EF Core 6 Migrations and Docker: Database Evolution and Containerization

Mateus · Follow
3 min read · Jan 8, 2024

Listen Share More

EF Core 6 Migrations and Docker, come together to orchestrate a harmonious symphony of database evolution and containerization. Today, let's dive into implementing migrations in a docker container.

EF Core 6 Migrations: The Database Maestro

EF Core 6 Migrations is the maestro of database evolution, conducting changes to database schema with finesse. It allows developers to manage database changes, keeping the schema aligned with the application's evolving needs. With EF Core Migrations, adding new tables, modifying columns, or altering relationships becomes a seamless part of the development process.

Key Elements of EF Core 6 Migrations:

1. Code-First Approach: Developers can define the database schema using code, and migrations are automatically generated based on changes in the code.
2. Version Control Integration: Migrations integrate seamlessly with version control systems, ensuring that the database schema aligns with the codebase across different stages of development.

3. Idempotent Scripts: Migrations generate idempotent SQL scripts, enabling easy deployment and rollback of database changes.

Docker: The Containerization Virtuoso

Docker provides a standardized, lightweight environment that encapsulates an application and its dependencies. Containers ensure consistency across different environments, from development to production, by bundling an application and its dependencies into a single unit.

Key Elements of Docker:

1. Isolation and Portability: Containers encapsulate applications and dependencies, ensuring they run consistently across diverse environments without worrying about differences in underlying infrastructure.
2. Efficiency and Resource Optimization: Docker containers share the host OS's kernel, making them lightweight and efficient. They consume fewer resources compared to traditional virtual machines.
3. Easy Deployment and Scaling: Docker simplifies deployment by packaging applications into containers, making it easy to deploy, scale, and manage applications in various environments.

The Synchronized Dance: EF Core Migrations and Docker

Now, imagine the symphony that unfolds when EF Core Migrations and Docker dance together. Here's how they complement each other:

1. Consistent Environments: Docker ensures a consistent development, testing, and production environment. EF Core Migrations, when encapsulated within a Docker container, guarantee that the database schema matches the application code in any environment.

2. Containerized Database Migrations: Developers can package their applications, including EF Core Migrations, into Docker containers. This containerized approach simplifies the deployment of both the application and its associated database changes.

3. Scalability and Versioned Deployments: Docker allows for easy scaling of applications, and when coupled with EF Core Migrations, developers can ensure versioned deployments, managing database changes seamlessly as they roll out updates.

Migrations in Docker

Our goal is not to guide you through configuring EF Core in a .NET solution (plenty of content is available on that). Instead, our focus is on the Docker side. Here, we'll outline three key files: `docker-compose.yaml` to define service containers, `Dockerfile` for building/running migrations, and optionally, a `Makefile` to streamline commands.

Dockerfile

The Dockerfile uses the default Microsoft .NET 6 SDK and Runtime images, enabling the build of our application. Additionally, it installs the `dotnet-ef` tool for executing migrations. Here, note that `src/Infrastructure` handles the EF Core packages and migrations, while `src/Api` exclusively handles startup configuration, defining DbContext, and Connection String.

```
ARG DOTNET_RUNTIME=mcr.microsoft.com/dotnet/aspnet:6.0
ARG DOTNET_SDK=mcr.microsoft.com/dotnet/sdk:6.0

FROM ${DOTNET_RUNTIME} AS base
ENV ASPNETCORE_URLS=http://+:7105
WORKDIR /home/app
EXPOSE 7105
```

```
COPY ["SolutionService.sln", "./"]
COPY ["src/Api/Api.csproj", "src/Api/Api.csproj"]
COPY ["src/Domain/Domain.csproj", "src/Domain/Domain.csproj"]
COPY ["src/Infrastructure/Infrastructure.csproj", "src/Infrastructure/Infrastru

COPY ["tests/Unit/Unit.csproj", "tests/Unit/Unit.csproj"]

RUN dotnet restore SolutionService.sln

COPY ["src/", "src/"]
COPY ["tests/", "tests/"]

## Run migrations
FROM buildbase as migrations
RUN dotnet tool install --version 6.0.9 --global dotnet-ef
ENV PATH="$PATH:/root/.dotnet/tools"
ENTRYPOINT dotnet-ef database update --project src/Infrastructure/ --startup-pr
```

Docker-Compose

In the `docker-compose.yaml`, we define two services: the database (SQL Server) and the migrations. The migrations service waits for the SQL Server, executes the steps outlined in the Dockerfile, and initiates a container to run migrations, facilitating the creation/update of the SQL Server database.

```
version: '3.5'
services:
  migrations:
    container_name: service-migrations
    image: service-migrations
    build:
      context: .
      dockerfile: Dockerfile
      target: migrations
    depends_on:
        - sqlserver

  sqlserver:
    container_name: sqlserver
    image: mcr.microsoft.com/mssql/server:2022-latest
    ports:
      - 1433:1433
    environment:
      - ACCEPT_EULA=Y
      - SA_PASSWORD=Password_123
```

Now we can spin up the SQL Server container and run the migrations

```
docker-compose up --abort-on-container-exit --exit-code-from migrations migrati
```

Makefile (Optional)

Rather than relying on the `docker-compose` command directly, we can encapsulate it within a Makefile, allowing us to execute `make` commands.

```
.PHONY: infra-local

infra-local:
 @docker-compose up -d sqlserver sqlserver
 @docker-compose up --abort-on-container-exit --exit-code-from migrations migra
```

Executing `make infra-local` now effortlessly initiates the SQL Server and seamlessly executes the migrations.

Docker Ef Core Dotnet Core Csharp Dotnet

Follow

Written by Mateus

31 Followers · 2 Following

Random stuff about development world