# M12 MATERI 1

1. **TUJUAN**

   CPMK : Mahasiswa dapat mengintegrasikan Frontend React JS dengan Backend ASP.NET Core dan Database Azure SQL.

   Sub-CPMK :

   a. Mahasiswa dapat membuat build production dari aplikasi React.
   b. Mahasiswa dapat mengonfigurasi ASP.NET Core.
   c. Mahasiswa dapat melakukan migrasi database menggunakan Entity Framework Core.

2. **DURASI WAKTU**

   2 pertemuan x 4 jam

3. **DASAR TEORI**

   React JS, Single Page Application (SPA), Entity Framework Core.

## 4. Percobaan

- **Membuat Frontend App**

1. Buatlah folder baru dengan nama **PRG4_M12_P1_XXX**
2. Lalu pada Visual Studio Code, buka terminal (Command Prompt/PowerShell) lalu jalankan perintah berikut untuk membuat proyek React baru:

```
npx create-react-app frontend-app
```

3. Tunggu instalasi selesai hingga keluar output “Happy Hacking”:

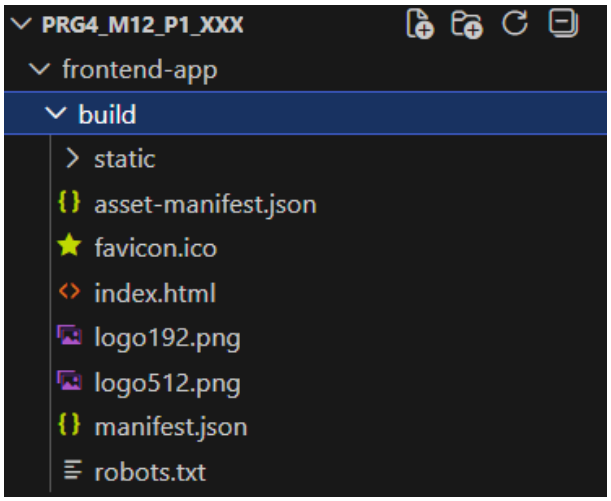
```
We suggest that you begin by typing:

  cd frontend-app
  npm start

Happy hacking!
```

4. Buka kembali terminal anda, lalu jalankan perintah berikut untuk pindah folder dan menjalankan proses build untuk mengubah kode React menjadi static file dan siap deploy:

```
\PRG4_M12_P1_XXX> cd frontend-app
```

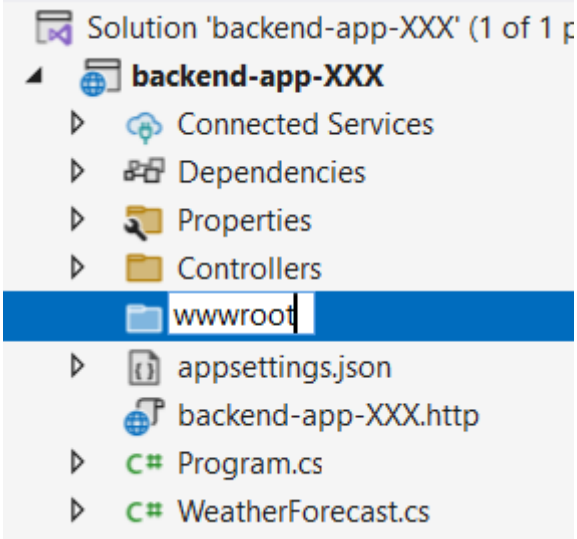
```
PRG4_M12_P1_XXX\frontend-app> npm run build
```

