# Chapter 2: Setting Up Your Environment 🚀

**Prerequisites**

Before getting started with .NET Aspire, ensure you have the following installed:

**1. .NET SDK (Latest Version)**

* Download and install the latest .NET SDK from the official [.NET download page](https://dotnet.microsoft.com/download).
* Verify installation by running:
* dotnet --version

**2. IDE (Integrated Development Environment)**

.NET Aspire works best with:

* **Visual Studio 2022 (v17.8 or later)** – Recommended for Windows users.
* **Visual Studio Code** with C# extension – Lightweight alternative for cross-platform development.

**3. Additional Tools**

* **Docker** (optional but recommended for containerized deployments)
* **Postman or REST client** (for testing APIs)

**Installing .NET Aspire**

To install .NET Aspire, follow these steps:

**For Visual Studio Users:**

1. Open **Visual Studio Installer**.
2. Select **Modify** on your existing installation.
3. Under **Workloads**, ensure **ASP.NET and Web Development** is selected.
4. Click **Modify** to install/update required components.

A screenshot of a computer

AI-generated content may be incorrect.

**For .NET CLI Users:**

Run the following command to install the Aspire templates:

dotnet new install Aspire.ProjectTemplates

Let’s see in action:

A screenshot of a computer

AI-generated content may be incorrect.

Here you can see, once you install the Aspire templates it shows you success message with the list of the templates.

You may check this list of templates if you have already installed the Aspire at your system by Verify the installation using following command on the dot.net CLI:

dotnet new aspire --list

## Creating Your First .NET Aspire Project

Now, let's create a simple Aspire application. There are two ways to start working with .NET aspire projects, Using CLI or IDE like Visual Studio or VS code (with extensions):

**1. Using .NET CLI:**

Run the following command to create a new Aspire project:

dotnet new aspire -n MyAspireAppByCLI

Let’s see in Action:

A screenshot of a computer program

AI-generated content may be incorrect.

Navigate into the project directory and run:

Dotnet run –project MyAspireAppBuCLI.AppHost

A screenshot of a computer program

AI-generated content may be incorrect.

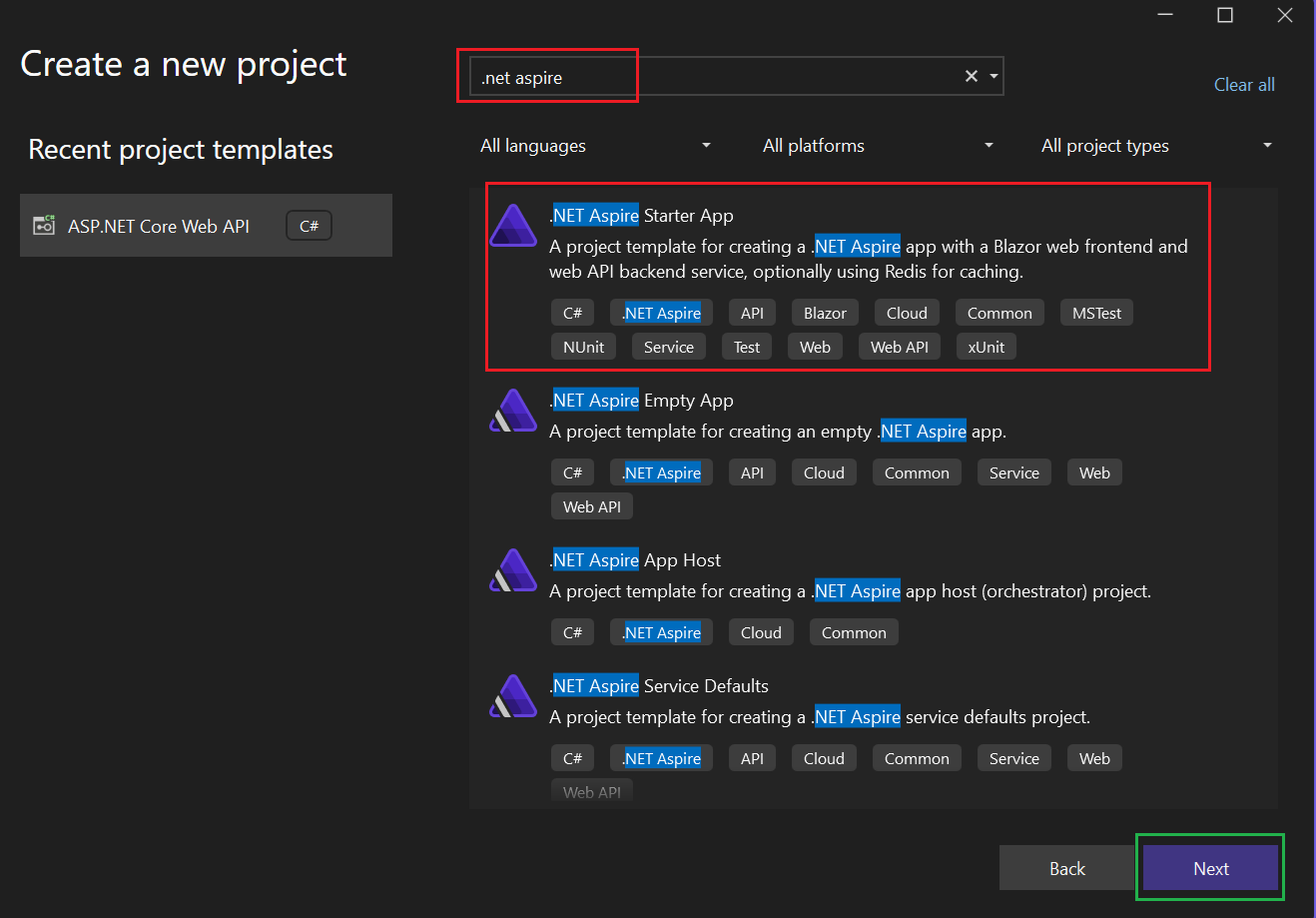
It will build the project and will run on a local port number, you can click on the linked as well as to check.

