

# How to create a .NET8 CRUD WebAPI Azure CosmosDB for PostgreSQL MicroService

---

The source code for this example can be found in this github repo: [https://github.com/luisccoco/MicroServices\\_dotNET8\\_CRUD\\_WebAPI-AzureCosmosDB-for-PostgreSQL](https://github.com/luisccoco/MicroServices_dotNET8_CRUD_WebAPI-AzureCosmosDB-for-PostgreSQL)

## 0. Prerequisites

---

- Install Visual Studio 2022 Community Edition
- Install .NET SDK 8.0.1
- Install Entity Framework Core tools reference - .NET Core CLI: <https://learn.microsoft.com/en-us/ef/core/cli/dotnet>

**dotnet ef** can be installed as either a global or local tool. Most developers prefer installing dotnet ef as a global tool using the following command:

```
dotnet tool install --global dotnet-ef
```

Update the tool using the following command:

```
dotnet tool update --global dotnet-ef
```

Before you can use the tools on a specific project, you'll need to add this package to your application

```
dotnet add package Microsoft.EntityFrameworkCore.Design
```

Run the following commands to verify that EF Core CLI tools are correctly installed:

```
dotnet ef
```

## 1. Create .NET8 CRUD WebAPI for PostgreSQL

---

**Step 1:** Create a New .NET Web API Project. Open a command line interface (CLI).

Run the following command to create a new Web API project:

```
dotnet new webapi -n BookApiProject
```

Navigate into your project directory:

```
cd BookApiProject
```

**Step 2:** Add Required Packages. Add the necessary NuGet packages:

```
dotnet add package Microsoft.EntityFrameworkCore  
dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL  
dotnet add package Swashbuckle.AspNetCore
```

**Step 3:** Input the Program.cs source code:

```
using Microsoft.AspNetCore.Mvc;  
using Microsoft.EntityFrameworkCore;  
using Microsoft.OpenApi.Models;  
using Swashbuckle.AspNetCore.Swagger;
```

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

// Configure your DbContext
var connectionString = builder.Configuration.GetConnectionString("MyPostgresDb");
builder.Services.AddDbContext<BookDbContext>(options =>
    options.UseNpgsql(connectionString));

// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();

// Add Swagger support
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "Book API", Version = "v1" });
});

var app = builder.Build();

// Initialize the database
using (var scope = app.Services.CreateScope())
{
    var services = scope.ServiceProvider;
    var dbContext = services.GetRequiredService<BookDbContext>();
    dbContext.Database.Migrate(); // This applies pending migrations or creates the database if it doesn't exist
}

// Configure Swagger middleware
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "Book API v1"));
}

// Map CRUD operations for Books
app.MapGet("/books", async (BookDbContext db) => await db.Books.ToListAsync());
app.MapGet("/books/{id}", async (int id, BookDbContext db) => await db.Books.FindAsync(id) is Book book ? Results.Ok(book) : Resu
```

```
app.MapPost("/books", async (Book book, BookDbContext db) =>
{
    db.Books.Add(book);
    await db.SaveChangesAsync();

    return Results.Created($"/books/{book.BookId}", book);
});
app.MapPut("/books/{id}", async (int id, Book inputBook, BookDbContext db) =>
{
    var book = await db.Books.FindAsync(id);

    if (book == null) return Results.NotFound();

    book.BookName = inputBook.BookName;

    await db.SaveChangesAsync();
    return Results.NoContent();
});
app.MapDelete("/books/{id}", async (int id, BookDbContext db) =>
{
    var book = await db.Books.FindAsync(id);

    if (book == null) return Results.NotFound();

    db.Books.Remove(book);
    await db.SaveChangesAsync();
    return Results.NoContent();
});

app.Run();

// Define your DbContext and Book entity
public class BookDbContext : DbContext
{
    public BookDbContext(DbContextOptions<BookDbContext> options)
        : base(options) { }
```

```
public DbSet<Book> Books { get; set; }  
  
public class Book  
{  
    public int BookId { get; set; }  
    public string BookName { get; set; }  
}
```

#### Step 4: Modify appsettings.json and set your database connection string

```
{  
  "Logging": {  
    "LogLevel": {  
      "Default": "Information",  
      "Microsoft.AspNetCore": "Warning"  
    }  
  },  
  "AllowedHosts": "*",  
  "ConnectionStrings": {  
    "MyPostgresDb": "Server=c-mypostgresql.znj364bc3mdfyx.postgres.cosmos.azure.com;Database=citus;Port=5432;User Id=citus;Passwo  
  }  
}
```

#### Step 5: Run Migrations (if using Code First). Run the following command to add a migration:

```
dotnet ef migrations add InitialCreate
```

Apply the migration to your database:

```
dotnet ef database update
```

## Step 6: Run Your Application

```
dotnet run
```

Access the Swagger UI by going to <http://localhost:5000/swagger> in your web browser.

## 2. Create Azure CosmosDB for PostgreSQL with Azure Portal

---

Sign in to Azure Portal:

Go to Azure Portal and sign in with your Azure account.

Create a New Resource:

Microsoft Azure

Search resources, services, and docs (G+/)

Home >

## Create a resource

Get Started

Recently created

### Categories

- AI + Machine Learning
- Analytics
- Blockchain
- Compute
- Containers
- Databases**
- Developer Tools

Search services and marketplace

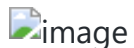
Popular Azure services [See more in All services](#)

- SQL Database**  
[Create](#) | [Docs](#) | [MS Learn](#)
- Azure SQL**  
[Create](#) | [Docs](#) | [MS Learn](#)
- Azure Cosmos DB**  
[Create](#) | [Docs](#) | [MS Learn](#)
- Azure Synapse Analytics**  
[Create](#) | [Docs](#) | [MS Learn](#)

Left sidebar navigation:

- Create a resource
- Home
- Dashboard
- All services
- FAVORITES
- All resources
- Resource groups
- Quickstart Center
- App Services
- Function App
- SQL databases
- Azure Cosmos DB
- Virtual machines

Click on "Create a resource" in the top left corner of the dashboard.

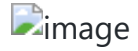


Search for Azure Cosmos DB:

In the "New" window, search for "Azure Cosmos DB" and select it from the results.

## Create Azure Cosmos DB Account:

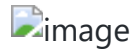
Click the "**Create**" button to start configuring your Azure Cosmos DB account.



Select the appropriate subscription and resource group (or create a new resource group).

Enter an account name.

Choose the API as "**Azure Cosmos DB for PostgreSQL**".



Input the new PostgreSQL values



[Home](#) > [Create a resource](#) > [Create an Azure Cosmos DB account](#) >

## Create an Azure Cosmos DB for PostgreSQL cluster

Microsoft

[Basics](#) [Networking](#) [Encryption](#) [Tags](#) [Review + create](#)

Create an Azure Cosmos DB for PostgreSQL cluster. Azure Cosmos DB for PostgreSQL cluster enables you to horizontally scale your Postgres relational database across multiple managed Postgres servers using sharding and replication. [Learn more](#)

### Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ

Subscription 1



Resource group \* ⓘ

myRG

[Create new](#)

### Cluster details

Cluster name \* ⓘ

mypostgresql

Location \* ⓘ

(Europe) West Europe

Scale \* ⓘ

**1 node, no high availability (HA)**

Burstable, 1 vCore(s) / 2 GiB RAM, 22.56 USD / month

[Configure](#)

Billing model



Azure Cosmos DB for PostgreSQL cluster currently only supports a vCore based billing model. [Learn more](#)

[Review + create](#)

&lt; Previous

Next : Networking &gt;

[+ Create a resource](#)[Home](#)[Dashboard](#)[All services](#)[FAVORITES](#)[All resources](#)[Resource groups](#)[Quickstart Center](#)[App Services](#)[Function App](#)[SQL databases](#)[Azure Cosmos DB](#)[Virtual machines](#)[Load balancers](#)[Storage accounts](#)[Virtual networks](#)[Microsoft Entra ID](#)[Monitor](#)[Advisor](#)[Microsoft Defender for Cloud](#)[Cost Management +](#)

+ Create a resource

Home

Dashboard

All services

FAVORITES

All resources

Resource groups

Quickstart Center

App Services

Function App

SQL databases

Azure Cosmos DB

Virtual machines

Load balancers

Storage accounts

Virtual networks

Microsoft Entra ID

Monitor

Advisor

Microsoft Defender for Cloud

Cost Management + Billing

[Home](#) > [Create a resource](#) > [Create an Azure Cosmos DB account](#) > [Create an Azure Cosmos DB for PostgreSQL cluster](#) >

## Create an Azure Cosmos DB for PostgreSQL cluster | Scale

Microsoft