5. Hasil build akan muncul di dalam folder bernama build:

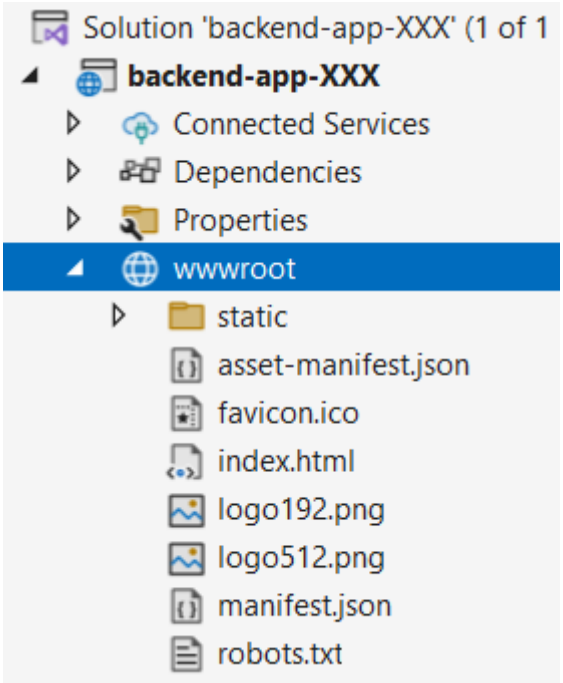


- **Membuat Backend ASP.NET Core Web API**

1. Buatlah project ASP. NET Core Web API baru dengan nama **backend-app-XXX**
2. Buat folder baru bernama wwwroot di dalam project backend:



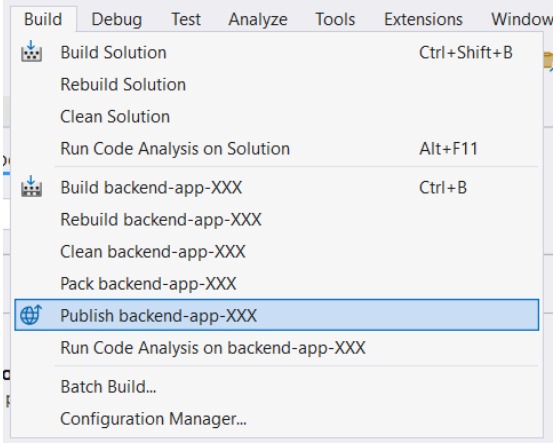

3. Lalu pindahkan seluruh hasil build yang ada di dalam folder frontend-app/build/* ke dalam folder wwwroot:



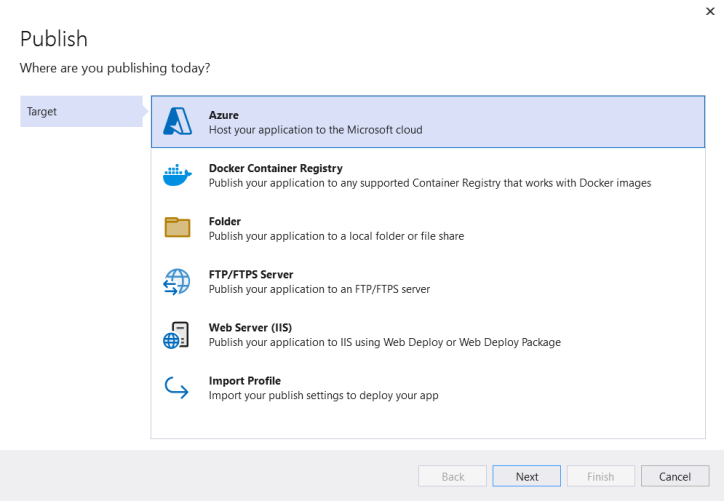

4. Modifikasi file Program.cs untuk mengizinkan akses static file. Tambahkan kode berikut sebelum app.Run():

```
19      app.UseHttpsRedirection();
20
21      app.UseAuthorization();
22
23      app.UseStaticFiles();
24      app.UseRouting();
25      app.MapControllers();
26      app.MapFallbackToFile("index.html");
27
28      app.Run();
29
```

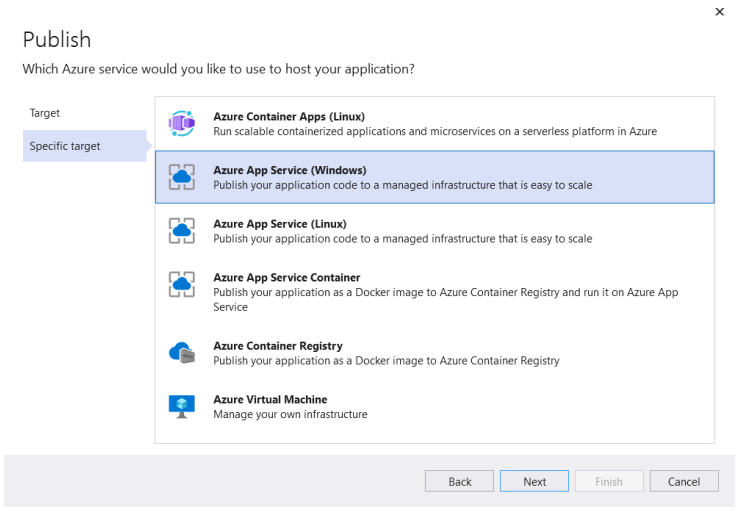
5. Jalankan menu **Build > Publish backend-app-xxx:**