A screenshot of a computer

AI-generated content may be incorrect.

**Using Visual Studio:**

1. Open **Visual Studio** and select **Create a New Project**.
2. Search for **Aspire Application** template and select it.



1. Name your project and click **Create**.

A screenshot of a computer

AI-generated content may be incorrect.

Once we click on next it’ll ask for .NET core version you want for this project and other details, select as per the requirement and click next/create.

A screenshot of a computer program

AI-generated content may be incorrect.

.NET Aspire solution, named **AspireAppByVS2022**, consists of **four projects**, each serving a specific role in the microservices-based architecture. These projects help structure cloud-native applications efficiently.

**1. AspireAppByVS2022.ApiService (API Service)**

**📌 Purpose:**

* This is a backend service (API) that exposes endpoints to be consumed by other applications.
* It handles business logic and data processing.

**📌 Key Files:**

* Program.cs: The entry point of the API, where the web application and services are configured.
* appsettings.json: Stores configuration settings like connection strings, logging levels, etc.

**📌 How It Works:**

* Typically, this project will have **controllers** (e.g., WeatherController.cs) that define API endpoints.
* Other services (e.g., the Web UI) can call this API service for fetching/storing data.

**2. AspireAppByVS2022.AppHost (Application Host)**

**📌 Purpose:**

* This is the **main orchestration project** responsible for running all Aspire services together.
* It acts as the entry point for launching the application and managing service dependencies.

**📌 Key Files:**

* Program.cs: Configures and starts all microservices.
* appsettings.json: Stores configuration settings.

**📌 How It Works:**

* When you run the application (dotnet run inside this project), it **automatically starts all the services** like Web, API, and dependencies.
* It simplifies service discovery and makes deployment easier.

**3. AspireAppByVS2022.ServiceDefaults**

**📌 Purpose:**

* This project provides **default configurations and reusable service settings** for the entire Aspire application.
* It is a **shared library** that other projects (API, Web, AppHost) can use.

**📌 Key Files:**

* Extensions.cs: Contains helper methods and extensions for configuring services.

**📌 How It Works:**

* Instead of writing the same configurations in every project, common settings (e.g., logging, security policies) are **centralized here**.
* The API and Web projects reference this to maintain consistency.

**4. AspireAppByVS2022.Web (Web UI)**

**📌 Purpose:**

* This is the **front-end web application** (possibly an ASP.NET Core Blazor or MVC project).
* It interacts with AspireAppByVS2022.ApiService to fetch and display data.

**📌 Key Files:**

* Program.cs: Configures the web application and registers dependencies.
* appsettings.json: Stores configurations specific to the web UI.
* WeatherApiClient.cs: A client for calling the API service.
* wwwroot/: Contains static files (CSS, JavaScript, images).

**📌 How It Works:**

* It serves the **user interface** (UI) for the application.
* It fetches data from the API (AspireAppByVS2022.ApiService) and presents it to users.

### How These Projects Work Together

1. **AppHost** starts and manages the lifecycle of all services.
2. **API Service** handles business logic and serves data.
3. **Web UI** communicates with the API to display data.
4. **ServiceDefaults** provides shared configurations.

### How to Run the Application

Since **AppHost** is the main entry point, you can run the entire Aspire application using command on the dotnet CLI or simply press F5 as we are in Visual Studio 2022 IDE and it understand it well:

dotnet run --project AspireAppByVS2022.AppHost

This will: ✅ Start the **API service**  
✅ Start the **Web UI**  
✅ Load necessary configurations from **ServiceDefaults**  
✅ Ensure smooth **service orchestration**

A screenshot of a computer

AI-generated content may be incorrect.

Next, in **Chapter 3**, we’ll dive into **Core Concepts of .NET Aspire** and understand how Aspire simplifies microservices development. 🚀