As your performance needs grow, easily add nodes to seamlessly use all the power of distributed PostgreSQL on multiple nodes. Tune performance for your workload by selecting the optimal number of nodes as well as compute, memory, and storage configuration.



### Coordinator - Worker

This configuration features a single node with the ability to create distributed tables on that node. You can use this configuration with or without distributed tables. As your needs grow in the future, easily add worker nodes to your cluster. [Learn more](#)

#### Nodes

Node count *	Single node
Node compute *	Burstable, 1 vCore, 2 GiB RAM
Node storage *	32 GiB

#### Availability zones

Preferred availability zones *	No preference
--------------------------------	---------------

[Save](#)

### Cost summary



#### Node compute - Burstable

1 vCore / month (in USD)	18.18
--------------------------	-------

#### Coordinator node storage - General purpose

Cost per GiB / month (in USD)	0.14
Storage selected (in GiB)	X 32

#### Cluster subtotal

High availability	--
-------------------	----

#### ESTIMATED COST / MONTH

22.56	USD
-------	-----

#### Additional charge per usage

See [pricing page](#) for more detail

[Home](#) > [Create a resource](#) > [Create an Azure Cosmos DB account](#) >

## Create an Azure Cosmos DB for PostgreSQL cluster

Microsoft

Cluster name \* ⓘ

mypostgresql

Location \* ⓘ

(Europe) West Europe

Scale \* ⓘ

**1 node, no high availability (HA)**

Burstable, 1 vCore(s) / 2 GiB RAM, 22.56 USD / month

[Configure](#)

Billing model

ⓘ Azure Cosmos DB for PostgreSQL cluster currently only supports a vCore based billing model. [Learn more](#)

PostgreSQL version ⓘ

16

Database name ⓘ

citus

### Administrator account

Admin username \* ⓘ

citus

Password \* ⓘ

.....

Confirm password \*

.....

[Review + create](#)

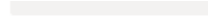
&lt; Previous

Next : Networking &gt;

[+ Create a resource](#)[Home](#)[Dashboard](#)[All services](#)

#### FAVORITES

[All resources](#)[Resource groups](#)[Quickstart Center](#)[App Services](#)[Function App](#)[SQL databases](#)[Azure Cosmos DB](#)[Virtual machines](#)[Load balancers](#)[Storage accounts](#)[Virtual networks](#)[Microsoft Entra ID](#)[Monitor](#)[Advisor](#)[Microsoft Defender for Cloud](#)[Cost Management +](#)



Review and create the account.

[Home](#) > [Create a resource](#) > [Create an Azure Cosmos DB account](#) >

## Create an Azure Cosmos DB for PostgreSQL cluster

Microsoft

provide rights for third-party offerings. For additional details see [Azure Marketplace Terms](#)

### Basics

Subscription	99888cc6-c635-4ebd-b0ac-1be1dace0089
Resource Group	myRG
Location	(Europe) West Europe
Cluster name	mypostgresql
Server admin login name	citus
Database name	citus
Worker node count	0
Worker node compute + storage	--
Coordinator node compute + storage	Burstable, 1 vCore(s), 2 GiB RAM, 32 GiB
Preferred availability zone	No preference
High availability	No
Backup	Zone-redundant

### Networking

Connectivity method	Public access (allowed IP addresses)
Allow public access from Azure services and resources within Azure to this cluster	No

Create a resource

Home

Dashboard

All services

FAVORITES

All resources

Resource groups

Quickstart Center

App Services

Function App

SQL databases

Azure Cosmos DB

Virtual machines

Load balancers

Storage accounts

Virtual networks

Microsoft Entra ID

Monitor

Advice



Microsoft Defender for Cloud

Cost Management + Billing

Firewall rules

0

Enable access to the worker nodes

Off

Create

< Previous

Next >

Download a template for automati

Once your account is created, go to it in the Azure portal.

Microsoft Azure

Search resources, services, and docs (G+)

luiscoenriquez@hotmail... PARTICULAR (LUISCOENRIQU...)

Home > CosmosDB-PG-mypostgresql-1825a5a480444e5c94bc9dcd41fbb815 | Overview