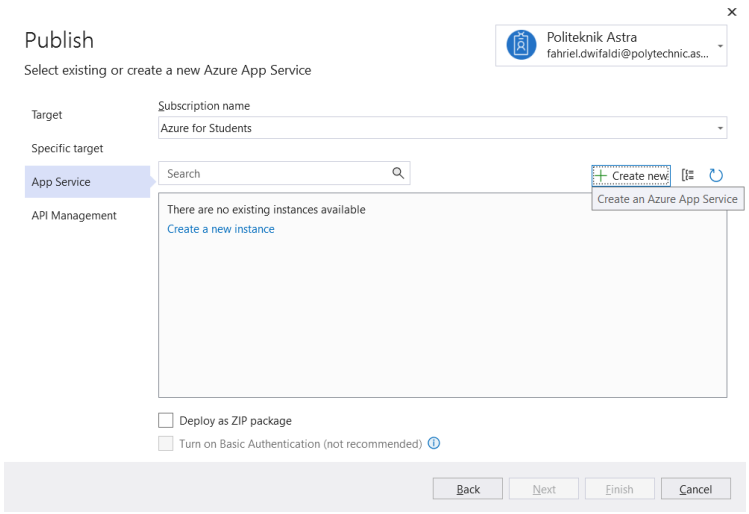

6. Pada pilihan **Target** publish pilih opsi **Azure:**



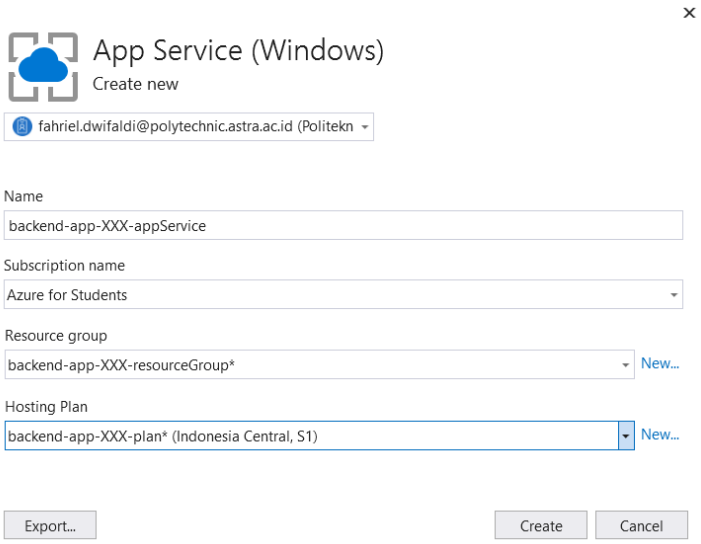

7. Lalu pada pilihan **Specific Target,** pilih opsi **Azure App Service (Windows):**



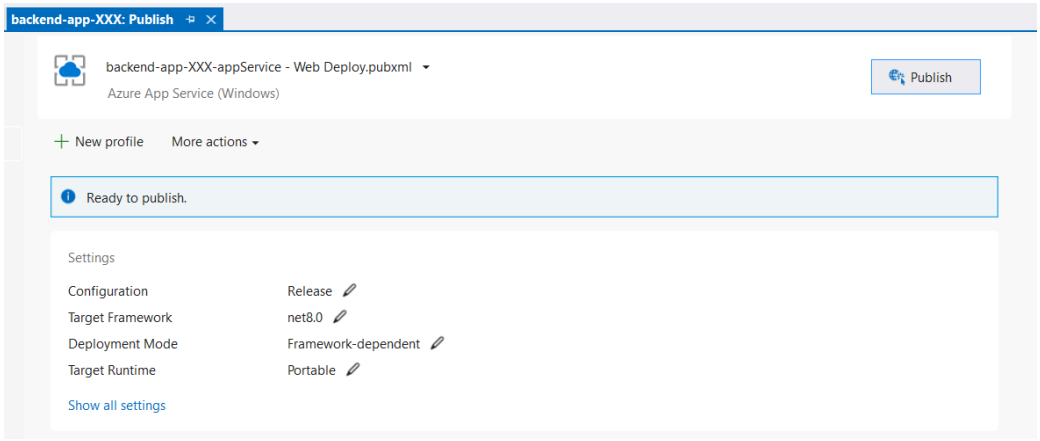

8. **Create New** pada pilihan App Service:

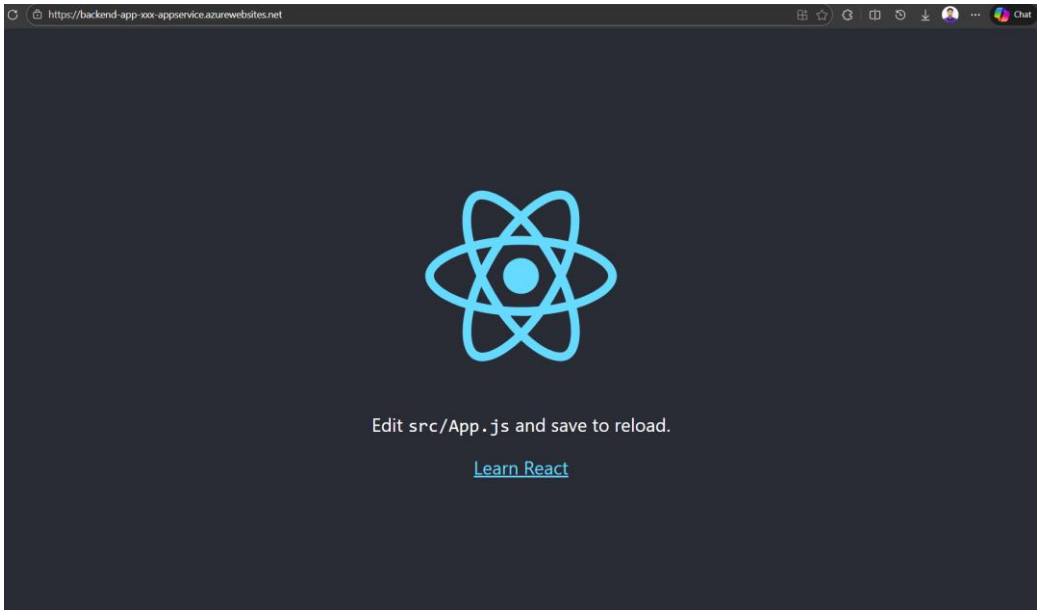



9. Sesuaikan nama dari **App Service, Resource Group**, dan **Hosting Plan**. Pastikan lokasi pada hosting plan diubah terlebih dahulu ke **Indonesia Central**:



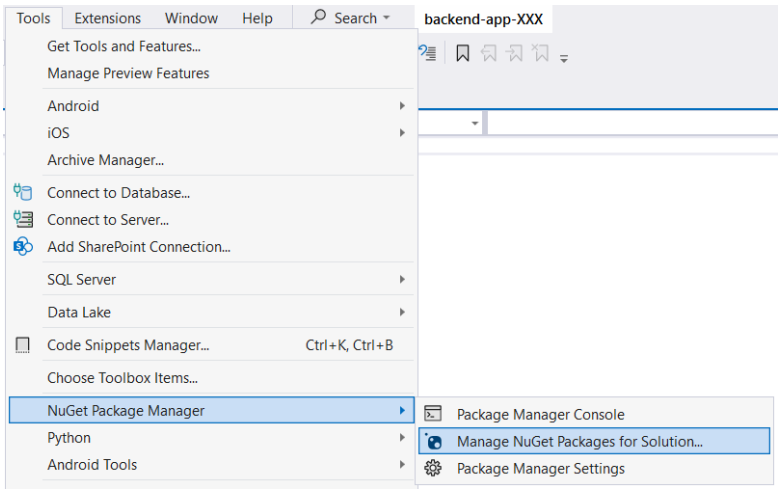
10. Pilih checkbox **Skip This Step** pada tahap **API Management** lalu tekan Finish dan tunggu hingga proses **Publish profile creation progress** selesai.
11. Tekan Publish, pastikan project sudah berhasil terdeploy menggunakan Azure:



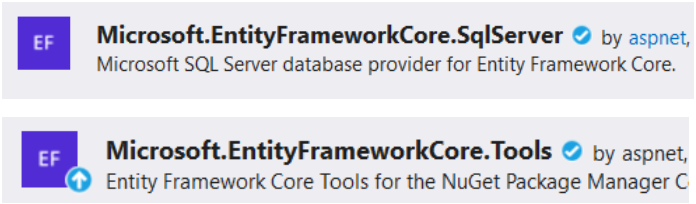



- **Integrasi Menggunakan Azure SQL Database**

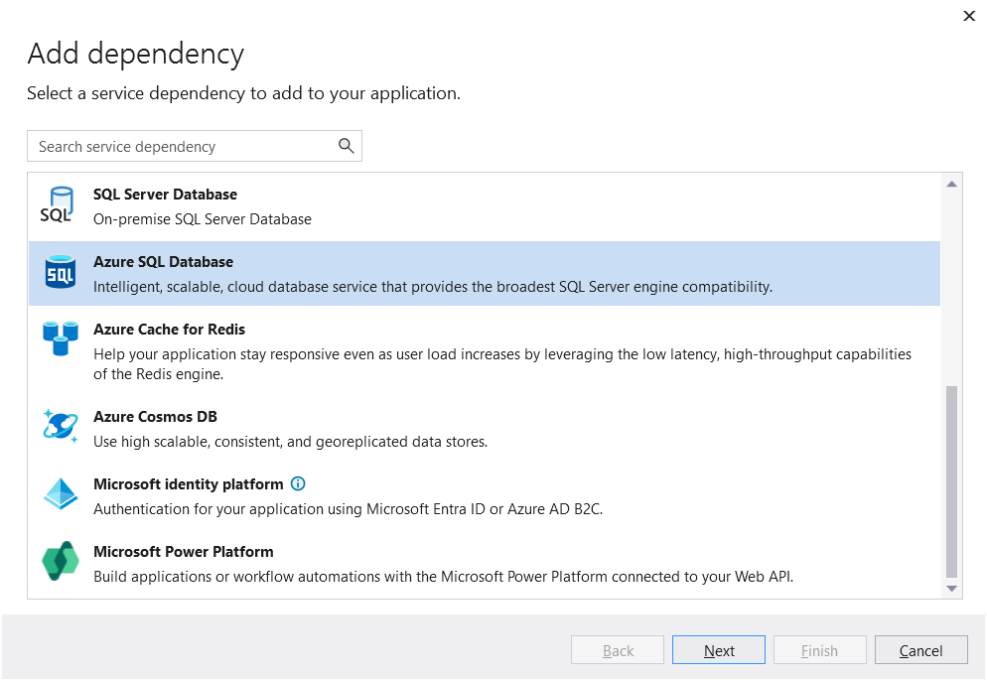
1. Buka menu **Tools > NuGet Package Manager > Manage NuGet Packages for Solution:**



