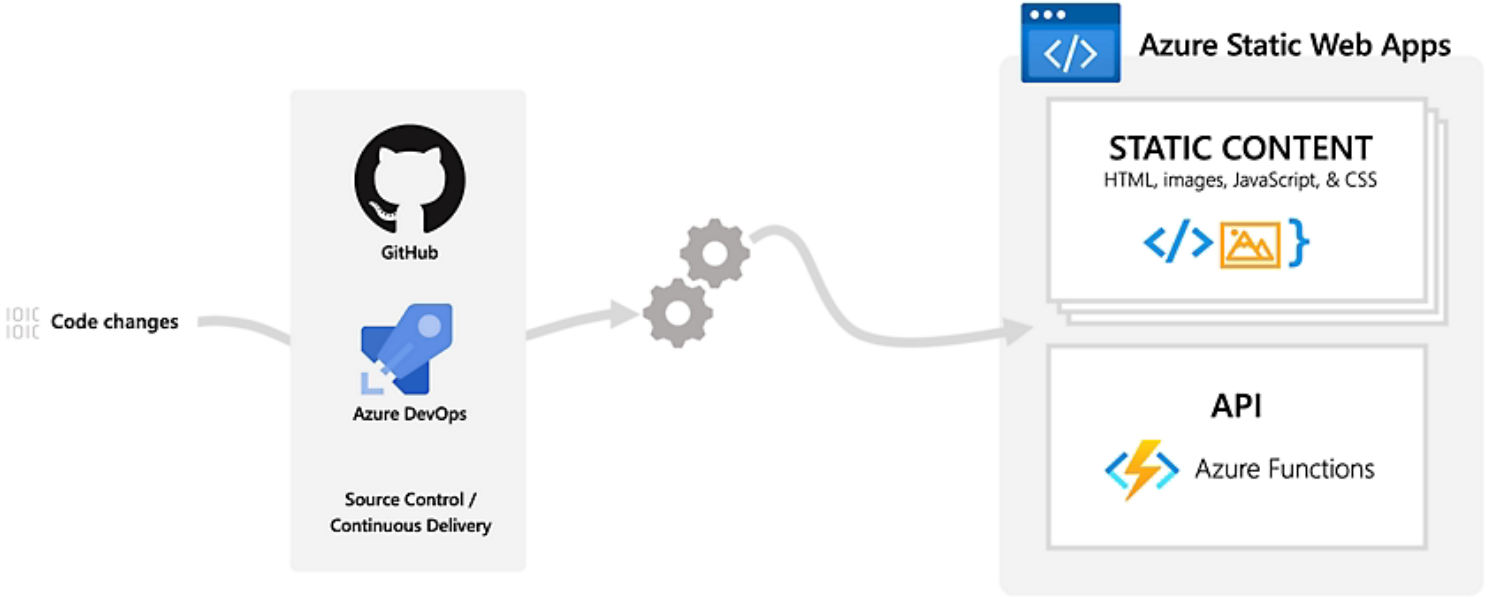
Azure Static Web Apps

- Build and deploy serverless full stack static web apps to Azure from a code repository
  - HTML, CSS, JavaScript and images
  - Vue, React, Angular, Blazor, Next.js, Nuxt.js and more
- Serverless APIs powered by Azure Functions
- Database connection string (Preview)
- Automated content geo-distribution
- Easy GitHub and Azure DevOps integration
  - Staging versions based on PRs

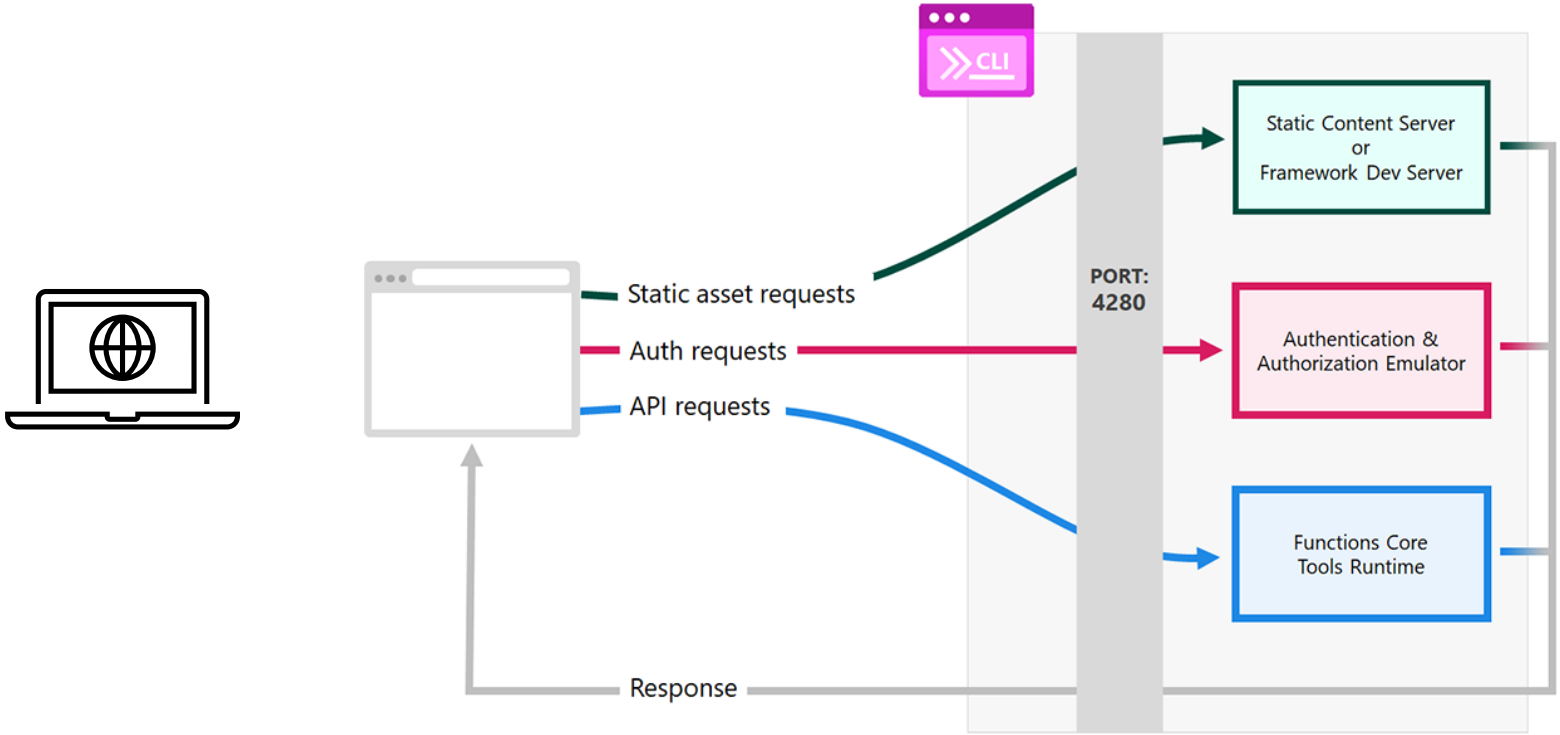
Azure Static Web Apps

- Custom domains
- Seamless security model
  - RBAC, reverse proxy
- Authentication integration
  - Microsoft Entra ID, GitHub
- Backend routing rules
- CLI support
  - Local development (emulator)

