<u>deploy.yml</u>

Purpose: automatically builds and deploys my ASP.NET Core API to Azure whenever I push to the main branch

Trigger (Automatic):

```
on:
push:
branches:
- main
```

Setup .NET 8 SDK:

```
- name: Setup .NET
uses: actions/setup-dotnet@v3
with:
dotnet-version: '8.0.x'
```

Deploy to Azure Web App:

```
- name: Deploy to Azure Web App
uses: azure/webapps-deploy@v2
with:
app-name: customer-api-app-jeet123
slot-name: production
publish-profile: ${{ secrets.AZURE_WEBAPP_PUBLISH_PROFILE }}
package: CustomerApi/publish
```

Program.cs

App Setup & Database Configuration - Sets up the app and connects to SQL Server using EF Core:

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

// Add DbContext with SQL Server connection
builder.Services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultCon
nection")));
```

Swagger - Enables Swagger UI at /swagger for interactive API documentation:

```
app.UseSwagger();
app.UseSwaggerUI(c =>
{
c.SwaggerEndpoint("/swagger/v1/swagger.json", "Customer API V1");
c.RoutePrefix = "swagger"; // Swagger UI is at /swagger now
});
```

CRUD Endpoints for /customers - Full Create, Read, Update, and Delete functionality using EF Core:

```
// CRUD endpoints for Customers using EF Core

app.MapGet("/customers", async (AppDbContext db) =>
await db.Customers.ToListAsync());

app.MapGet("/customers/{id}", async (int id, AppDbContext db) =>
{
var customer = await db.Customers.FindAsync(id);
return customer is not null ? Results.Ok(customer) : Results.NotFound();
});

app.MapPost("/customers", async (Customer newCustomer, AppDbContext db) =>
{
db.Customers.Add(newCustomer);
await db.SaveChangesAsync();
return Results.Created($"/customers/{newCustomer.Id}", newCustomer);
});

app.MapPut("/customers/{id}", async (int id, Customer updatedCustomer, AppDbContext
db) =>
{
var existing = await db.Customers.FindAsync(id);
if (existing is null) return Results.NotFound();

existing.Name = updatedCustomer.Name;
existing.Email = updatedCustomer.Email;
existing.Phone = updatedCustomer.Phone;
await db.SaveChangesAsync();

return Results.Ok(existing);
});

app.MapDelete("/customers/{id}", async (int id, AppDbContext db) =>
{
var customer = await db.Customers.FindAsync(id);
if (customer is null) return Results.NotFound();

db.Customers.Remove(customer);
await db.SaveChangesAsync();

return Results.NoContent();
});
```

AppDbContext.cs

- This is the link between my C# models and the actual SQL database — it's how my API performs CRUD operations on the Customer table

```
using Microsoft.EntityFrameworkCore;
using CustomerApi.Models;

namespace CustomerApi.Data
{
public class AppDbContext : DbContext
{
public AppDbContext(DbContextOptions<AppDbContext> options)
: base(options) { }

public DbSet<Customer> Customers { get; set; }
}
}
```

Customer.cs

- This is the Customer model — it defines the structure of a customer record in my database
- Id is the primary key, and Name, Email, and Phone are the main fields for each customer
- This class maps directly to the Customers table in my SQL database through EF Core

```
namespace CustomerApi.Models
{
public class Customer
{
public int Id { get; set; } // e.g. 1
public string Name { get; set; } = ""; // e.g. "Jeet Patel"
public string Email { get; set; } = ""; // e.g. "jeet@example.com"
public string Phone { get; set; } = ""; // e.g. "jeet@example.com"
}
}
```