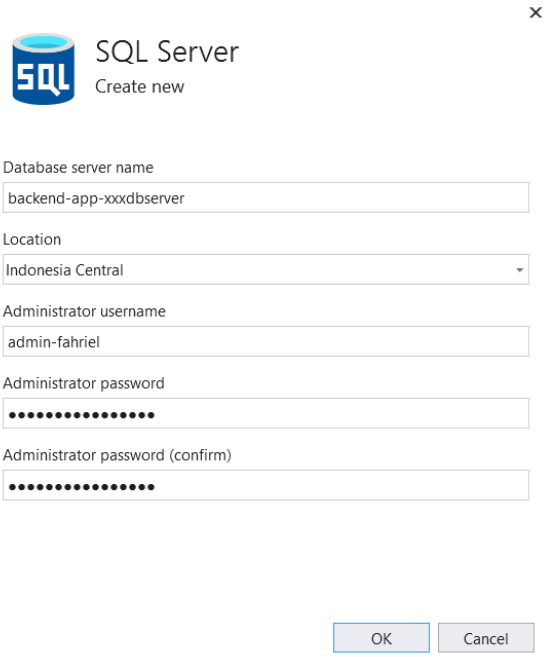

2. Lalu install 2 packages EntityFrameworkCore SqlServer dan Tools:



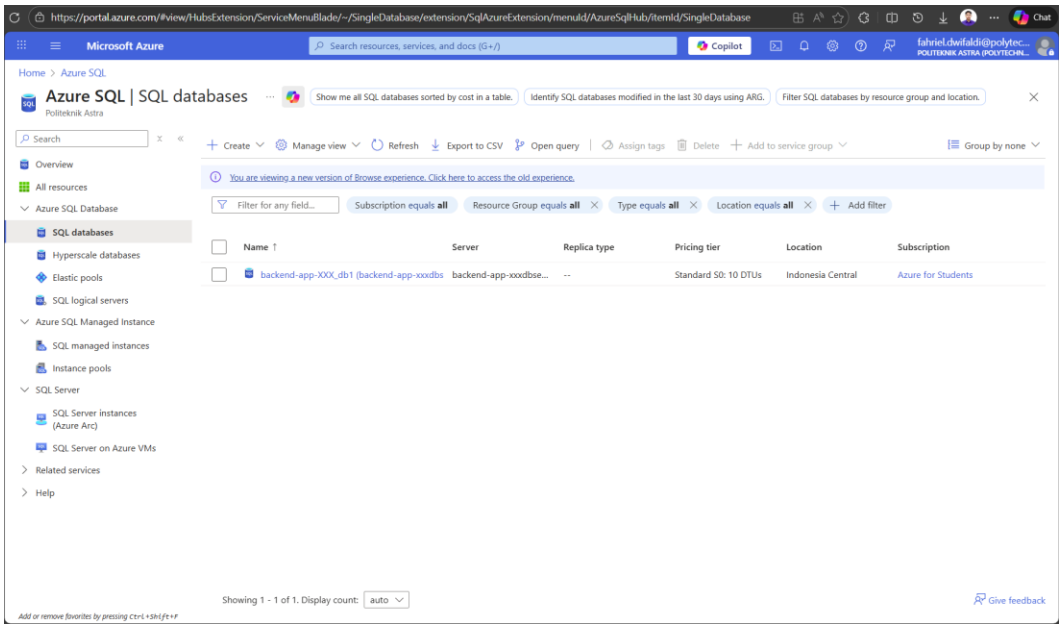

3. Buka tampilan menu **Publish** yang ada pada langkah sebelumnya, tambahkan **Service Dependencies > SQL Server Database:**



4. **Create New > New Database Server,** lalu masukkan username dan password dari database administrator:




5. Pastikan Azure SQL Database berhasil dibuat dengan cara membuka website **Portal Azure > Azure SQL Database**:




6. Buka Azure SQL Database yang baru saja dibuat, lalu pada menu **Settings > Connection Strings** copy value dari ADO.NET (SQL Authentication).
7. Buka file appsettings.json dan tambahkan konfigurasi koneksi database dengan isi DefaultConnection dengan value yang sudah diambil di tahap 6:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Server=tcp:SERVER_ANDA.database.windows.net,1433;Initial Catalog=NAMA_DB;User ID=USER;Password=PASS;Encrypt
  }
}
```

8. Buat model baru dan beri nama Product.cs

```
namespace backend_app
{
    4 references
    public class Product
    {
        1 reference
        public int Id { get; set; }

        0 references
        public string Name { get; set; } = string.Empty;

        [Column(TypeName = "decimal(18, 2)")]
        0 references
        public decimal Price { get; set; }
    }
}
```

9. Tambahkan controller baru dan beri nama ProductsController.cs

```
namespace backend_app.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    1 reference
    public class ProductsController : ControllerBase
    {
        private readonly AppDbContext _context;

        0 references
        public ProductsController(AppDbContext context)
        {
            _context = context;
        }

        [HttpGet]
        1 reference
        public async Task<ActionResult<IEnumerable<Product>>> GetProducts()
        {
            return await _context.Products.ToListAsync();
        }

        [HttpPost]
        0 references
        public async Task<ActionResult<Product>> PostProduct(Product product)
        {
            _context.Products.Add(product);
            await _context.SaveChangesAsync();
            return CreatedAtAction(nameof(GetProducts), new { id = product.Id }, product);
        }
```

```
        [HttpDelete("{id}")]
        0 references
        public async Task<IActionResult> DeleteProduct(int id)
        {
            var product = await _context.Products.FindAsync(id);
            if (product == null) return NotFound();