mypostgresql Azure Cosmos DB for PostgreSQL Cluster

Search

Delete Reset password Upgrade Stop Restore Restart Feedback

Overview

Activity log

Tags

Quick start (preview)

Settings

Scale

Networking

Connection strings

High availability

Coordinator node parameters

Maintenance

Locks

Cluster management

Authentication

Shard rebalancer

Replicate data globally

Data encryption

Monitoring

Essentials

Resource group : myRG

Location : West Europe

Subscription : Subscription 1

Subscription ID : 99888cc6-c635-4ebd-b0ac-1be1dace0089

PostgreSQL version : 16

Citus version : 12.1

Configuration : Single node configuration

Coordinator name : c-mypostgresql.x7wzviwo42ae6e.postgres.cosmos.azure.com

Database name : citus

Admin username : citus

Connectivity method : Public access (allowed IP addresses)

Coordinator node : Bursttable, 1 vCores / 2 GiB RAM, 32 GiB storage

Worker nodes : ---

Replication role : No replicas

High availability : No

Backup : Zone-redundant

Tags (edit) : Add tags

Node Monitoring Recommendations (0)

Looking for metrics? Click here

Name	Type	Status	High availability	Availability zone	Fully qualified domain name
mypostgresql-c	Coordinator	Available	No	2	c-mypostgresql.x7wzviwo42ae6e.postgres.cosmos.a...

Under "Quick start (preview)" create a new database and a container within that database.

The screenshot shows the Microsoft Azure portal interface. On the left is a sidebar with navigation options like 'Create a resource', 'Home', 'Dashboard', 'All services', and 'FAVORITES'. The main content area displays the 'mypostgresql | Quick start (preview)' page. At the top, there's a search bar and a link to 'Open PSQL shell'. A yellow banner at the top right contains a warning: 'The Network settings for this account are preventing access from Data Explorer. Please allow access from Azure Portal to proceed.' Below this, the 'Quick start (preview)' section is visible. The main heading is 'Welcome to Azure Cosmos DB for PostgreSQL', followed by the text 'Get started with our sample datasets, documentation, and additional tools.' Three cards are presented: 'Launch quick start' (highlighted with a red box), 'PostgreSQL Shell', and 'Connect with pgAdmin'.

### 3. Copy the Connection String in appsettings.json file

Home > CosmosDB-PG-mypostgresql-1825a5a480444e5c94bc9dcd41fbb815 | Overview > mypostgresql

### mypostgresql | Connection strings

Azure Cosmos DB for PostgreSQL Cluster

Public IP addresses on worker nodes: ☐ Off

Database name: citus

Show connection strings for: mypostgresql-c

PgBouncer connection strings: ☐

PostgreSQL connection URL: postgres://citus:(your\_password)@c-mypostgresql.x7wzviwo42ae6e.postgres.cosmos.azure.com:5432/citus?sslmode=require

psql: psql "host=c-mypostgresql.x7wzviwo42ae6e.postgres.cosmos.azure.com port=5432 dbname=citus user=citus password=(your\_password) sslmode=require"

JDBC: jdbc:postgresql://c-mypostgresql.x7wzviwo42ae6e.postgres.cosmos.azure.com:5432/citus?user=citus&password=(your\_password)&sslmode=require

Node.js, Python, Ruby, PHP, C++ (libpq): host=c-mypostgresql.x7wzviwo42ae6e.postgres.cosmos.azure.com port=5432 dbname=citus user=citus password=(your\_password) sslmode=require

ADO.NET: Server=c-mypostgresql.x7wzviwo42ae6e.postgres.cosmos.azure.com;Database=citus;Port=5432;User Id=citus;Password=(your\_password);Ssl Mode=Require;

Secure connections: Only secure connections are supported. For production use cases, we recommend using the 'verify-full' mode to enforce TLS certificate verification. You will need to download the Azure Cosmos DB for PostgreSQL certificate, and provide it when connecting to the database. [Learn more](#)

**IMPORTANT NOTE:** Do not forget to input your PASSWORD (Password=LuisCoco123456) in the connection string, by default is not set.

appsettings.json

```
{  
  "Logging": {  
    "LogLevel": {
```



```
"Default": "Information",
"Microsoft.AspNetCore": "Warning"
},
"AllowedHosts": "*",
"ConnectionStrings": {
  "MyPostgresDb": "Server=c-mypostgresql.x7wzviwo42ae6e.postgres.cosmos.azure.com;Database=citus;Port=5432;User Id=citus;Passwo
}
}
```


## 4. Add the FireWall rules



Search resources, services, and docs (G+)

luiscoenriquez@hotmail...  
PARTICULAR (LUISCOCOENRIQU...)

Home > CosmosDB-PG-mypostgresql-1825a5a480444e5c94bc9dcd41fbb815 | Overview > mypostgresql

 **mypostgresql** | Networking ☆ ...

Azure Cosmos DB for PostgreSQL Cluster

Search

Overview

Activity log

Tags

Quick start (preview)

Settings

Scale

**Networking**

Connection strings

High availability

Coordinator node parameters

Maintenance

Locks

Cluster management

Authentication

Shard rebalancer

Replicate data globally

Data encryption

Monitoring

Save Discard Feedback

This cluster enforces encrypted connections using Transport Layer Security (TLS). See documentation for the information on TLS configuration as well as certificate download and verification. [Learn More](#)

**Private access**

Create private endpoints to allow hosts in the selected virtual network to access nodes of this cluster.

+ Add private endpoint ✓ Approve ✗ Reject 🗑 Remove ↻ Refresh

Filter by name... All connection states

Private endpoint ↑↓	Connection state ↑↓	Virtual network / subnet	Connection name ↑↓	Description ↑↓
No results.				

**Public access**

Inbound connections from the public IP addresses specified below will be allowed to port 5432 (Postgres) and 6432 (PgBouncer) on the coordinator (mypostgresql-c) in this cluster (mypostgresql). If you enable public IP addresses on the worker nodes, all firewall rules will be effective for port 5432 on all worker nodes in this cluster. [Learn More](#)

☒ Allow public access from Azure services and resources within Azure to this cluster ⓘ

☐ Enable access to the worker nodes ⓘ

+ Add current client IP address ( 80.30.34.207 ) + Add 0.0.0.0 - 255.255.255.255

Firewall rule name	Start IP address	End IP address
Firewall rule name	Start IP address	End IP address

The screenshot shows the Azure portal interface for the 'mypostgresql' Azure Cosmos DB for PostgreSQL cluster. The left sidebar contains navigation options like 'Create a resource', 'Home', 'Dashboard', 'All services', and 'FAVORITES'. The main content area is titled 'mypostgresql | Networking' and includes a search bar, 'Save', 'Discard', and 'Feedback' buttons. The 'Networking' section is expanded in the left sidebar, showing options like 'Overview', 'Activity log', 'Tags', 'Quick start (preview)', 'Settings', 'Scale', 'Connection strings', 'High availability', 'Coordinator node parameters', 'Maintenance', 'Locks', 'Cluster management', 'Authentication', 'Shard rebalancer', 'Replicate data globally', 'Data encryption', and 'Monitoring'. The 'Encrypted connections' section states that the cluster enforces encrypted connections using Transport Layer Security (TLS). The 'Private access' section allows creating private endpoints to access nodes of the cluster. Below this, there are buttons for 'Add private endpoint', 'Approve', 'Reject', 'Remove', and 'Refresh'. A table shows the 'All connection states' with columns for 'Private endpoint', 'Connection state', 'Virtual network / subnet', 'Connection name', and 'Description'. The table currently shows 'No results.' The 'Public access' section allows enabling public access from Azure services and resources within Azure to the cluster. Below this, there are buttons for 'Add current client IP address (80.30.34.207)' and 'Add 0.0.0.0 - 255.255.255.255'. A table shows the 'Firewall rule name', 'Start IP address', and 'End IP address'. The table currently shows one rule: 'AllowAll\_2024-1-10\_8-33-38' with 'Start IP address' 0.0.0.0 and 'End IP address' 255.255.255.255.

## 5. Create Azure CosmosDB for PostgreSQL with Azure CLI

Ensure you have the Azure CLI installed and you're logged in. If not, you can download it from the Azure CLI website and log in using az login.

Create a Resource Group (if you don't already have one):

```
az group create --name YourResourceGroupName --location eastus
```

Create an Azure Cosmos DB Account:

```
az cosmosdb create --name YourCosmosDBAccountName --resource-group YourResourceGroupName --capabilities EnablePostgreSQL
```

Create a Database and Container:

Currently, creating databases and containers specifically for Azure Cosmos DB for PostgreSQL through the Azure CLI might not be supported directly.

You might need to use the Azure Portal or Cosmos DB SDK for this step.

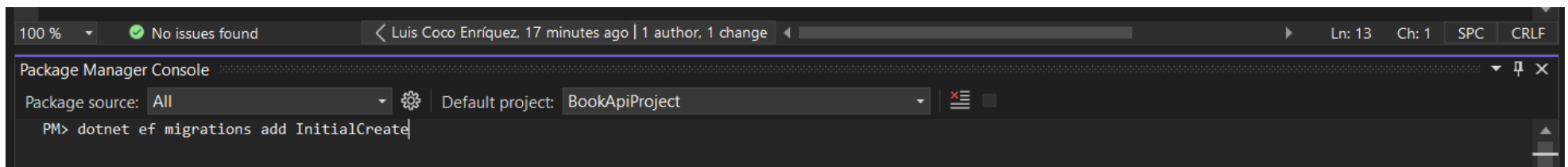
After setting up your Azure Cosmos DB for PostgreSQL, you'll need to retrieve the connection string to use in your .NET application.

You can find this in the Azure Portal under your Cosmos DB account's "Connection String" section.

## 6. Migrate the database

Run this command for creating the initial migration

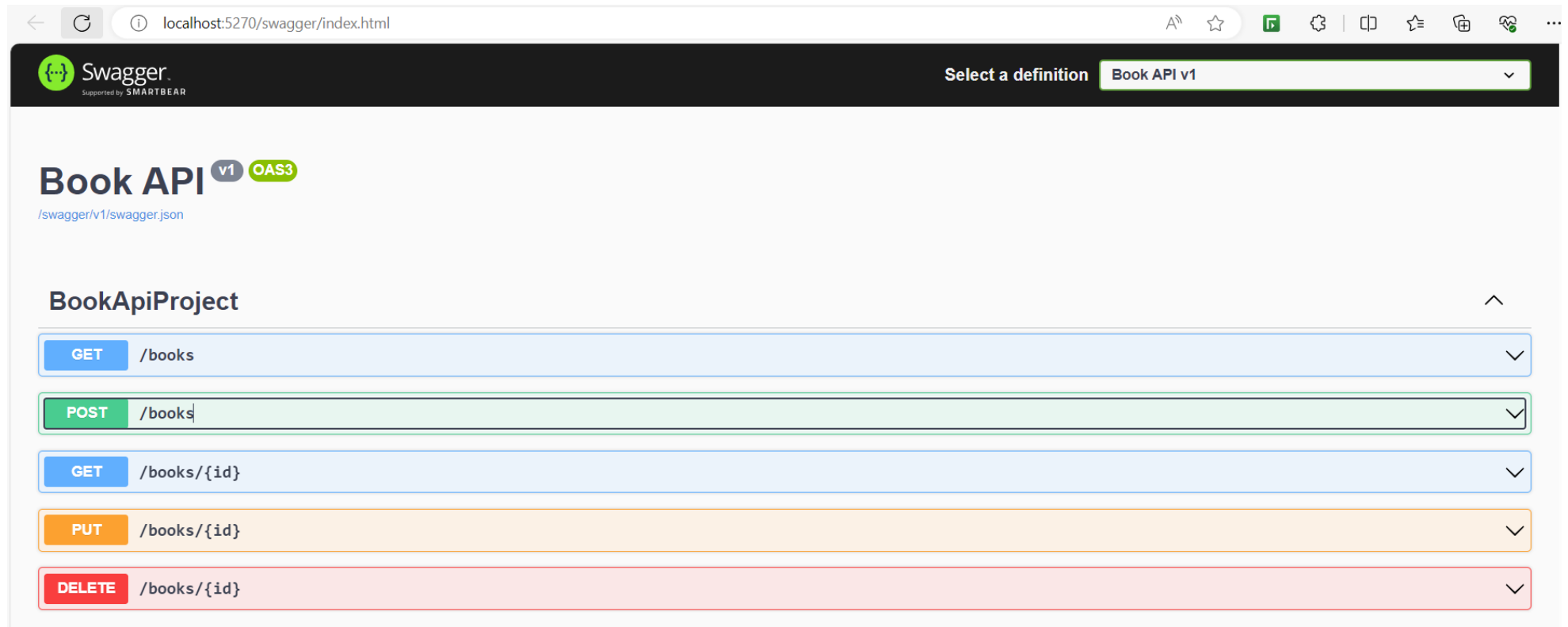
```
dotnet ef migrations add InitialCreate
```



Or this command for updating the migration

```
dotnet ef database update InitialCreate
```

## 7. Verify your application

<http://localhost:5270/swagger/index.html>

The image shows a web browser displaying the Swagger UI for a REST API. The browser's address bar shows the URL `localhost:5270/swagger/index.html`. The Swagger UI header includes the Swagger logo, the text "Supported by SMARTBEAR", and a dropdown menu labeled "Select a definition" with "Book API v1" selected. The main content area displays the API title "Book API" with version "v1" and "OAS3" tags, and a link to the Swagger JSON file. Below this, the API is titled "BookApiProject". A list of five API endpoints is shown, each with a method button, a path, and a dropdown arrow:

- GET** `/books`
- POST** `/books`
- GET** `/books/{id}`
- PUT** `/books/{id}`
- DELETE** `/books/{id}`