Azure Static Web Apps

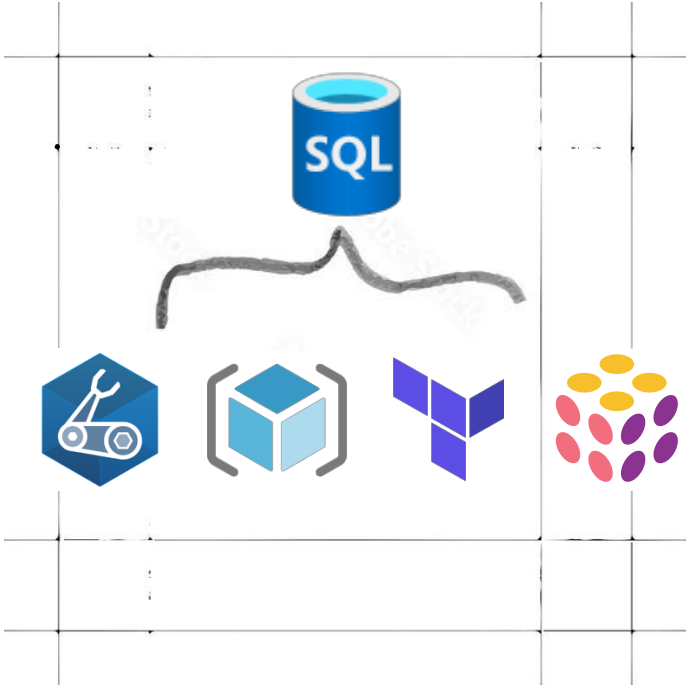


Azure Static Web Apps

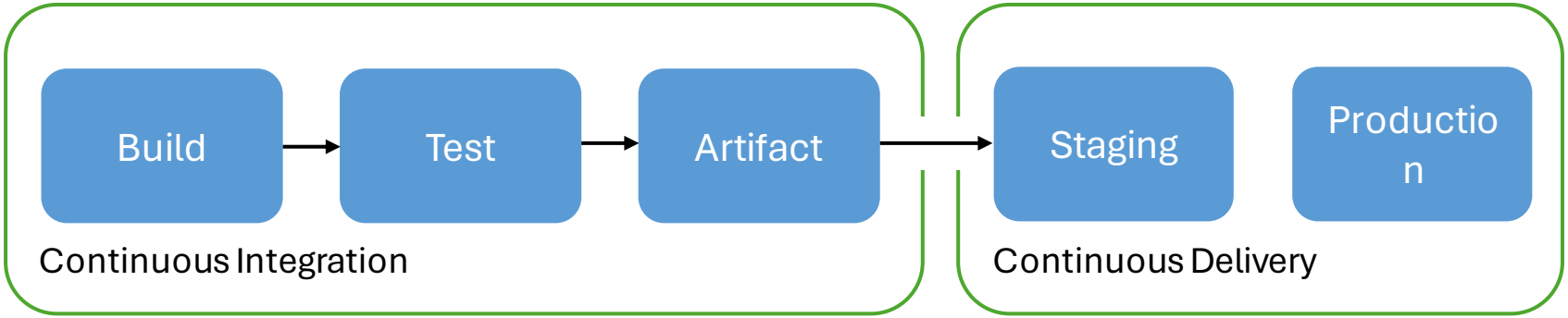


Infrastructure as Code + Azure SQL

- IaC for short
  - Automation, consistency and scalability
- Options for Azure SQL?
  - ARM / Bicep
  - Terraform
  - Pulumi
- Benefits and use cases
  - Repeatability
  - Replication of environments
  - Reduces human errors



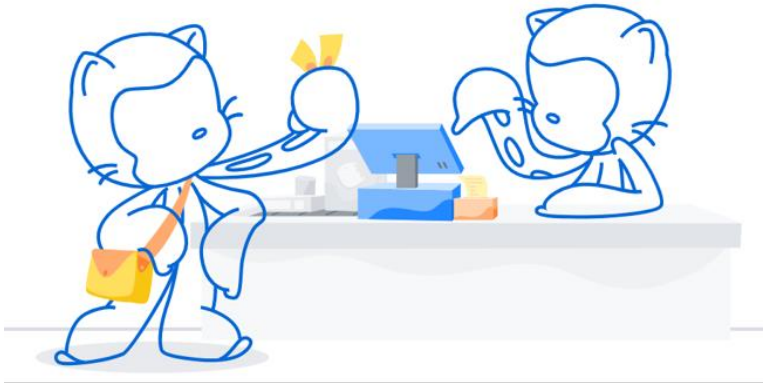
CI/CD


Build
Test
Artifact
Staging
Productio
n
Continuous Integration
Continuous Delivery

GitHub actions for Azure SQL

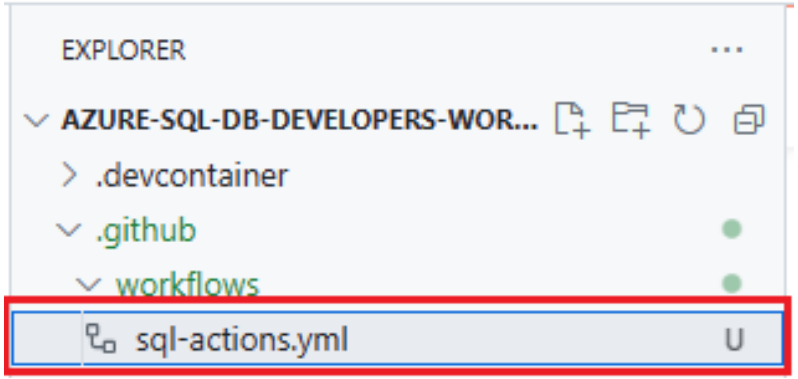
- Automate workflows (CI\CD) in GitHub
  - Automated builds, test and deployments
  - YAML syntax
  - Large action library

- Azure SQL use cases:
  - Deploy schema changes
  - Run data migrations
  - Execute SQL scripts
  - Validate database state



Create a workflow

- GitHub actions are in the `.github/workflows` folder
- Can contain multiple action workflows



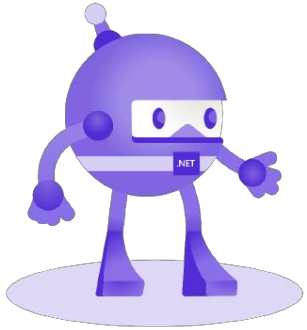
Example

```yaml
jobs:
  build:
    # SDK sqlproj build on Windows + Linux
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Azure SQL Deploy
        uses: Azure/sql-action@v2.2.1
        with:
          # The connection string
          connection-string: ${{ secrets.AZURE_SQL_CONNECTION_STRING }}
          # Path to DACPAC artifact
          path: ./Database.sqlproj
           action: Publish
```

Demo

.NET Aspire

Modernizando tus aplicaciones con .NET Aspire



.NET Aspire

Agenda

- Aplicaciones Distribuidas
- Que son las aplicaciones Nativas para la Nube
- Qué es .NET Aspire
- Prerequisitos para utilizar .NET Aspire
- Características de .NET Aspire
- Demo inicial
- Azure Developer CLI (azd)
- Dev Containers y .NET Aspire
- Demo de despliegue

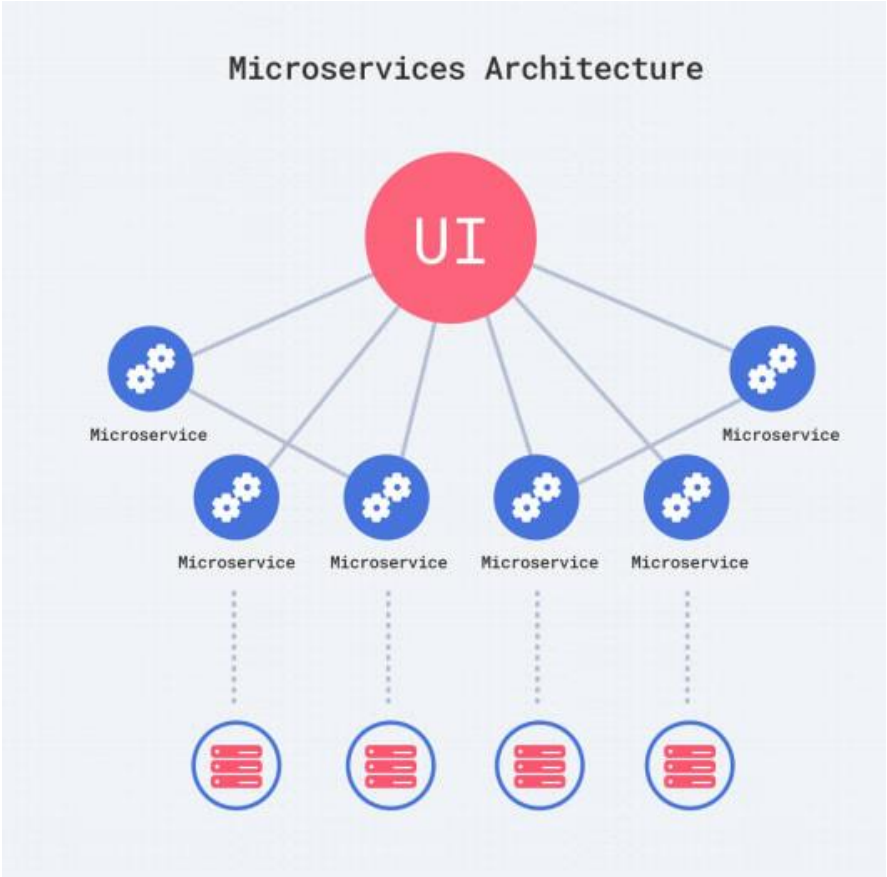
¡Hola!

Geovani de León
Cloud Solution Architect at PlusTI
CoLíder de la comunidad Tech Community GT

Guatemalteco

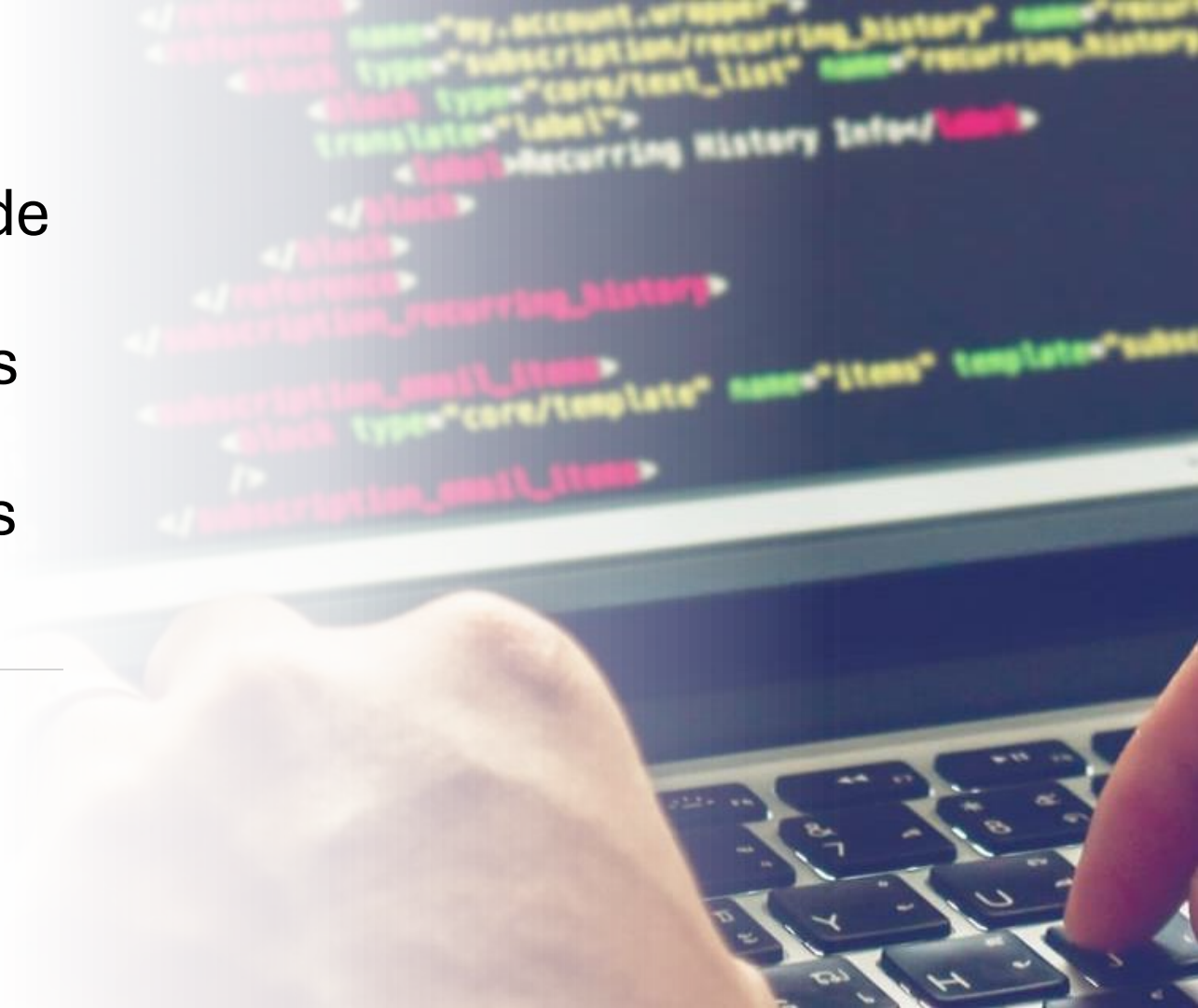
https://github.com/yovafree
https://yovadeleon.dev

Aplicaciones Distribuidas / Microservicios



Las aplicaciones distribuidas / microservicios son difíciles

Crear un entorno de desarrollo local para aplicaciones distribuidas / microservicios es difícil



Consideraciones

01

Una aplicación distribuida es aquella que utiliza recursos computacionales en múltiples nodos, como contenedores que se ejecutan en diferentes hosts.

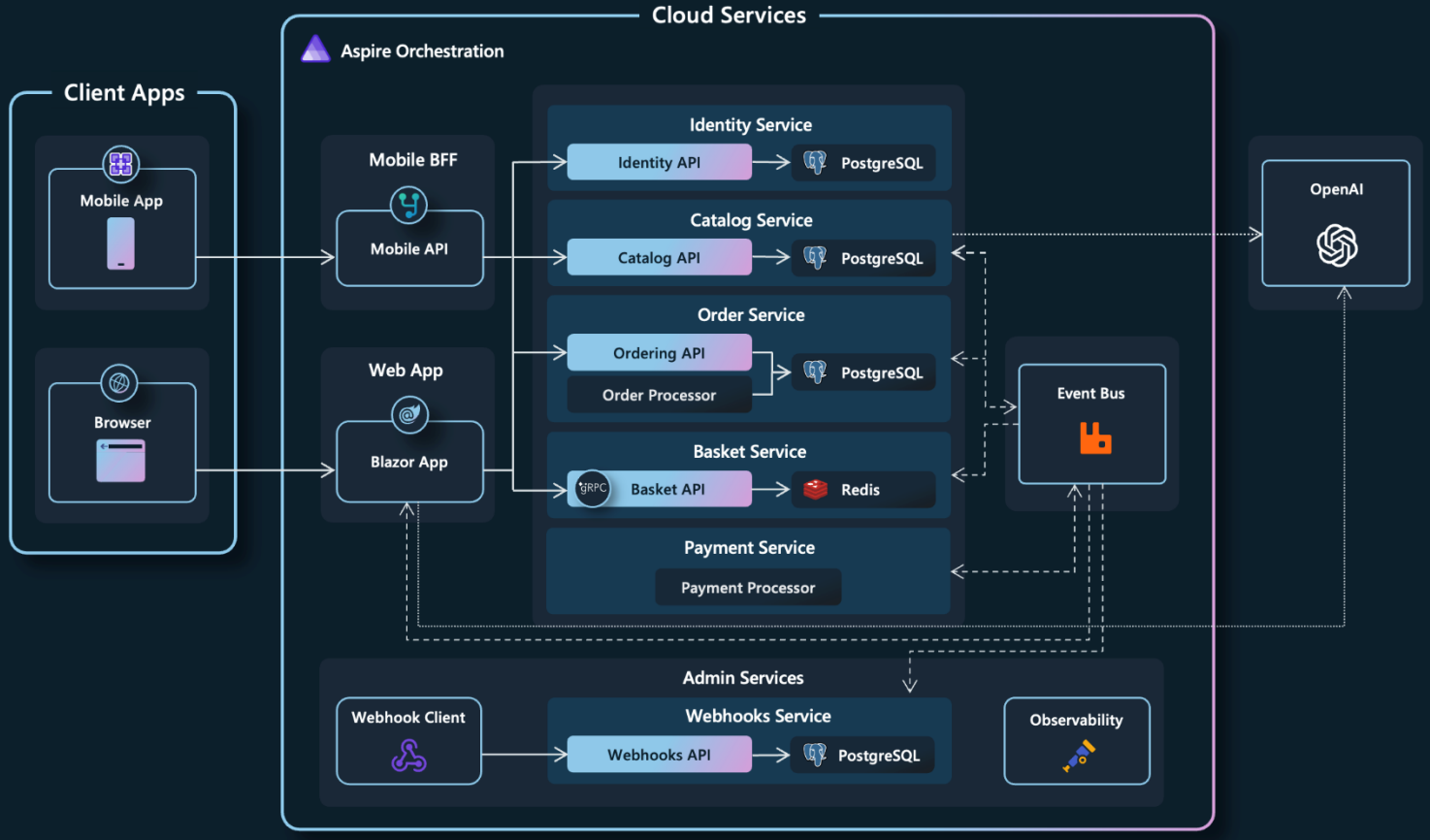
02

Las aplicaciones nativas de la nube generalmente consumen una gran cantidad de servicios, como bases de datos, mensajería y almacenamiento en caché.

03

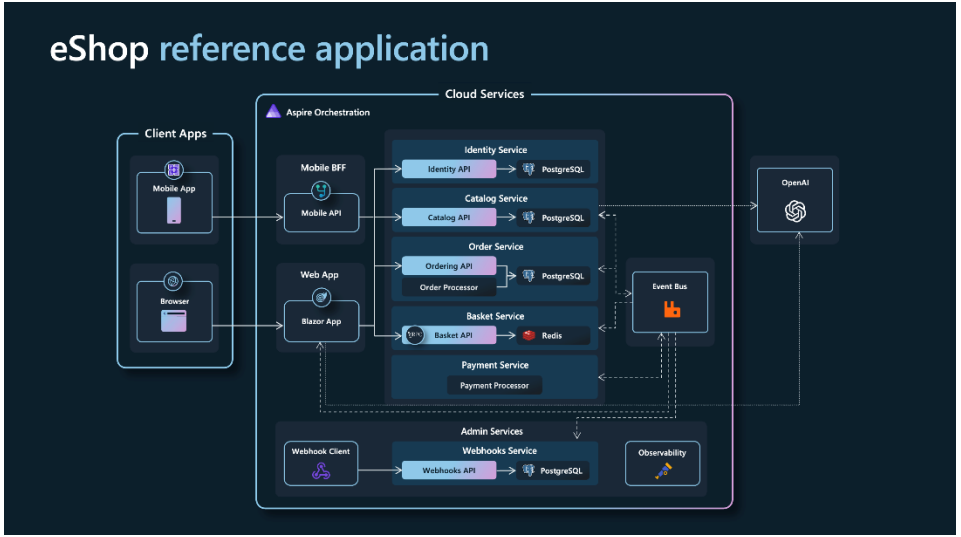
Una aplicación nativa de la nube es un tipo específico de aplicación distribuida que aprovecha al máximo la escalabilidad, la resiliencia y la capacidad de administración de las infraestructuras de la nube.

eShop reference application



Arquitectura de Referencia eShop

- Ejecuta múltiples servicios interconectados
  - Expresar la estructura del sistema.
  - Configuración del descubrimiento del servicio
  - Múltiples instancias
- Desarrollo orientado a producción: confiabilidad, controles de estado de salud, registros, telemetría, etc.
- Experiencia local para observabilidad y depuración.



Qué es .NET Aspire

- .NET Aspire es un stack para construir aplicaciones distribuidas observables y listas para producción orientadas a la nube.
- Se entrega a través de una colección de paquetes NuGet que implementan características específicas nativas de la nube.
- .NET Aspire está diseñado para mejorar la experiencia de crear aplicaciones .NET nativas de la nube. Proporciona un conjunto de herramientas y patrones que ayudan a crear y ejecutar aplicaciones distribuidas.

Qué nos ofrece .NET Aspire

proporciona funciones para ejecutar y conectar aplicaciones de múltiples proyectos y sus dependencias para entornos de

.NET Aspire son paquetes NuGet para servicios de uso común, como Redis o Postgres, con interfaces estandarizadas que garantizan que se conecten de manera consistente y sin problemas con su

Herramientas: .NET Aspire viene con plantillas de proyecto y experiencias de herramientas para Visual Studio y dotnet CLI para ayudarlo a crear aplicaciones .NET Aspire e interactuar con ellas.

.NET Aspire

.NET Aspire

Composición y Orquestación:

Lenguaje de modelado para definir los recursos del sistema y sus referencias.

Un runtime para ejecutar y conectar aplicaciones multiproyecto y sus referencias.

Componentes:

Paquetes de servicios de uso común como Redis o PostgreSQL, con interfaces y características estandarizadas.

Herramientas:

Dashboard

Aplicaciones distribuidas VS soporte

Valores predeterminados del servicio

Utilidades de publicación o despliegue

.NET Aspire

Prerequisitos para utilizar .NET Aspire

Para trabajar con .NET Aspire, necesita lo siguiente instalado localmente:

- .NET 8.0
- Carga de trabajo de .NET Aspire:
  - Utilice el instalador de Visual Studio
  - Utilice el comando dotnet workload install aspire
- Docker Desktop
- Entorno de desarrollador integrado (IDE) o editor de código, como por ejemplo:
  - Visual Studio 2022 Preview version 17.9 o superior (opcional)
  - Código de Visual Studio (opcional)

Ver: https://learn.microsoft.com/en-us/dotnet/aspire/fundamentals/setup-tooling?tabs=dotnet-cli%2Cwindows

.NET Aspire

.NET Aspire

Demo

Composición y Orquestación

Repositorio de las demostraciones

https://github.com/yovafree/modernizing-apps-with-dotnet-aspire

.NET Aspire

Orquestación en .NET Aspire

Orquestación en .NET Aspire: Simplifica la gestión de la configuración e interconexiones de la app nativa de la nube.

No reemplaza sistemas robustos como Kubernetes en producción.

Ofrece abstracciones para facilitar la configuración de descubrimiento de servicios, variables de entorno y configuraciones de contenedores.

Elimina la necesidad de manejar detalles de implementación a bajo nivel.

Garantiza un patrón de configuración consistente en aplicaciones con numerosos componentes y servicios.

Facilita la gestión de aplicaciones complejas durante la fase de desarrollo.

.NET Aspire

VS Code - Creación de proyecto de ejemplo

```
dotnet new aspire-starter --use-redis-cache --output AspireSample
```

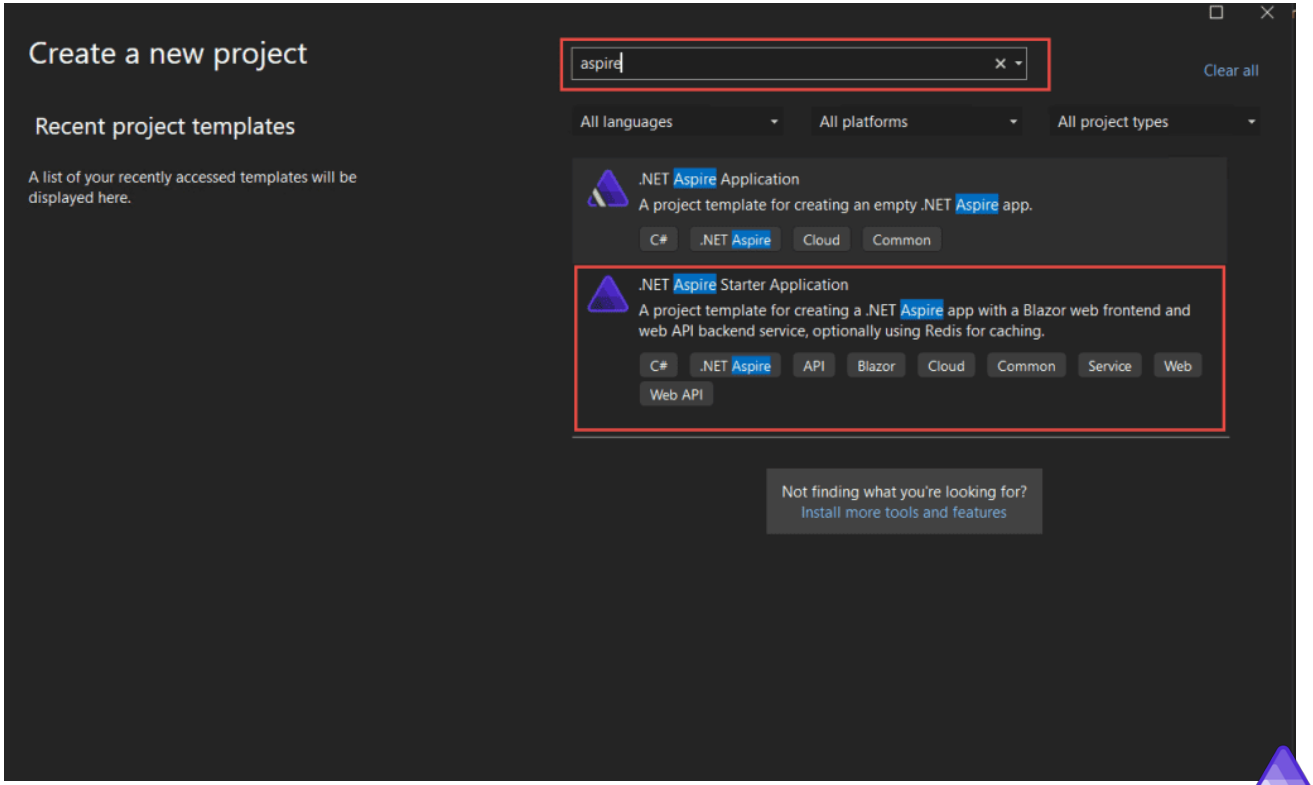
.NET Aspire

VS Code - Probar localmente el proyecto

```
dotnet run --project AspireSample/AspireSample.AppHost
```

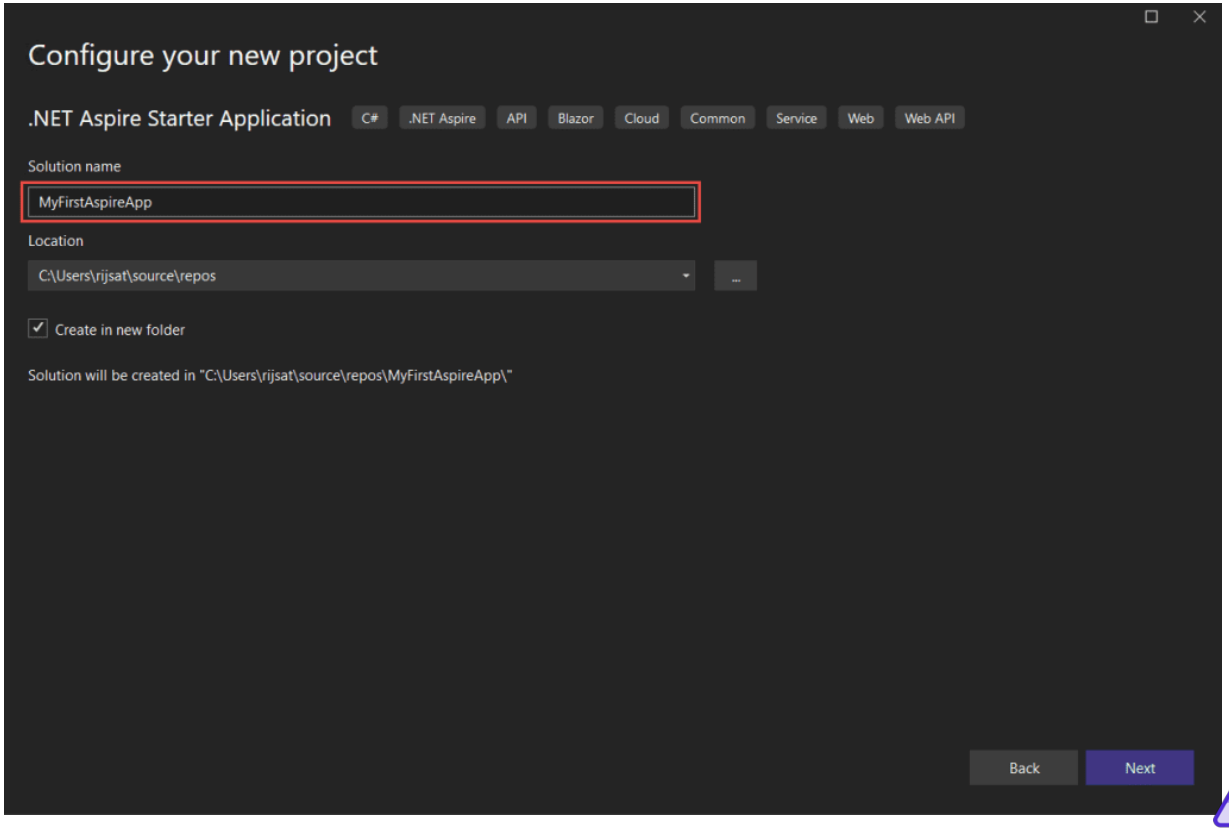
.NET Aspire

VS – Creación de proyecto de ejemplo



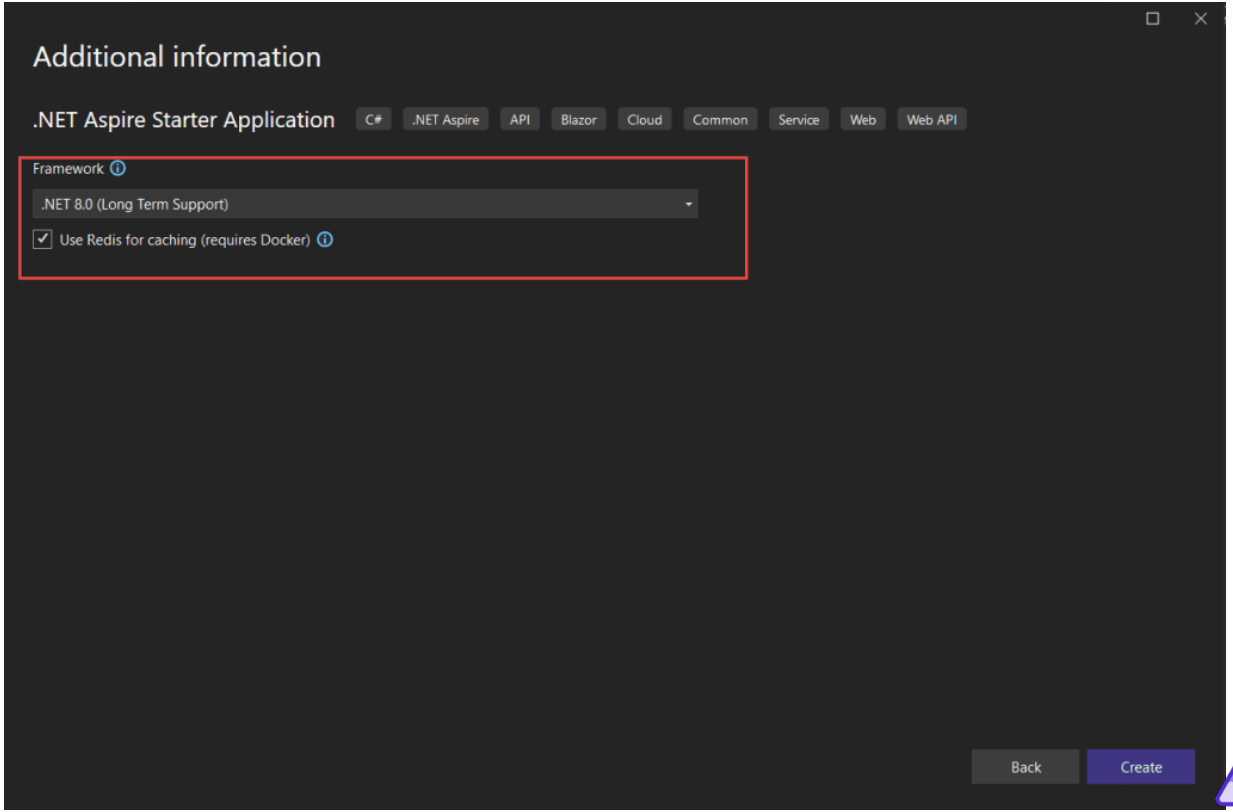
.NET Aspire

VS – Creación de proyecto de ejemplo



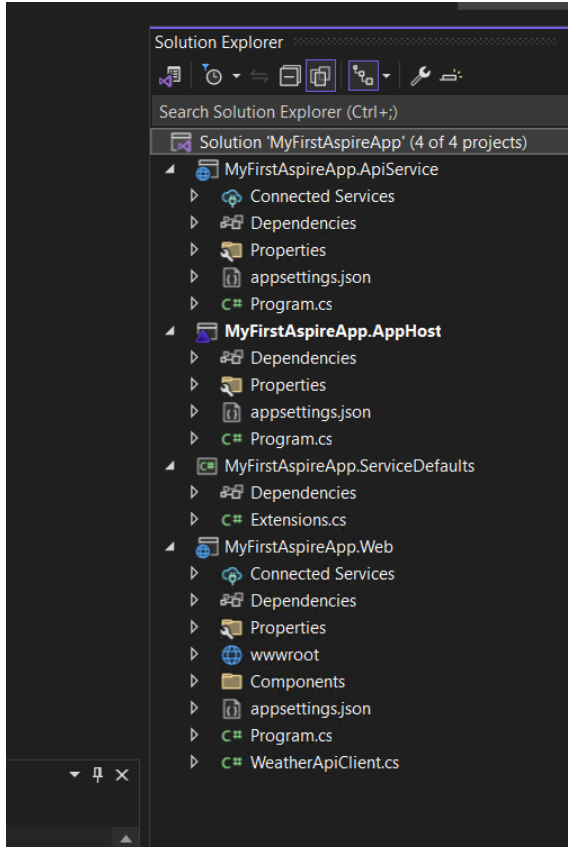
.NET Aspire

VS – Creación de proyecto de ejemplo



.NET Aspire

VS – Creación de proyecto de ejemplo



.NET Aspire

Exploremos el archivo program.cs del proyecto AppHost

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

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

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

builder.AddProject<Projects.MyFirstAspireApp_Web>("webfrontend")
    .WithReference(cache)
    .WithReference(apiservice);

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

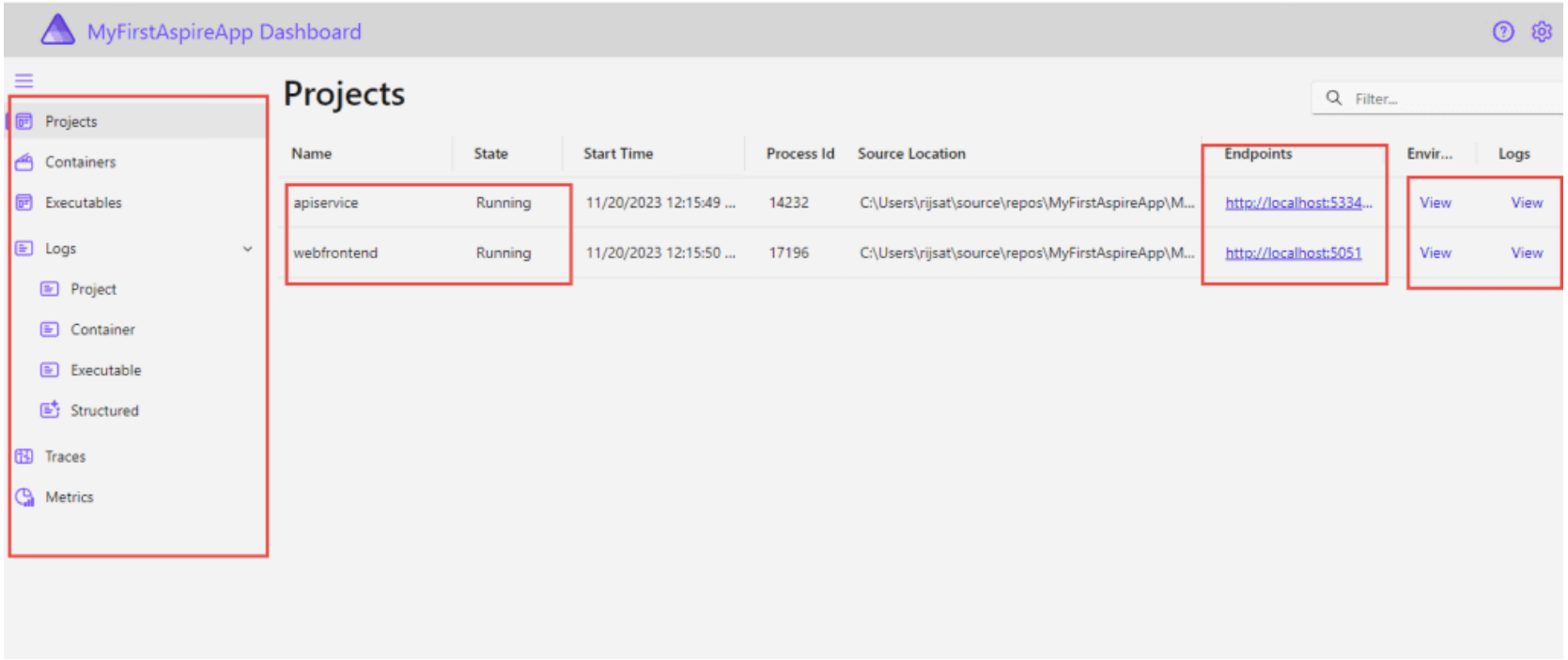
.NET Aspire

Exploraremos el program.cs del proyecto frontend

```
using MyFirstAspireApp.Web;
using MyFirstAspireApp.Web.Components;

var builder = WebApplication.CreateBuilder(args);

// Add service defaults & Aspire components.
builder.AddServiceDefaults();
builder.AddRedisOutputCache("cache");

// Add services to the container.
builder.Services.AddRazorComponents()
    .AddInteractiveServerComponents();

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

var app = builder.Build();

if  !app.Environment.IsDevelopment())
{ (
    app.UseExceptionHandler("/Error", createScopeForErrors: true);
}

app.UseStaticFiles();

app.UseAntiforgery();

app.UseOutputCache();

app.MapRazorComponents<App>()
    .AddInteractiveServerRenderMode();

app.MapDefaultEndpoints();

app.Run();
```

.NET Aspire

Dashboard



.NET Aspire

.NET Aspire

.NET Aspire

Componentes

.NET Aspire

Composición y Orquestación:

Lenguaje de modelado para definir los recursos del sistema y sus referencias.

Un runtime para ejecutar y conectar aplicaciones multiproyecto y sus referencias.

Componentes:

Paquetes de servicios de uso común como Redis o PostgreSQL, con interfaces y características estandarizadas.

Herramientas:

Dashboard

Aplicaciones distribuidas VS soporte

Valores predeterminados del servicio

Utilidades de publicación o despliegue

.NET Aspire

Estandarizar cómo utilizamos los recursos del sistema

Cada recurso del sistema del que depende nuestra aplicación debe ser consumido por un cliente estandarizado.

El cliente debe proporcionar

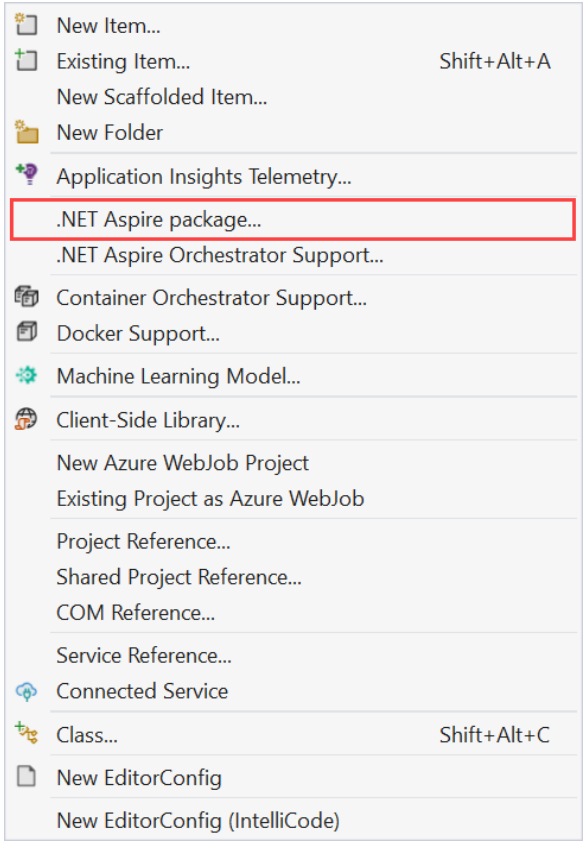
- Observabilidad y telemetría.
  - Registros
  - Traza
  - Métricas
- Controles de salud
- Resiliencia: la capacidad de su sistema para reaccionar ante fallas y seguir siendo funcional.
  - Reintentos
  - Tiempos de espera
  - Circuit breakers

Todo lo anterior debe ajustarse a las mismas convenciones (configuración, API surface, etc.)

Ver: https://github.com/dotnet/aspire/blob/main/src/Components/README.md

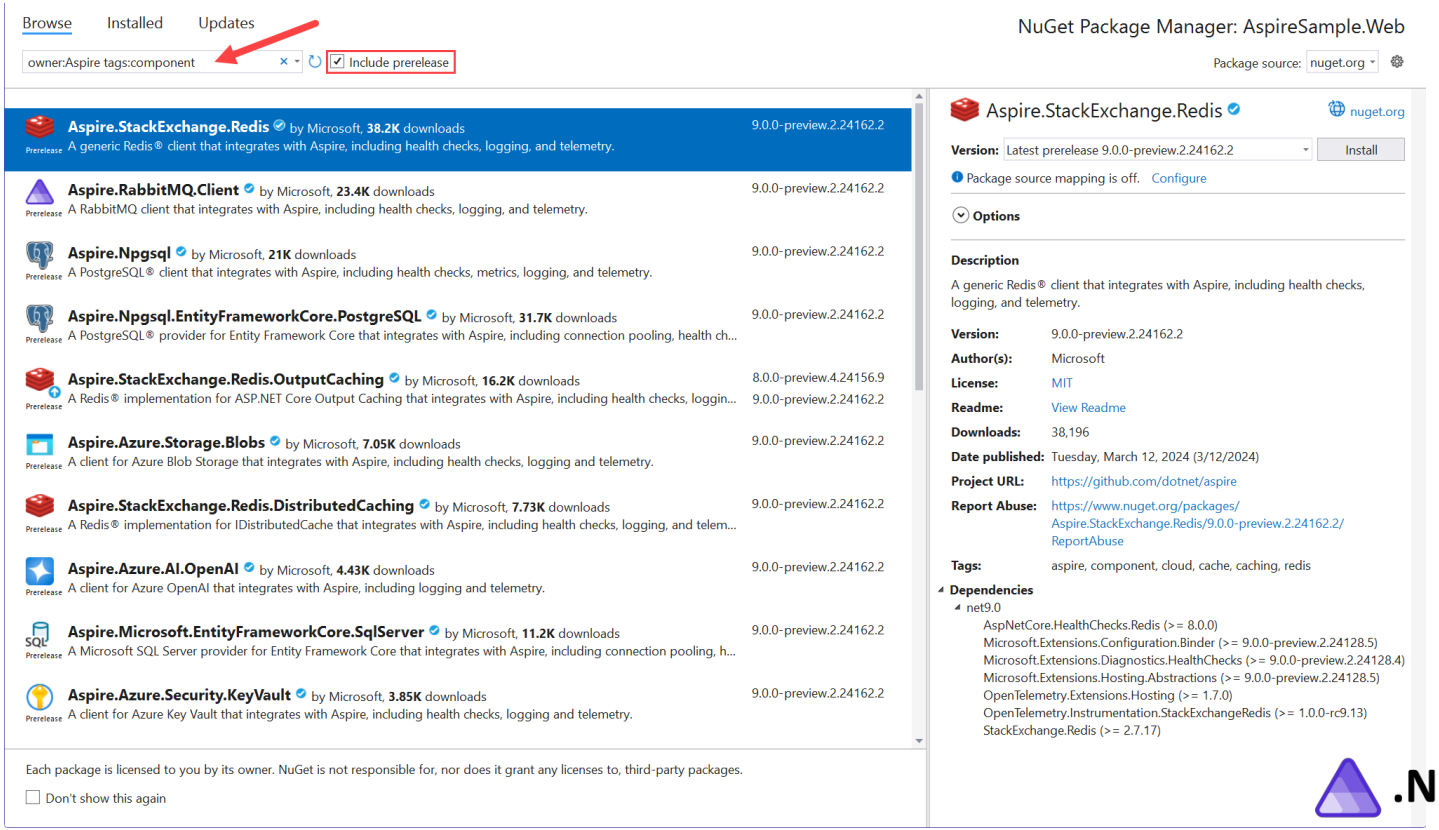
.NET Aspire

VS: Agregar un componente



.NET Aspire

VS: Agregar un componente



.NET Aspire

Componentes

Cloud-agnostic components

Component	Description
PostgreSQL Entity Framework Core	Provides a client library for accessing PostgreSQL databases using Entity Framework Core.
PostgreSQL	Provides a client library for accessing PostgreSQL databases.
RabbitMQ	Provides a client library for accessing RabbitMQ.
Redis Distributed Caching	Provides a client library for accessing Redis caches for distributed caching.
Redis Output Caching	Provides a client library for accessing Redis caches for output caching.
Redis	Provides a client library for accessing Redis caches.
SQL Server Entity Framework Core	Provides a client library for accessing SQL Server databases using Entity Framework Core.
SQL Server	Provides a client library for accessing SQL Server databases.

Azure specific components

Component	Description
Azure Blob Storage	Provides a client library for accessing Azure Blob Storage.
Azure Cosmos DB Entity Framework Core	Provides a client library for accessing Azure Cosmos DB databases with Entity Framework Core.
Azure Cosmos DB	Provides a client library for accessing Azure Cosmos DB databases.
Azure Key Vault	Provides a client library for accessing Azure Key Vault.
Azure Service Bus	Provides a client library for accessing Azure Service Bus.
Azure Storage Queues	Provides a client library for accessing Azure Storage Queues.

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

.NET Aspire

.NET Aspire

Demo 2

Componentes

Agregar PostgreSQL y RabbitMQ al proyecto

.NET Aspire

.NET Aspire

Herramientas

Publicación y Despliegue

.NET Aspire

Composición y Orquestación:

Lenguaje de modelado para definir los recursos del sistema y sus referencias.

Un runtime para ejecutar y conectar aplicaciones multiproyecto y sus referencias.

Componentes:

Paquetes de servicios de uso común como Redis o PostgreSQL, con interfaces y características estandarizadas.

Herramientas:

Dashboard

Aplicaciones distribuidas VS soporte

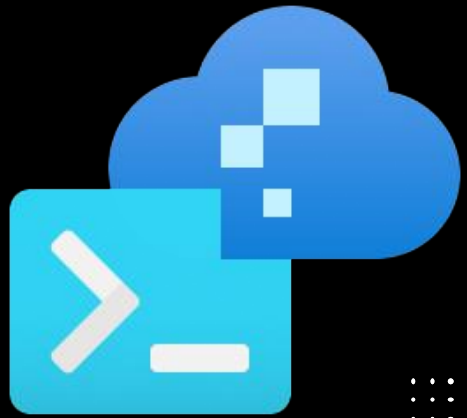
Valores predeterminados del servicio

Utilidades de publicación o despliegue

Qué es Azure Developer CLI (azd)

Azure Developer CLI (azd) es una herramienta de código abierto que acelera el tiempo necesario para llevar su aplicación del entorno de desarrollo local a Azure.

azd proporciona prácticas recomendadas y comandos fáciles de usar para desarrolladores que se asignan a etapas clave de su flujo de trabajo, ya sea que esté trabajando en la terminal, su editor o entorno de desarrollo integrado (IDE) o CI/CD (integración continua/implementación continua).



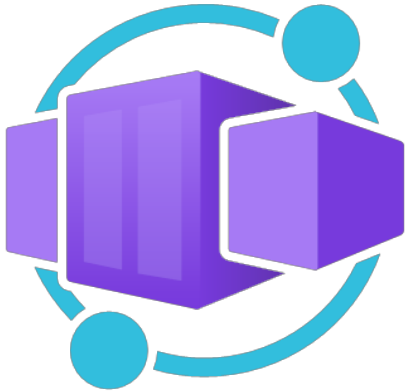
Qué es Azure Container Apps

Azure Container Apps es una plataforma sin servidor que reduce la infraestructura y costos al ejecutar aplicaciones en contenedores.

Proporciona recursos de servidor para mantener la estabilidad y seguridad de las aplicaciones, sin preocuparse por la configuración del servidor ni la orquestación de contenedores.

Sus usos más comunes incluyen:

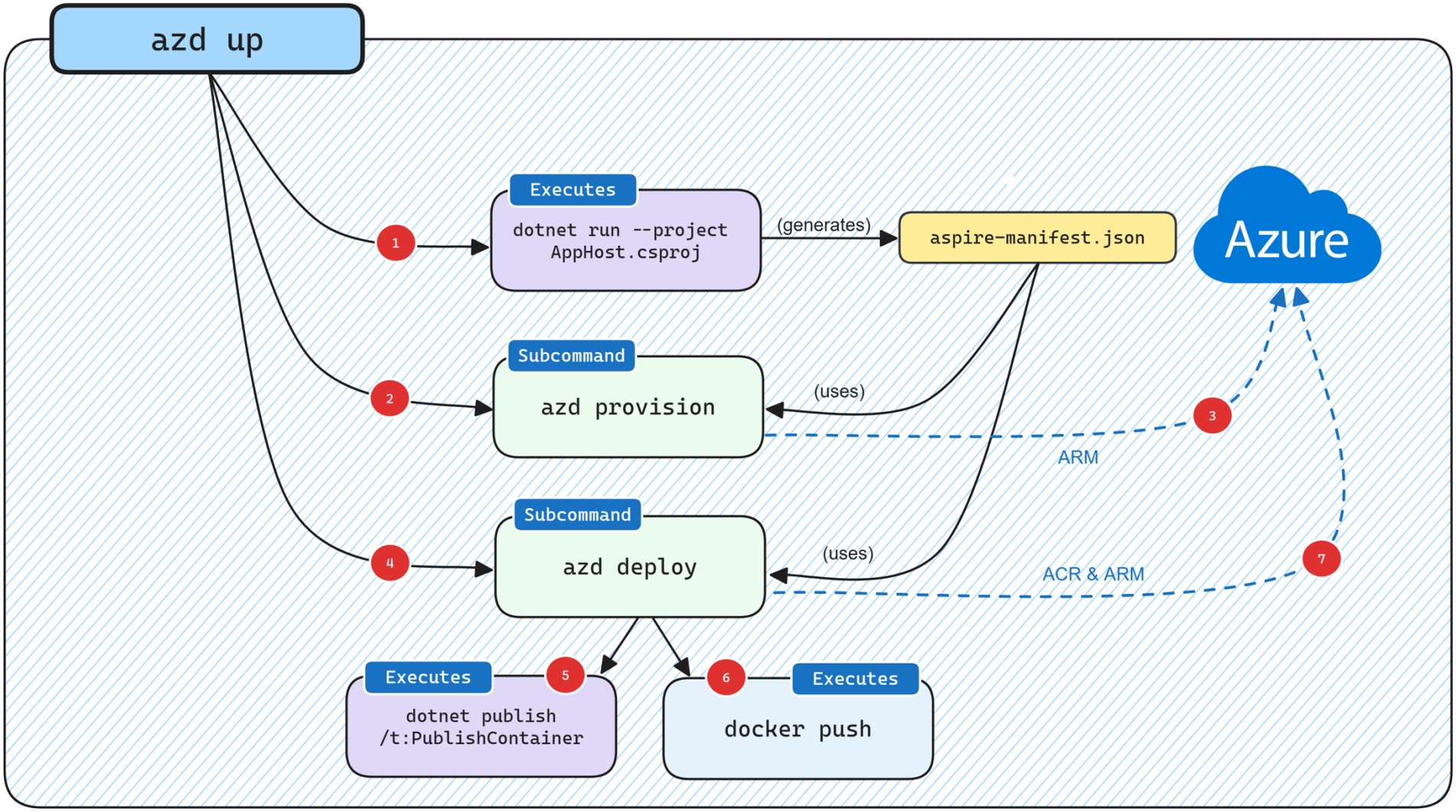
- Implementación de API endpoints
- Alojamiento de cargas de procesamiento en segundo plano
- Manejo de procesamiento basado en eventos
- Ejecución de microservicios



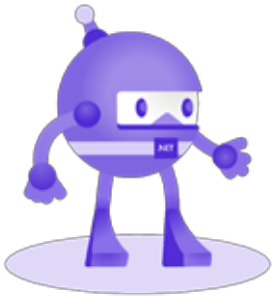
Demo 2

Despliegue de la App a Azure utilizando AZD

.NET Aspire

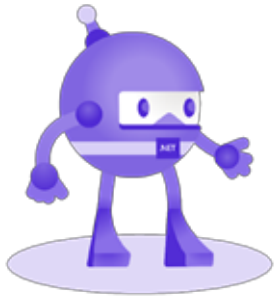

azd up
1
Executes
dotnet run --project AppHost.csproj
(generates)
aspire-manifest.json
Azure
2
Subcommand
azd provision
(uses)
3
ARM
4
Subcommand
azd deploy
(uses)
7
ACR & ARM
Executes
5
dotnet publish /t:PublishContainer
6
Executes
docker push

Conclusiones
y consideraciones



.NET Aspire

¡Gracias!



.NET Aspire

Resources

aka.ms/azuresql

aka.ms/dab

aka.ms/sqlai
aka.ms/sqlaisamples

aka.ms/sqldev

aka.ms/azuresqlblog

aka.ms/azuresqldevblog

🚨 Attention developers 🚨

Join us in shaping the future of #AzureSQLDatabase! 

We want to hear from you 👨‍💻👨‍💻. Share your insights and help us tailor Azure SQL to meet your needs, ensuring scalability, performance, and top-notch security.

https://aka.ms/datadevlist

Microsoft

Thank you!


TECH
COMMUNITY