            _context.Products.Remove(product);
            await _context.SaveChangesAsync();

            return NoContent();
        }
    }
}
```

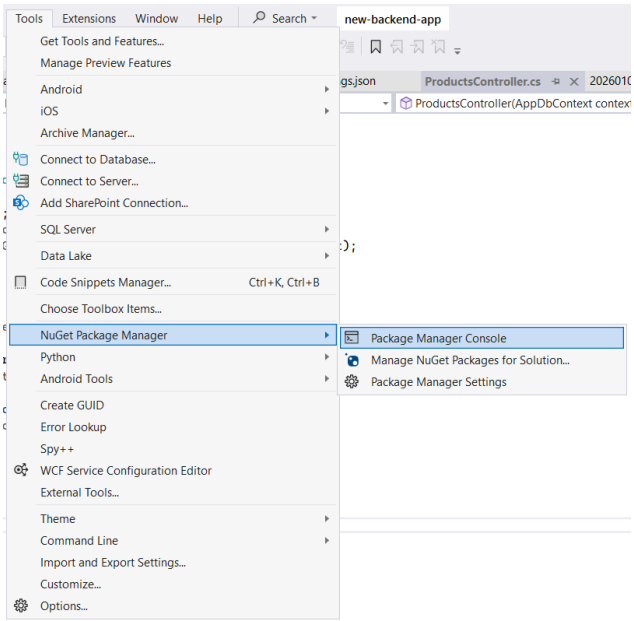
10. Buat file AppDbContext.cs baru dan tulis ulang kode berikut:

```
namespace backend_app
{
    7 references
    public class AppDbContext : DbContext
    {
        0 references
        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)
        {
        }

        4 references
        public DbSet<Product> Products { get; set; }
    }
}
```

11. Buka file Program.cs. Tambahkan kode berikut sebelum baris var app = builder.Build():

```
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<AppDbContext>(
    options => options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

var app = builder.Build();
```

12. Akses menu **Tools > NuGet Package Manager > Package Manager Console**:



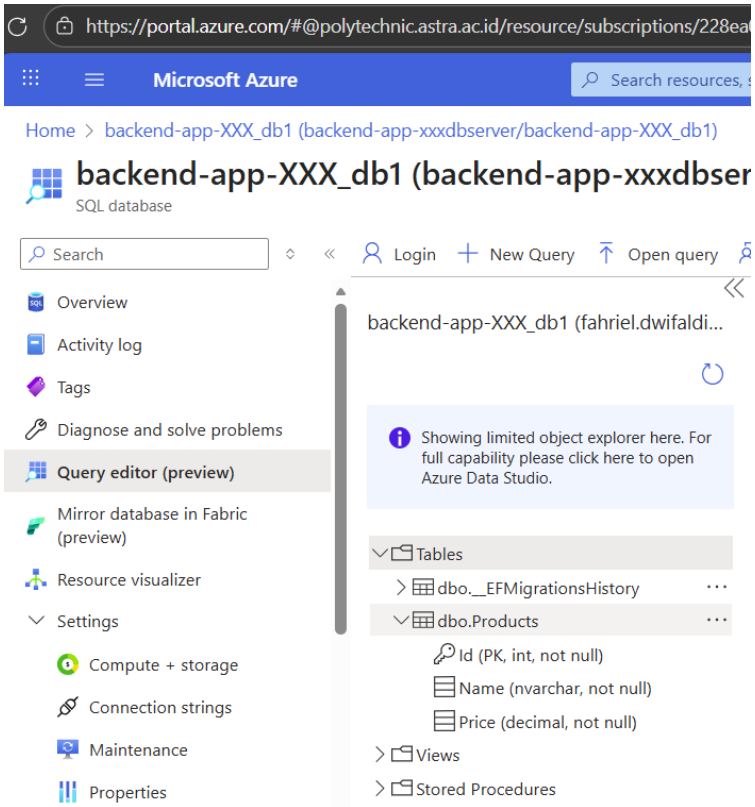
13. Tambahkan migration baru dengan nama InitialMigration untuk membuat tabel di database:

```
PM> Add-Migration InitialCreate
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
```

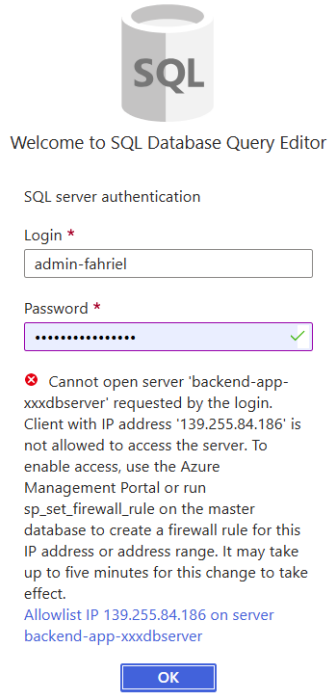
14. Apabila sudah berhasil, jalankan perintah **Update-Database**:

```
PM> Update-Database
Build started...
Build succeeded.
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (31ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT 1
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (22ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT OBJECT_ID(N'[__EFMigrationsHistory]');
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (10ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT 1
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (20ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT OBJECT_ID(N'[__EFMigrationsHistory]');
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (11ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT [MigrationId], [ProductVersion]
      FROM [__EFMigrationsHistory]
      ORDER BY [MigrationId];
Microsoft.EntityFrameworkCore.Migrations[20402]
      Applying migration '20260108002602_InitialCreate'.
Applying migration '20260108002602_InitialCreate'.
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (10ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      INSERT INTO [__EFMigrationsHistory] ([MigrationId], [ProductVersion])
      VALUES (N'20260108002602_InitialCreate', N'8.0.10');
Done.
```

15. Cek kembali pada **Portal Azure > Azure SQL Database > SQL Databases > { Nama_database } > Query Editor:**



16. Apabila terjadi masalah ketika login, klik bagian **Allowlist ...**




17. Buka kembali file App.js yang ada di frontend, dan update kode nya seperti pada gambar berikut:

frontend-app > src > App.js > App

```
import React, { useState, useEffect } from 'react';
import './App.css';

function App() {
  const [products, setProducts] = useState([]);
  const [name, setName] = useState("");
  const [price, setPrice] = useState("");

  useEffect(() => {
    fetchProducts();
  }, []);

  const fetchProducts = async () => {
    const response = await fetch('/api/products');
    const data = await response.json();
    setProducts(data);
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    if (!name || !price) return;

    const newProduct = { name: name, price: parseFloat(price) };

    await fetch('/api/products', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(newProduct)
    });

    setName("");
    setPrice("");
    fetchProducts();
  };

  const handleDelete = async (id) => {
    await fetch(`/api/products/${id}`, {
      method: 'DELETE'
    });
    fetchProducts();
  };
```

```jsx
  return (
    <div className="App">
      <header className="App-header">
        <h1>Manajemen Produk</h1>

        <form onSubmit={handleSubmit} style={{ marginBottom: '20px' }}>
          <input
            type="text"
            placeholder="Nama Produk"
            value={name}
            onChange={(e) => setName(e.target.value)}
            style={{ padding: '10px', marginRight: '10px' }}
          />
          <input
            type="number"
            placeholder="Harga"
            value={price}
            onChange={(e) => setPrice(e.target.value)}
            style={{ padding: '10px', marginRight: '10px' }}
          />
          <button type="submit" style={{ padding: '10px 20px', cursor: 'pointer' }}>
            Tambah
          </button>
        </form>

        <table className='table' style={{ width: '50%', margin: '0 auto', color: 'black', background: 'white' }}>
          <thead>
            <tr style={{ background: '#007bff', color: 'white' }}>
              <th style={{ padding: '10px' }}>Nama</th>
              <th style={{ padding: '10px' }}>Harga</th>
              <th style={{ padding: '10px' }}>Aksi</th>
            </tr>
          </thead>
          <tbody>
            {products.map((product) => (
              <tr key={product.id} style={{ borderBottom: '1px solid #ccc' }}>
                <td style={{ padding: '10px' }}>{product.name}</td>
                <td style={{ padding: '10px' }}>{product.price}</td>
                <td style={{ padding: '10px' }}>
                  <button
                    onClick={() => handleDelete(product.id)}
                    style={{ background: 'red', color: 'white', border: 'none', padding: '5px 10px', cursor: 'pointer' }}
                  >
                    Hapus
                  </button>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      </header>
    </div>
  );
}

export default App;
```

18. Buka kembali file App.js yang ada di frontend, dan update kode nya seperti pada gambar berikut:

frontend-app > src > # App.css > th

```css
.App {
  text-align: center;
  font-family: Arial, sans-serif;
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

table {
  border-collapse: collapse;
  width: 80%;
  margin-top: 20px;
  color: black;
  background-color: white;
  border-radius: 8px;
  overflow: hidden;
}

th, td {
  padding: 12px;
  text-align: left;
  border-bottom: 1px solid #ddd;
}

th {
  background-color: #007bff;
  color: white;
}
```

19. Pastikan ketika sudah dipublish ulang, kode sudah terupdate dan berhasil menyimpan ke database: