Aqui está um arquivo docker-compose.yml completo para subir uma aplicação ASP.NET com PostgreSQL, incluindo também o PGAdmin para gestão do banco de dados:

version: '3.8'

services:

webapp:

image: mcr.microsoft.com/dotnet/aspnet:8.0

container\_name: aspnet-app

restart: always

ports:

- "5000:80"

volumes:

- ./:/app

working\_dir: /app

environment:

- ASPNETCORE\_ENVIRONMENT=Development

- ConnectionStrings\_\_DefaultConnection=Server=db;Database=Projetos;User Id=admin;Password=admin;

depends\_on:

- db

db:

image: postgres:15-alpine

container\_name: postgres-db

restart: always

environment:

- POSTGRES\_USER=admin

- POSTGRES\_PASSWORD=admin

- POSTGRES\_DB=Projetos

volumes:

- postgres\_data:/var/lib/postgresql/data

ports:

- "5432:5432"

pgadmin:

image: dpage/pgadmin4

container\_name: pgadmin

restart: always

environment:

- PGADMIN\_DEFAULT\_EMAIL=admin@admin.com

- PGADMIN\_DEFAULT\_PASSWORD=admin

ports:

- "5050:80"

depends\_on:

- db

volumes:

postgres\_data:

Como usar:

1. Salve este conteúdo em um arquivo docker-compose.yml na raiz do seu projeto
2. Ajuste as seguintes configurações conforme necessário:
   * Portas mapeadas
   * Credenciais (substitua por valores seguros em produção)
   * Nomes dos containers
   * Variáveis de ambiente da aplicação ASP.NET

Pré-requisitos:

* Docker e Docker Compose instalados
* Seu projeto ASP.NET deve ter um Dockerfile na raiz

Comandos para executar:

bash

docker-compose build *# Construir as imagens*

docker-compose up -d *# Iniciar os containers em background*

Acessos após subir:

* Aplicação ASP.NET: [http://localhost:5000](http://localhost:5000/)
* PGAdmin (interface para PostgreSQL): [http://localhost:5050](http://localhost:5050/)
* Banco de dados PostgreSQL: acessível via porta 5432

Observação: Para ambientes de produção, você deve:

1. Usar imagens otimizadas (multi-stage build)
2. Configurar variáveis de ambiente de forma segura
3. Implementar health checks
4. Configurar volumes apropriados para persistência de dados