ASP NET Web API REST

Introduction

CSCI E-94

Fundamentals of Cloud Computing - Azure

Joseph Ficara

Copyright © 2013-2026

Agenda

- ASP.NET Core Essentials
  - Overview
    - What is it?
    - Why do you care?
    - Essentials
  - Simple REST GET Example
  - Essential REST Verbs
    - GET, POST, PUT, PATCH,DELETE

Overview

ASP.NET Web API

Overview

- Overview
  - Framework for building HTTP-based services
  - RESTful APIs for consumption by
    - Web, Mobile, Desktop and others
  - Runs on ASP.NET Core
    - Is cross-platform
      - Windows
      - Linux
      - macOS

ASP.NET Web API
Overview

- What is it?
  - Lightweight server framework
    - Exposing functionality over HTTP
  - Uses controllers & minimal APIs
    - To handle requests and return responses
  - Supports a middleware pipeline
    - Enables extension of request & response processing
    - Centralizes cross-cutting concerns
    - Handles authentication and authorization
    - Establishes and enriches user context
      - For downstream components

ASP.NET Web API
Overview

- Why do you care?
  - Enables clean separation
    - Between frontend & backend
    - Scales from small internal services
      - To large, cloud-hosted APIs
    - Built-in support for
      - Azure App Service, Containers, and Functions
    - Industry-standard approach for building
      - Interoperable services

ASP.NET Core - Essentials

- ASP.NET Core:
  - Easy creation of REST services
    - Excellent support for HTTP Responses
  - Automatic documentation
  - Asynchronous execution
  - Support for key media types
    - JSON
    - XML
    - Plain Text
    - BSON

ASP.NET Core
Essentials

- CORS
  Cross Origin Request Sharing support
  - Support for browsers to allow some CORS
    - While rejecting others
    - See: Enable Cross-Origin Requests (CORS) in ASP.NET Core
- Middleware
  - Centralized handling of requests & responses
    - See: ASP.NET Core Middleware

ASP.NET Core
Essentials

- Supports several authentication schemes
  - ASP.NET Identity
    See: Configure ASP.NET Core Identity
  - Individual Accounts (Custom)
    - Creating projects managed in a local database
  - External Authentication Services
    - Facebook, Google, Microsoft, Apple (preview)
    - GitHub
    - OpenId Connect generic providers
    - X (formerly Twitter)
    - Microsoft Entra Id

ASP.NET Core
Essentials

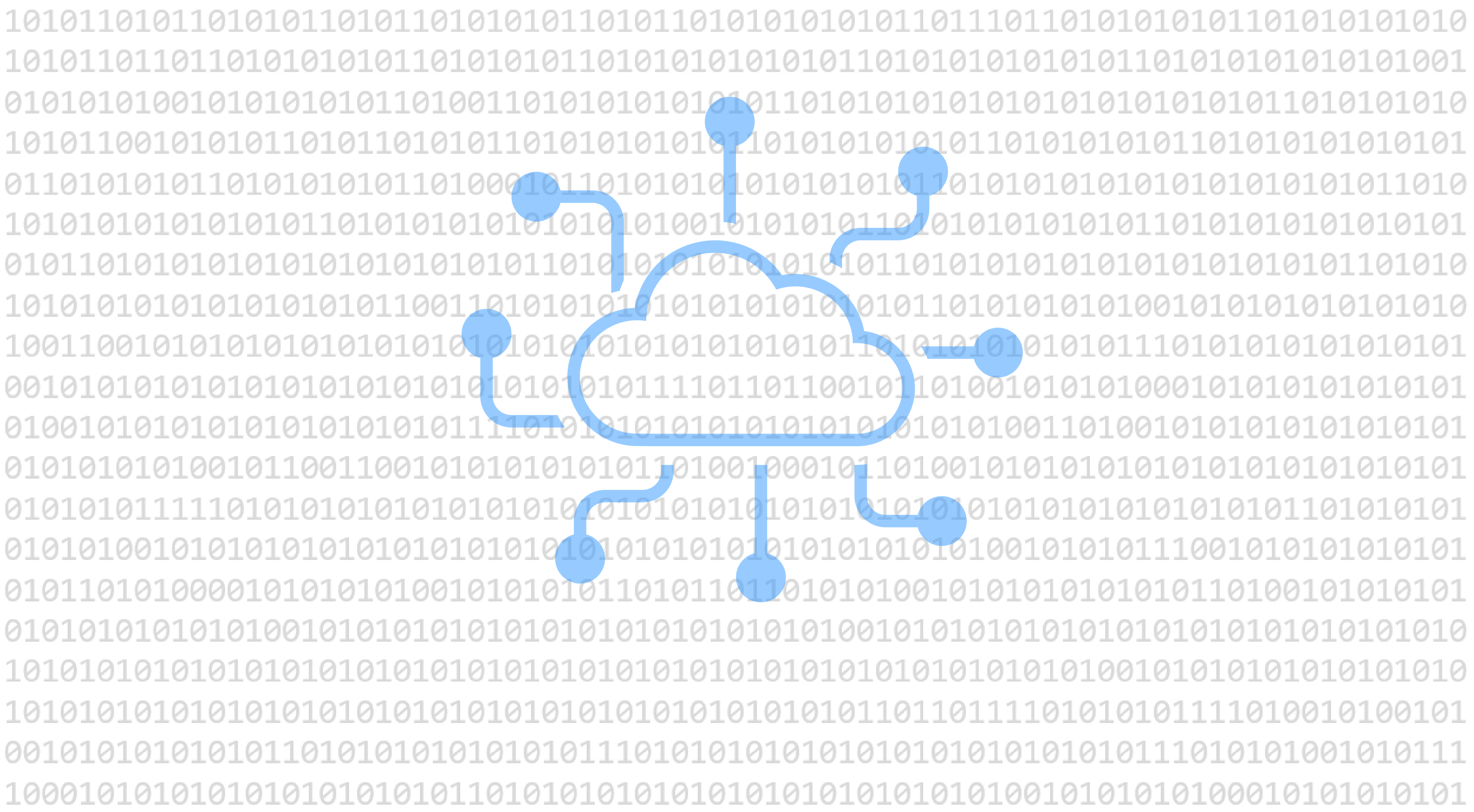
- Azure Active Directory B2C
- Basic Authentication
- Forms Authentication
- Integrated Windows Authentication
- OAUTH 2.0
- OpenID Connect
- …

ASP.NET Core
Essentials

- Support for OpenAPI 3
  - Open API JSON document
    - Useful for
      - Generating client-side SDK
      - Integration into Azure services such as API Management
  - Interactive UI
    - Allows for a “Developer” playground
      - Try out your APIs
    - Customizable
      - Style it to your liking

Overview

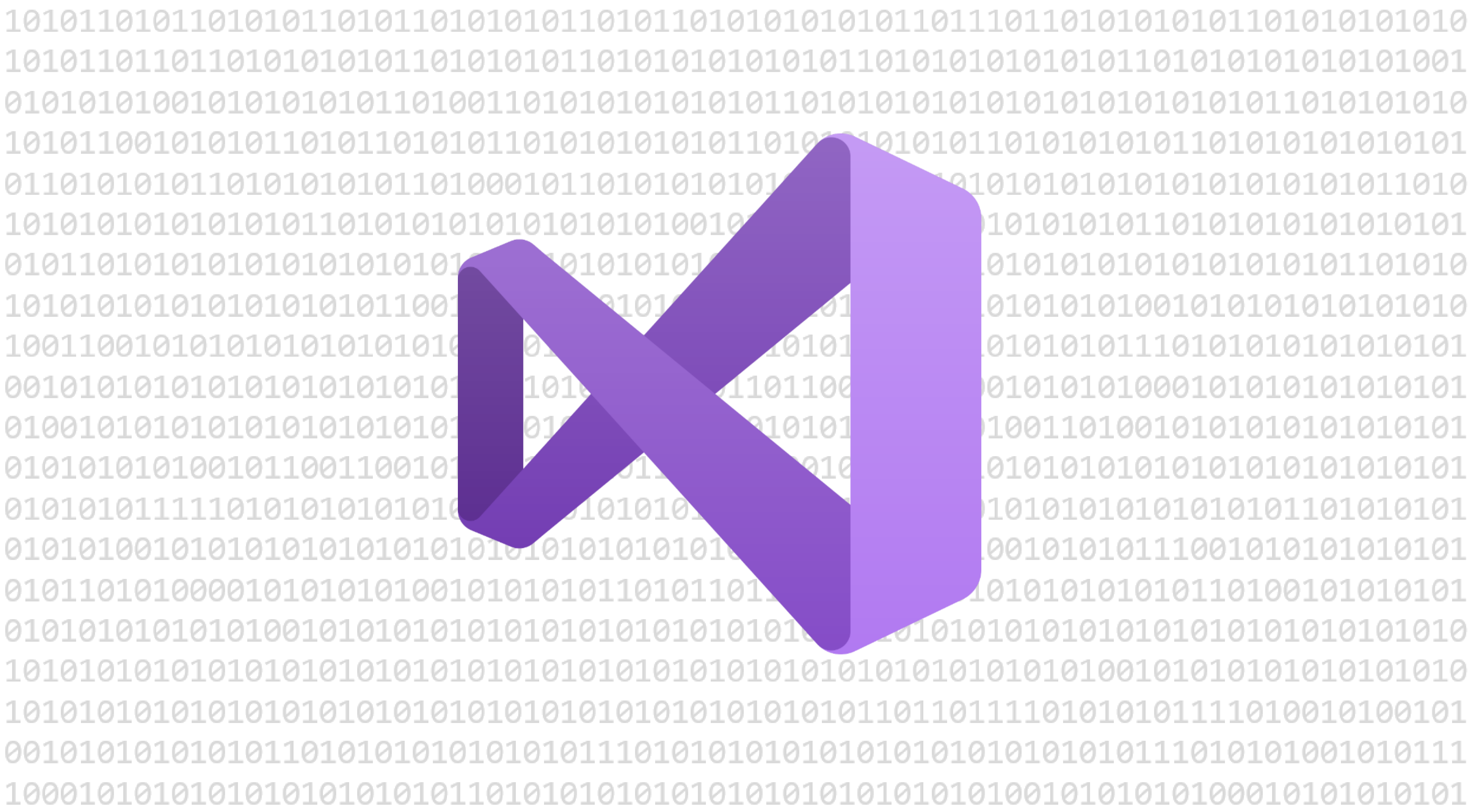
Let's Code!



Weather Azure App Service

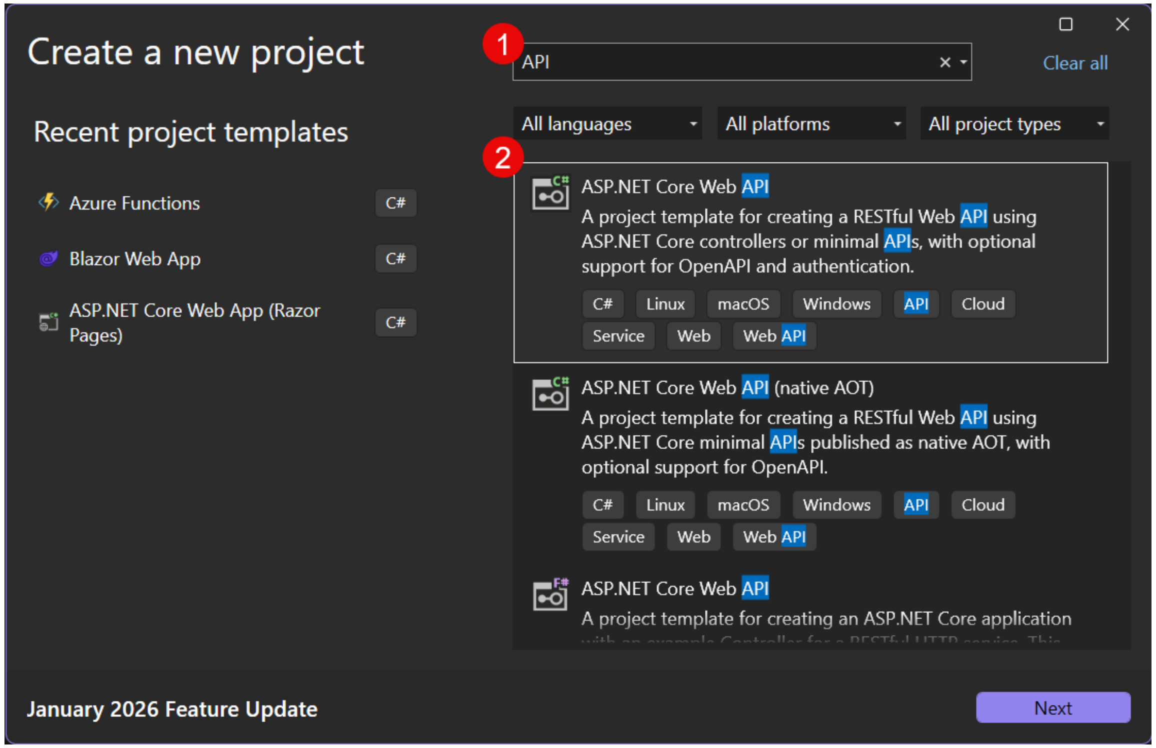
- Several templates available for .NET 10
  - ASP.NET Core Empty
  - ASP.NET Core Web App
  - ASP.NET Core Web API
  - ASP.NET Core Web API (native AOT)
  - ASP.NET Core Web App (Model-View-Controller)
  - ASP.NET Core gRPC Service
  - ASP.NET Core Web App (Razor Pages)
  - …

Visual Studio 2026



Weather Azure App Service

- Starting with the default template …
  - Create a new project



Create a new project

Recent project templates

Azure Functions	C#
Blazor Web App	C#
ASP.NET Core Web App (Razor Pages)	C#

ASP.NET Core Web API
A project template for creating a RESTful Web API using ASP.NET Core controllers or minimal APIs, with optional support for OpenAPI and authentication.
C# Linux macOS Windows API Cloud Service Web Web API

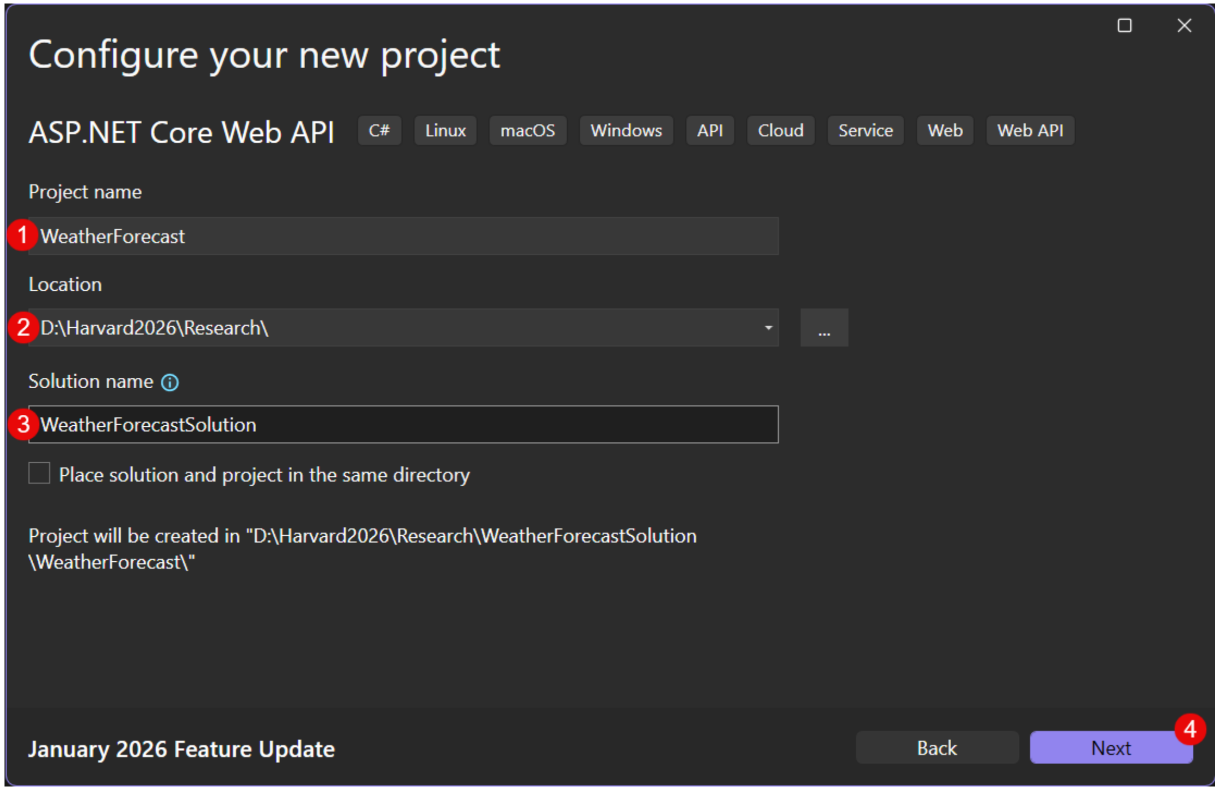
ASP.NET Core Web API (native AOT)
A project template for creating a RESTful Web API using ASP.NET Core minimal APIs published as native AOT, with optional support for OpenAPI.
C# Linux macOS Windows API Cloud Service Web Web API

ASP.NET Core Web API
A project template for creating an ASP.NET Core application

January 2026 Feature Update

Weather Azure App Service

- Additional Information
  - Name your project and solution



Configure your new project

ASP.NET Core Web API

Project name

1 WeatherForecast

Location

2 D:\Harvard2026\Research\

Solution name

3 WeatherForecastSolution

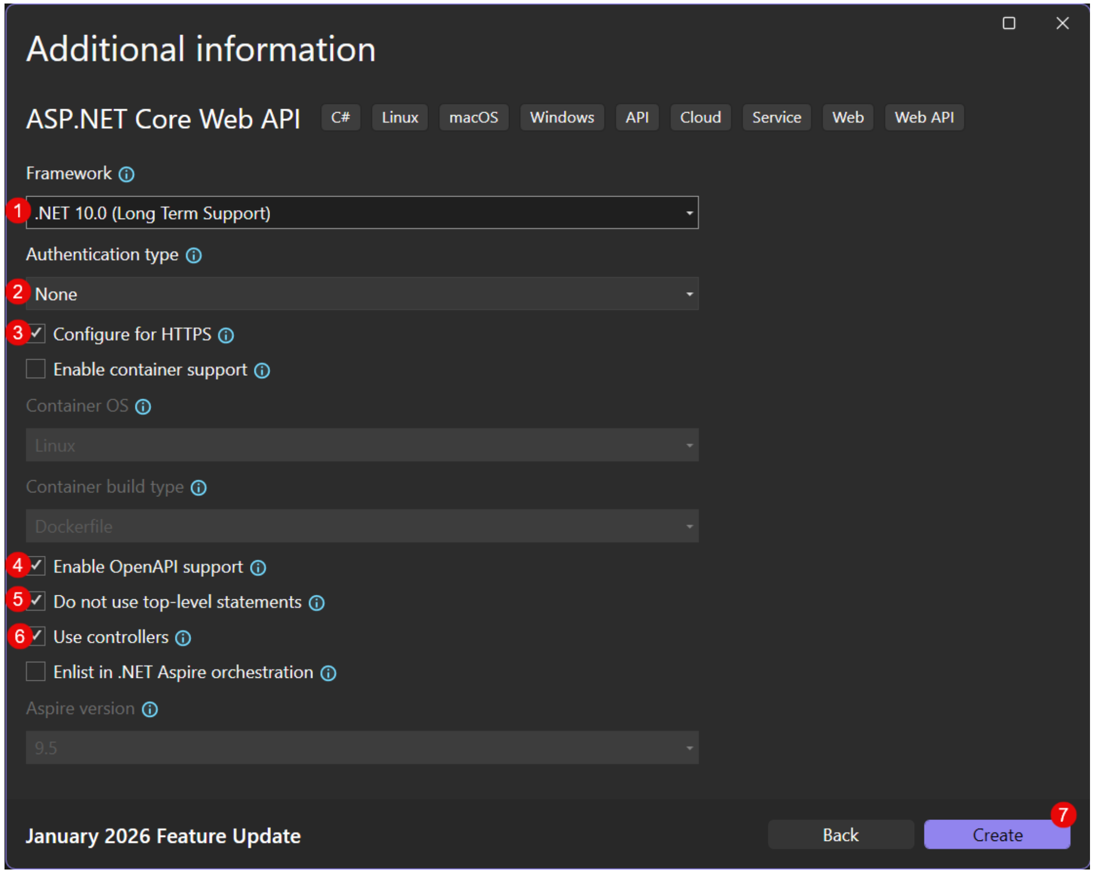
☐ Place solution and project in the same directory

Project will be created in "D:\Harvard2026\Research\WeatherForecastSolution \WeatherForecast\"

January 2026 Feature Update

Weather Azure App Service

- Additional Information



Additional information

ASP.NET Core Web API

Framework ⓘ

1 .NET 10.0 (Long Term Support)

Authentication type ⓘ

2 None

3 ✓ Configure for HTTPS ⓘ

☐ Enable container support ⓘ

Container OS ⓘ

Linux

Container build type ⓘ

Dockerfile

4 ✓ Enable OpenAPI support ⓘ

5 ✓ Do not use top-level statements ⓘ

6 ✓ Use controllers ⓘ

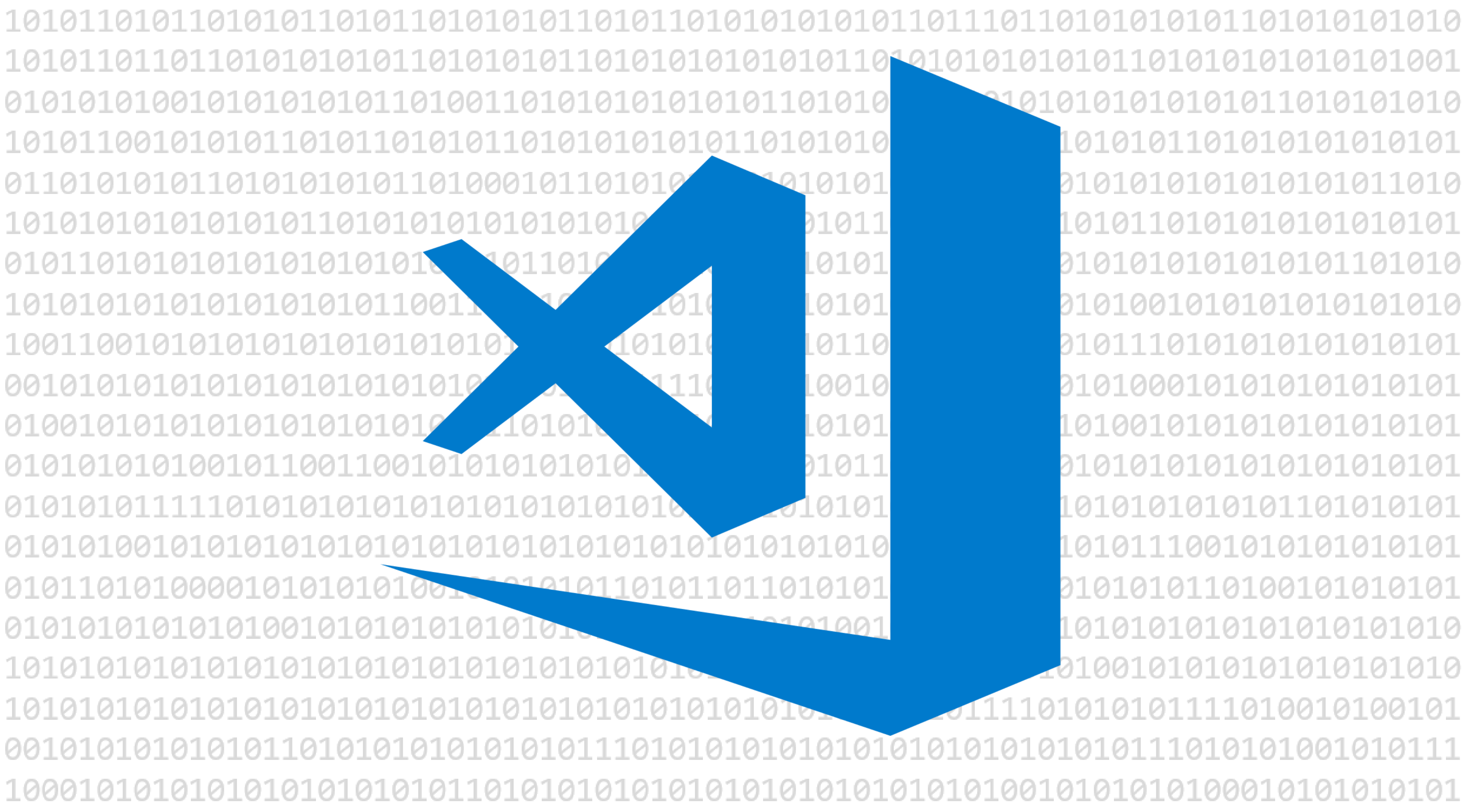
☐ Enlist in .NET Aspire orchestration ⓘ

Aspire version ⓘ

9.5

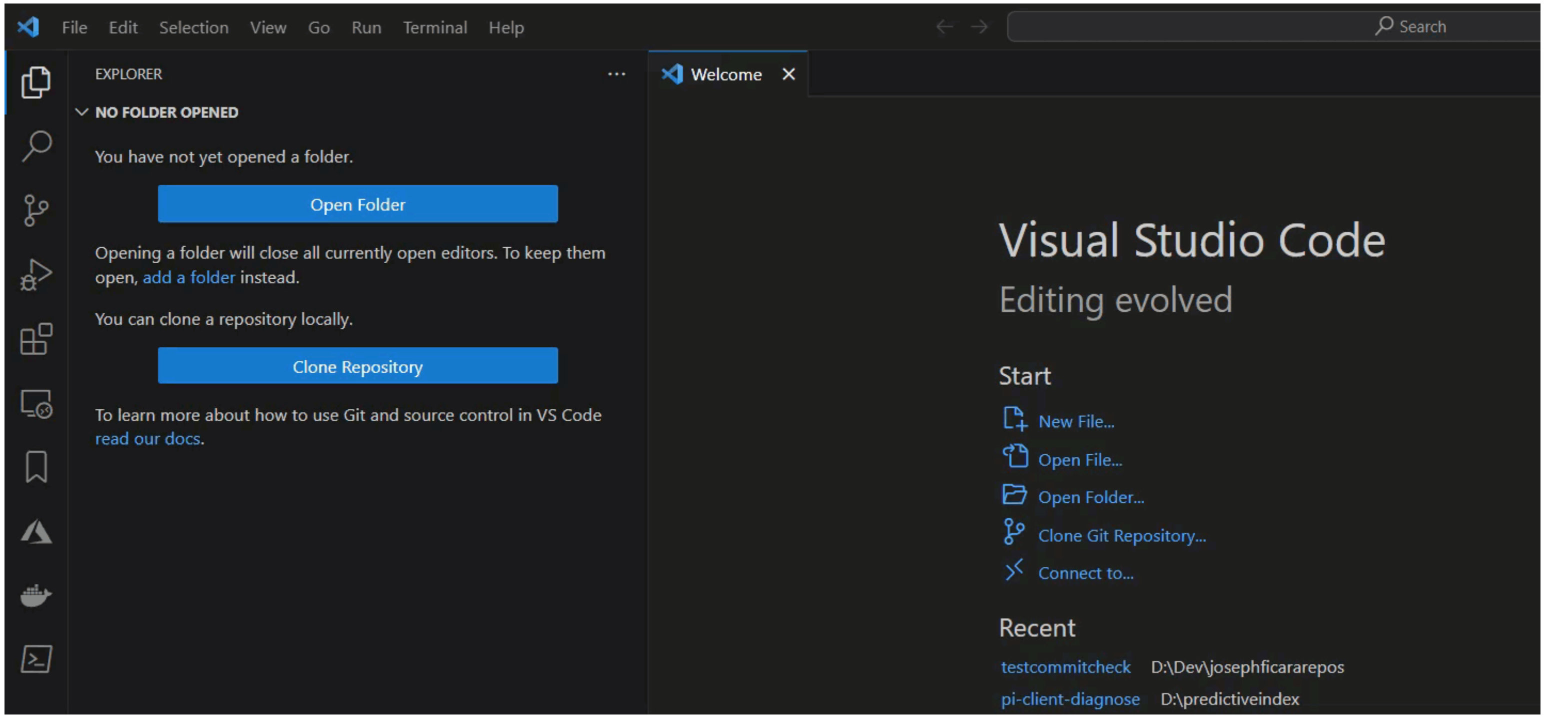
January 2026 Feature Update

VS Code



Weather Azure App Service

- VS Code
  - Install the VS Code Dev Kit Extension



You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

You can clone a repository locally.

To learn more about how to use Git and source control in VS Code read our docs.

Visual Studio Code

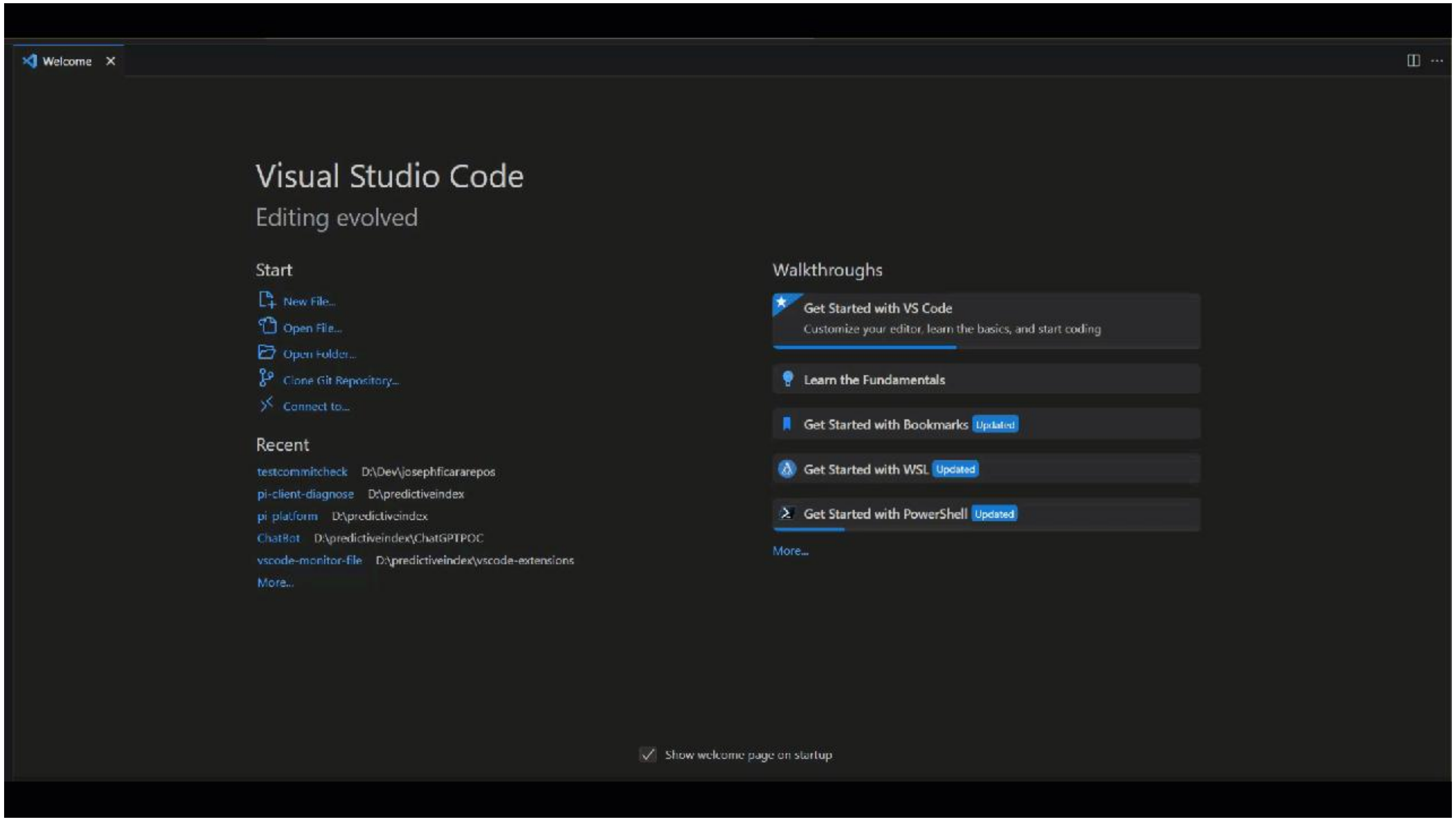
Editing evolved

testcommitcheck D:\Dev\josephficararepos

pi-client-diagnose D:\predictiveindex

Weather Azure App Service
VSCode

- Follow steps Getting Started with C# Dev Kit



Welcome ×

Visual Studio Code
Editing evolved

Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...

Recent

testcommitcheck	D:\Dev\josephficararepos
pi-client-diagnose	D:\predictiveindex
pi platform	D:\predictiveindex
ChatBot	D:\predictiveindex\ChatGPTPOC
vscode-monitor-file	D:\predictiveindex\vscode-extensions

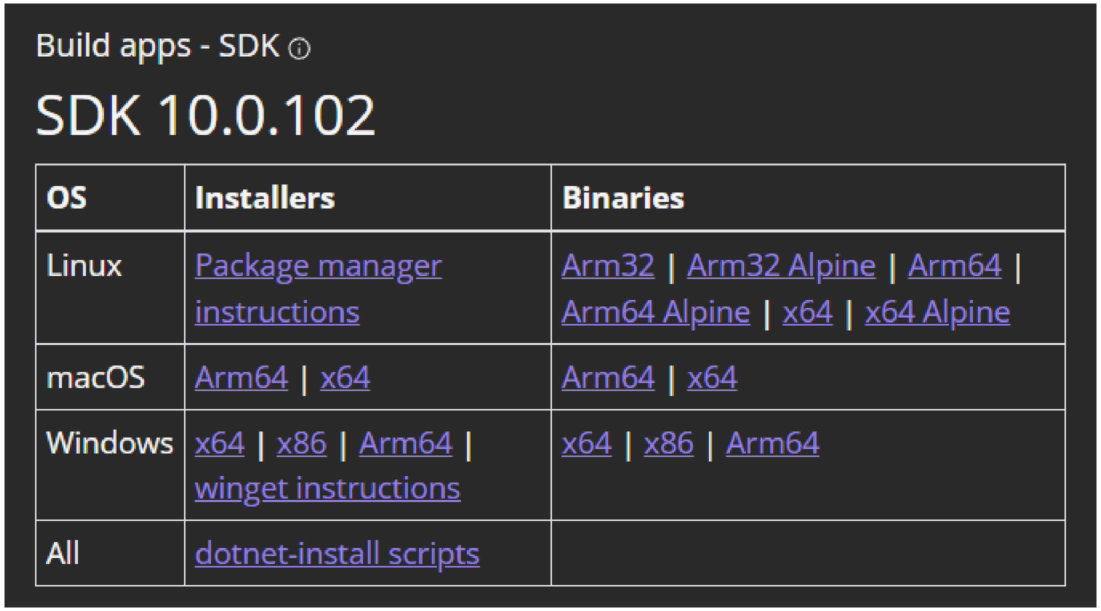
More...

Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Learn the Fundamentals
Get Started with Bookmarks Updated
Get Started with WSL Updated
Get Started with PowerShell Updated
More...

✓ Show welcome page on startup

Weather Azure App Service
VSCode

- .NET 10 SDK Download & Install
  - Download .NET 10.0



Build apps - SDK ⓘ

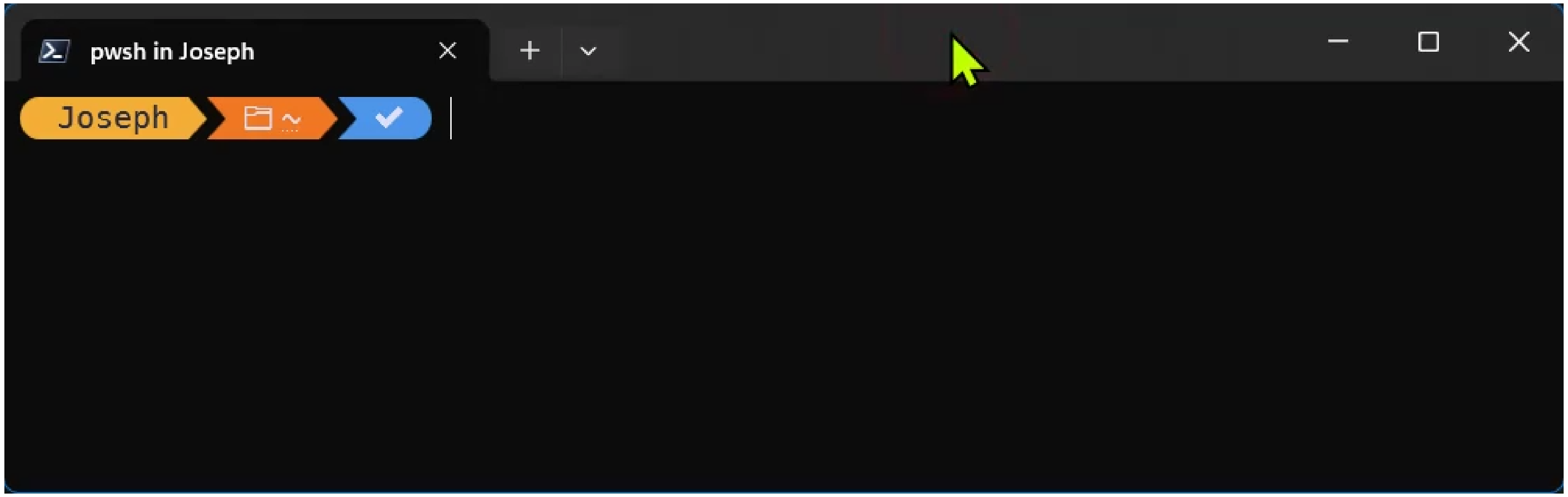
SDK 10.0.102

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 | Arm32 Alpine | Arm64 | Arm64 Alpine | x64 | x64 Alpine
macOS	Arm64 | x64	Arm64 | x64
Windows	x64 | x86 | Arm64 | winget instructions	x64 | x86 | Arm64
All	dotnet-install scripts	

Weather Azure App Service

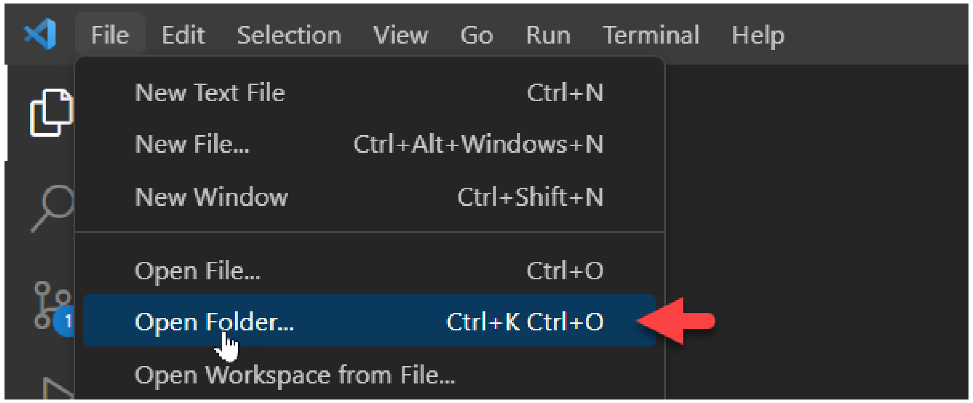
VSCode

- Verify .NET 10 SDK is Installed
  - 10.0.102 or greater is fine



Weather Azure App Service
VSCode

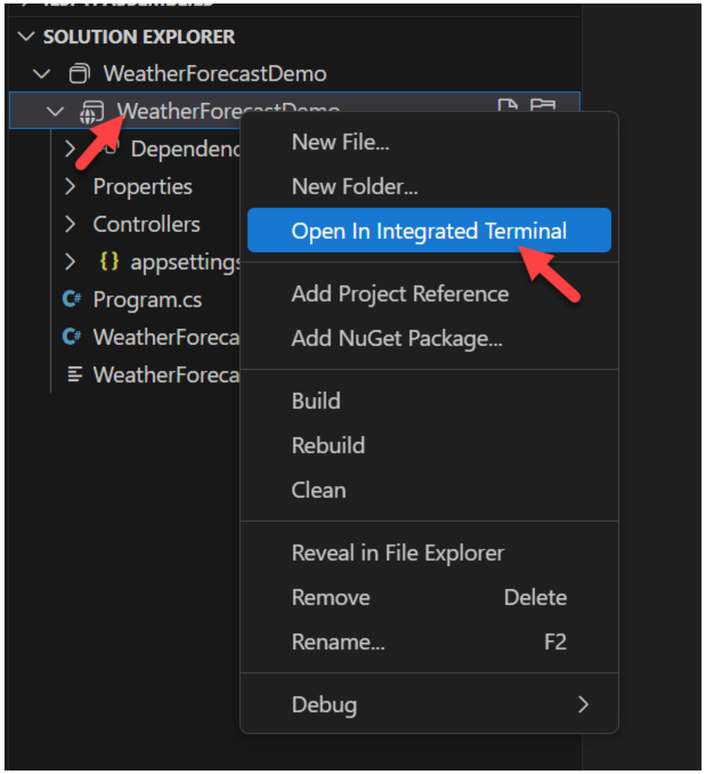
- Select an empty folder in VS Code



- Use the command line to create your project
  - dotnet new webapi --framework net10.0 --use-controllers --use-program-main -n <project name>

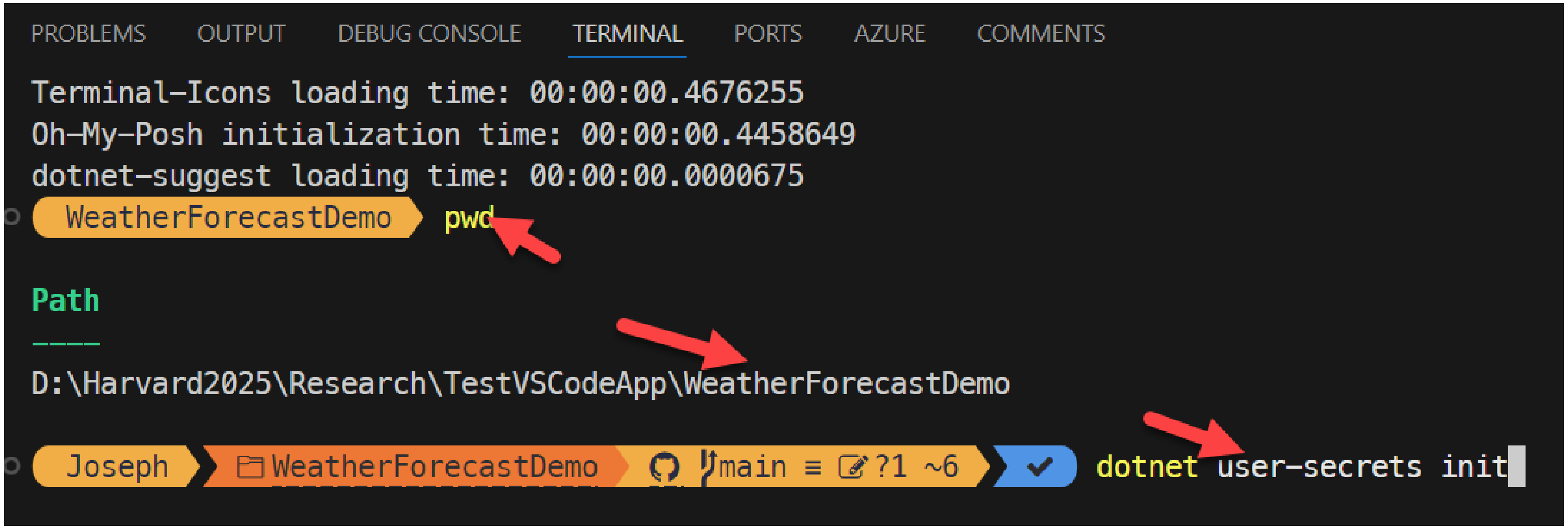
Weather Azure App Service
VSCode

- Generate your **secrets.json** file
  - Open a terminal in your project directory



Weather Azure App Service
VSCode

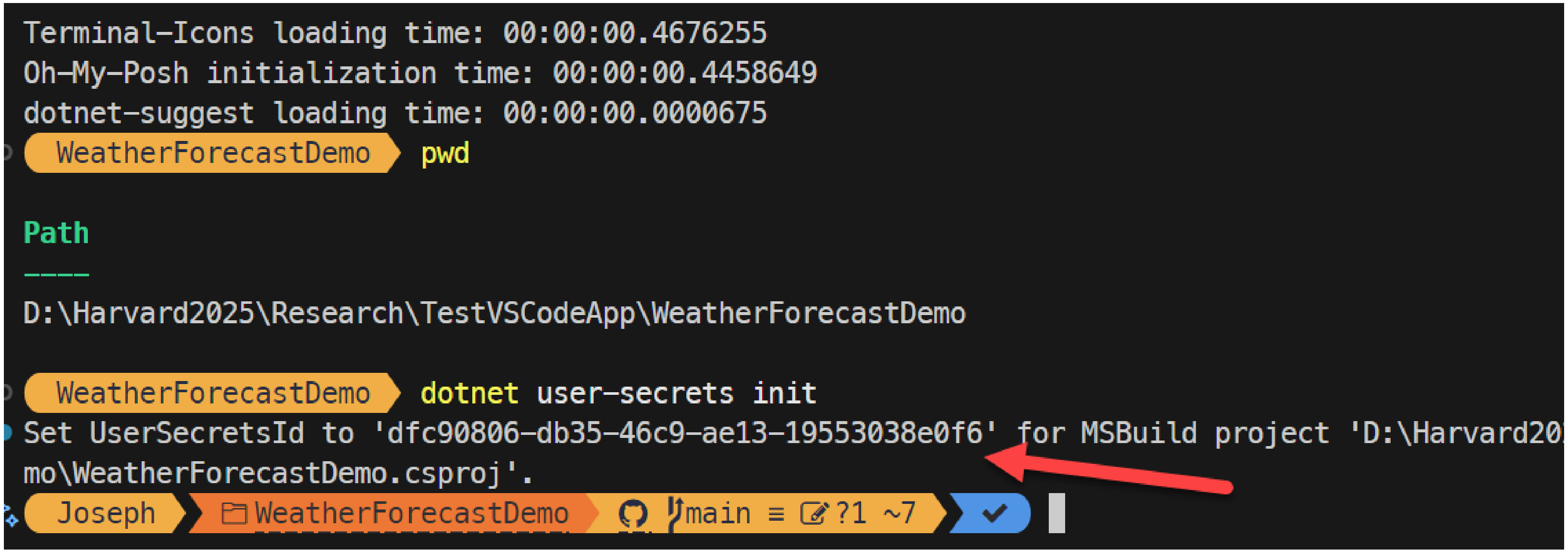
- Generate your **secrets.json** file …
  - Verify that you are in your project directory
  - Run **dotnet user-secrets init**



```
Terminal-Icons loading time: 00:00:00.4676255
Oh-My-Posh initialization time: 00:00:00.4458649
dotnet-suggest loading time: 00:00:00.0000675
WeatherForecastDemo  pwd

Path
----
D:\Harvard2025\Research\TestVSCodeApp\WeatherForecastDemo

Joseph  WeatherForecastDemo   main ≡  ?1 ~6  ✓  dotnet user-secrets init
```

Weather Azure App Service
VSCode

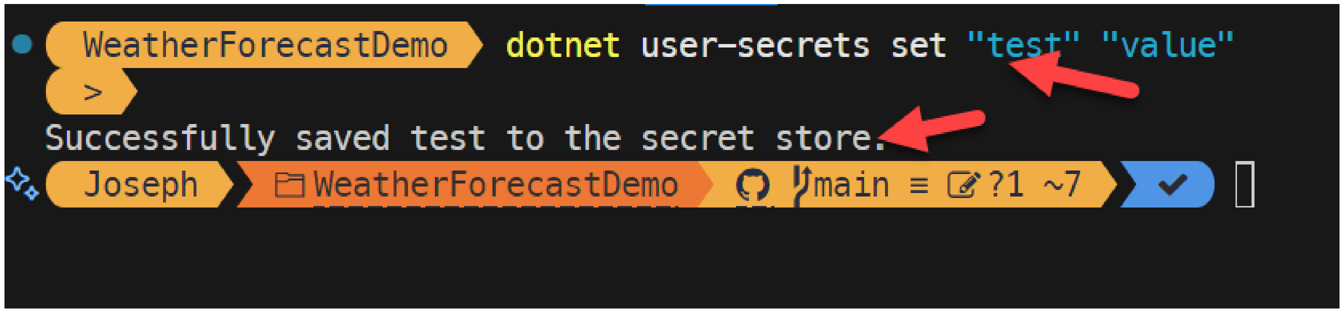
- Generate your **secrets.json** file …
  - Result should look like this



```
Terminal-Icons loading time: 00:00:00.4676255
Oh-My-Posh initialization time: 00:00:00.4458649
dotnet-suggest loading time: 00:00:00.0000675
 WeatherForecastDemo  pwd

Path
----
D:\Harvard2025\Research\TestVSCodeApp\WeatherForecastDemo

 WeatherForecastDemo  dotnet user-secrets init
Set UserSecretsId to 'dfc90806-db35-46c9-ae13-19553038e0f6' for MSBuild project 'D:\Harvard20
mo\WeatherForecastDemo.csproj'.
 Joseph  WeatherForecastDemo  main ≡ ?1 ~7  ✔
```

Weather Azure App Service
VSCode

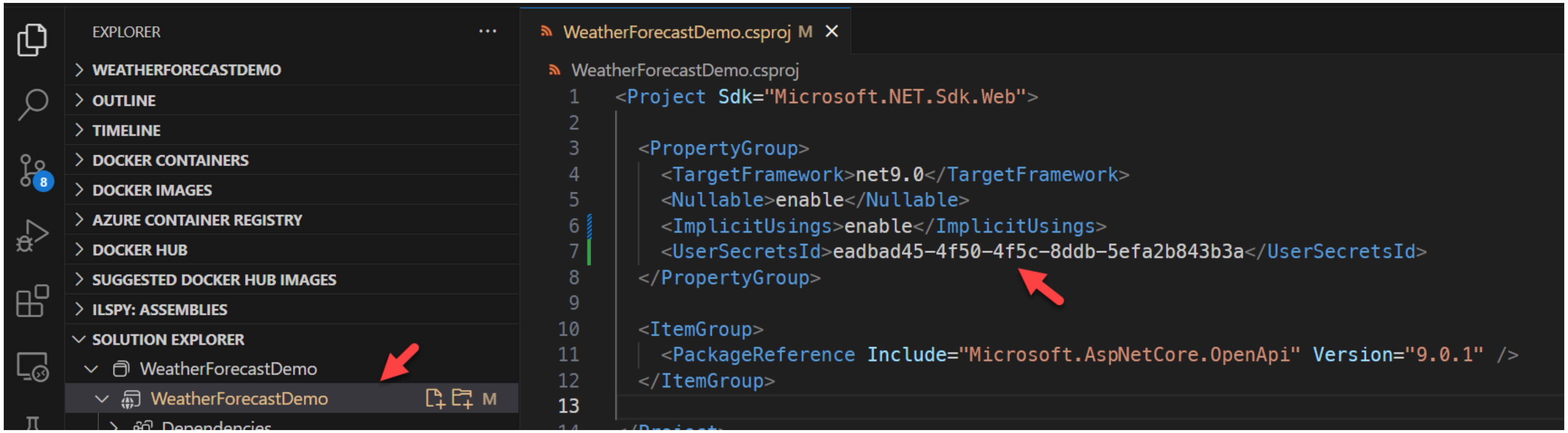
- Generate your **secrets.json** file …
  - Set a test value
    - To generate the **secrets.json** file



```
WeatherForecastDemo  dotnet user-secrets set "test" "value"
>
Successfully saved test to the secret store.
Joseph  WeatherForecastDemo  main ≡ ?1 ~7
```

Weather Azure App Service
VSCode

- Generate your **secrets.json** file …
  - Double click on your project file
    - To verify the secret folder name



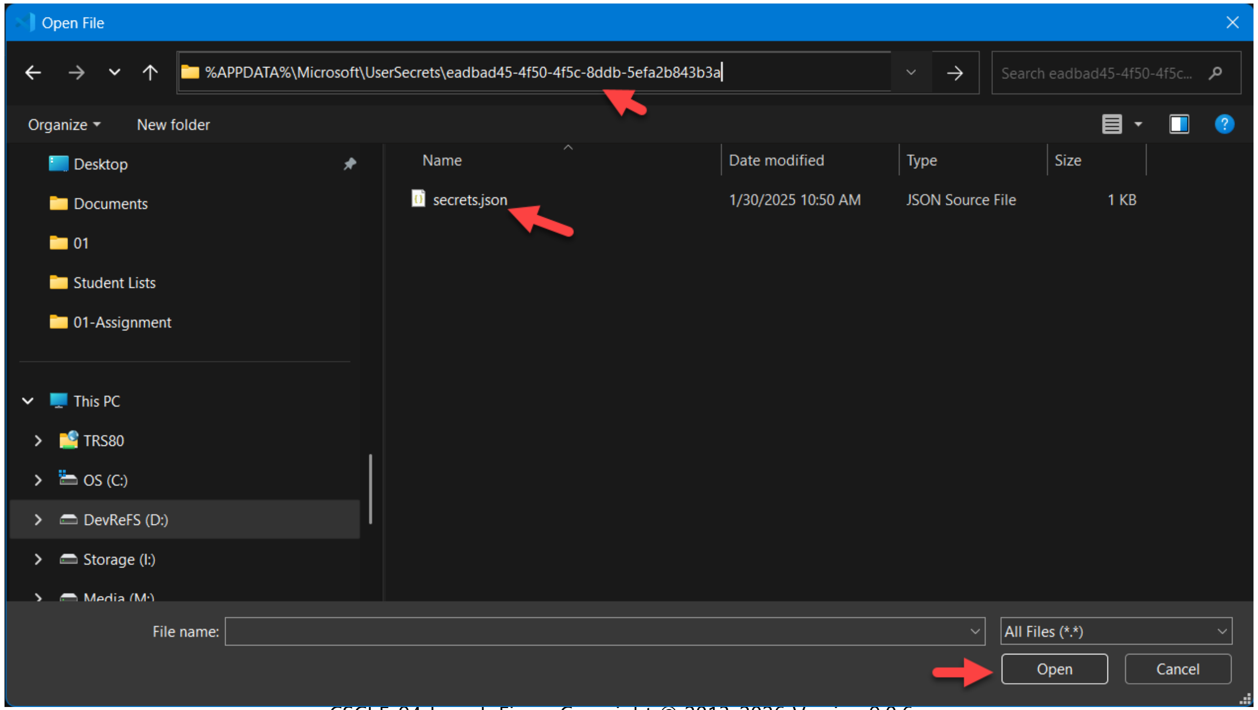
WeatherForecastDemo.csproj

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net9.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <UserSecretsId>eadbad45-4f50-4f5c-8ddb-5efa2b843b3a</UserSecretsId>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="9.0.1" />
  </ItemGroup>

</Project>
```

Weather Azure App Service
VSCode

- Edit the **secrets.json** file in vs code
  - The windows path will be

%APPDATA%\Microsoft\UserSecrets\eadbad45-4f50-4f5c-8ddb-5efa2b843b3a

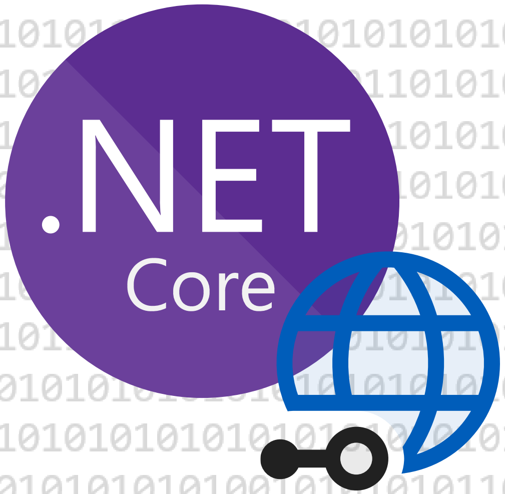


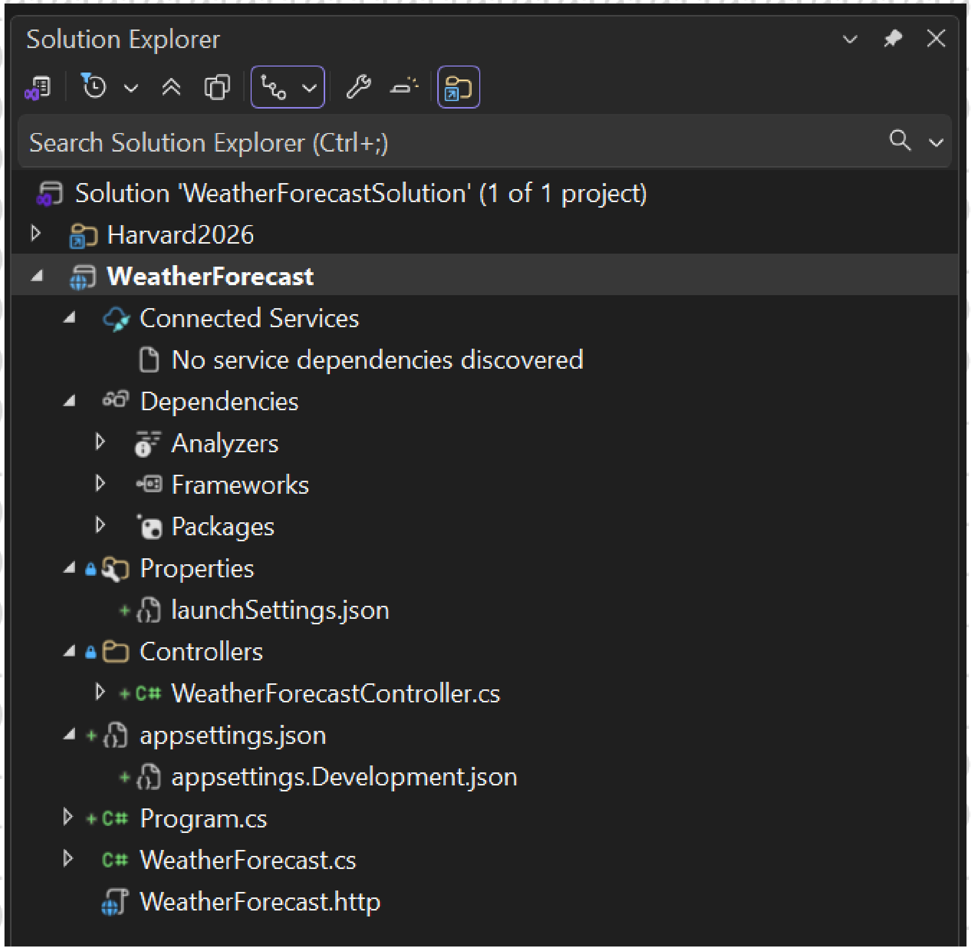
Name	Date modified	Type	Size
secrets.json	1/30/2025 10:50 AM	JSON Source File	1 KB

Weather Azure App Service
VSCode

- **secrets.json** folder paths for
  - Windows
    - %APPDATA%\Microsoft\UserSecrets\{UserSecretsId}\secrets.json
  - macOS/Linux
    - ~/.microsoft/usersecrets/{UserSecretsId}/secrets.json

ASP.NET Core Web API Structure




Solution Explorer

Search Solution Explorer (Ctrl+;)

```
Solution 'WeatherForecastSolution' (1 of 1 project)
  Harvard2026
  WeatherForecast
    Connected Services
      No service dependencies discovered
    Dependencies
      Analyzers
      Frameworks
      Packages
    Properties
      launchSettings.json
    Controllers
      WeatherForecastController.cs
    appsettings.json
      appsettings.Development.json
    Program.cs
    WeatherForecast.cs
    WeatherForecast.http
```

Weather Azure App Service

- ASP.NET Core Web API template
  - Project structure
    - Properties
      - Publisher profiles will reside here
        - Used to define how to publish your Web API to Azure
          - Don't share them or put them in source control
      - Service Dependencies
        - Azure Resource Templates that define the resources used
      - launchSettings.json
        - Used by Visual Studio to direct how to run the app locally

Weather Azure App Service

ASP.NET Core Web API template …

- Project structure …
  - Controllers
    - Classes that handle HTTP requests go here
    - Http Verbs automatically routed to methods
      - HTTP Verb GET routes to a method called Get()
    - *Clearer to use the C# Attribute* [HttpGet]
    - Controllers/WeatherForcastController.cs
      - Sample code that generates random weather results

Weather Azure App Service
ASP.NET Core Web API template …

- Project structure …
  - appsettings.json
    - Contain configuration in JSON format
      - appsettings.development.json
        - Settings used for local development
  - Program.cs
    - Main entry point for the Web API app
  - WeatherForcast.cs
    - Class that defines result of GET action
  - WeatherForecast.http
    - A .http file used for testing your Web APIs

Weather Azure App Service

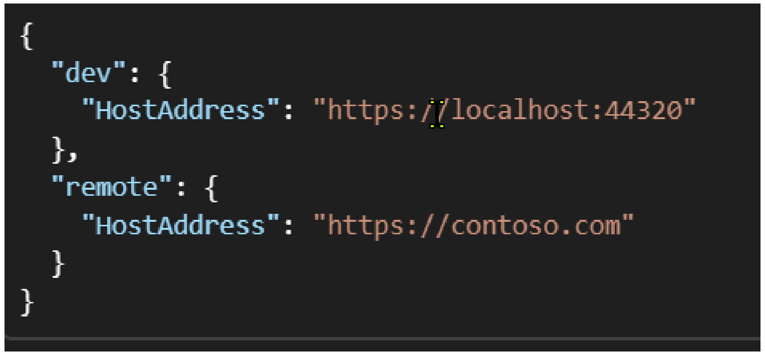
ASP.NET Core Web API template …

- Project structure …
  - http-client.env.json
    - Not added by default
    - Used to define the environments for the .http file
  - readme.md
    - Not added by default
    - Used to describe / provide notes about the application
- Let's create both using GitHub Copilot

Weather Azure App Service

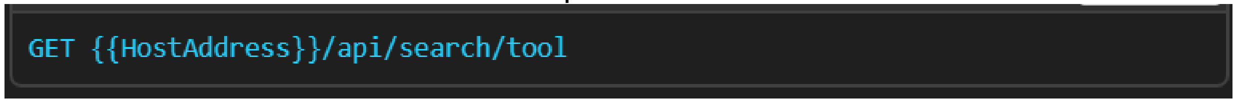
AI Coding

- http-client.env.json file
  - Defines environments used for testing
    - Specifies an environment name and URL
      - dev & remote are the environment names
        - HostAddress is the variable the URL is assigned to



```
{
  "dev": {
    "HostAddress": "https://localhost:44320"
  },
  "remote": {
    "HostAddress": "https://contoso.com"
  }
}
```

    - Referenced in the http file like this

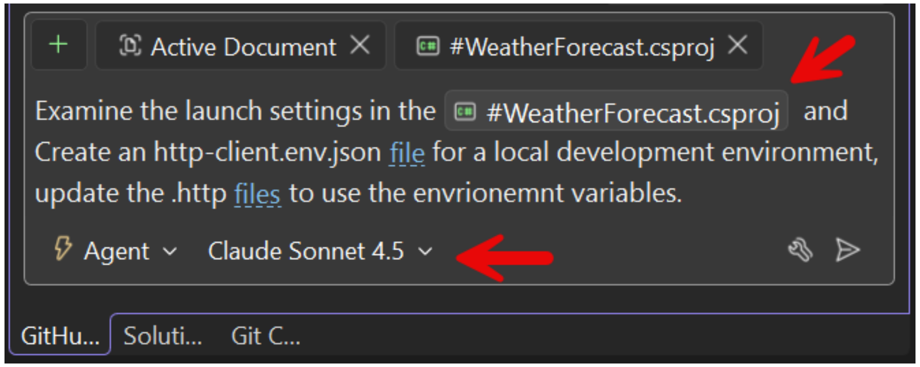


```
GET {{HostAddress}}/api/search/tool
```

Weather Azure App Service

AI Coding

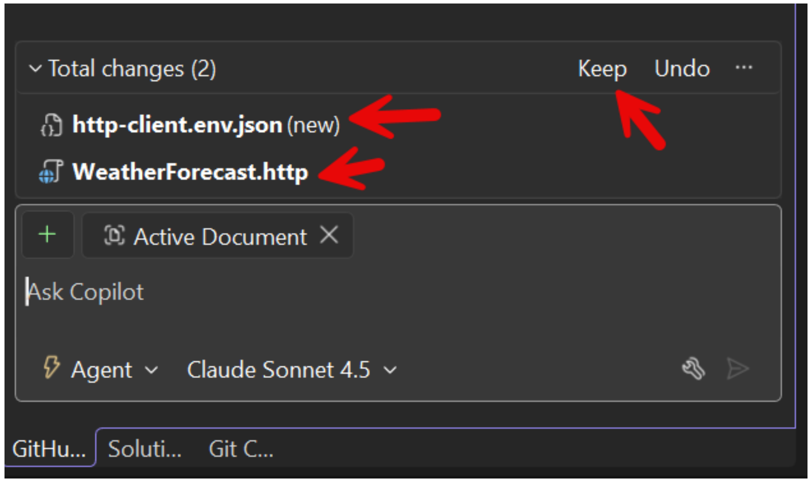
- Many more capabilities
  - See Environment Files
- GitHub Copilot can create .env file
  - update .http file



Active Document

#WeatherForecast.csproj

Examine the launch settings in the #WeatherForecast.csproj and Create an http-client.env.json file for a local development environment, update the .http files to use the envrionemnt variables.



Total changes (2)

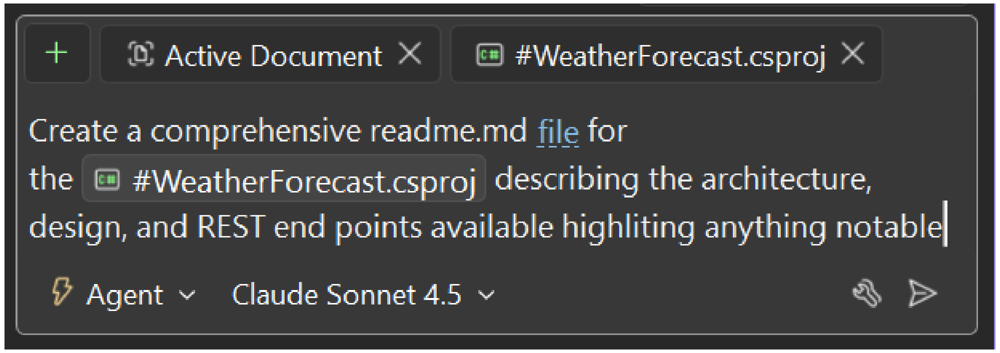
http-client.env.json (new)

WeatherForecast.http

Weather Azure App Service

AI Coding

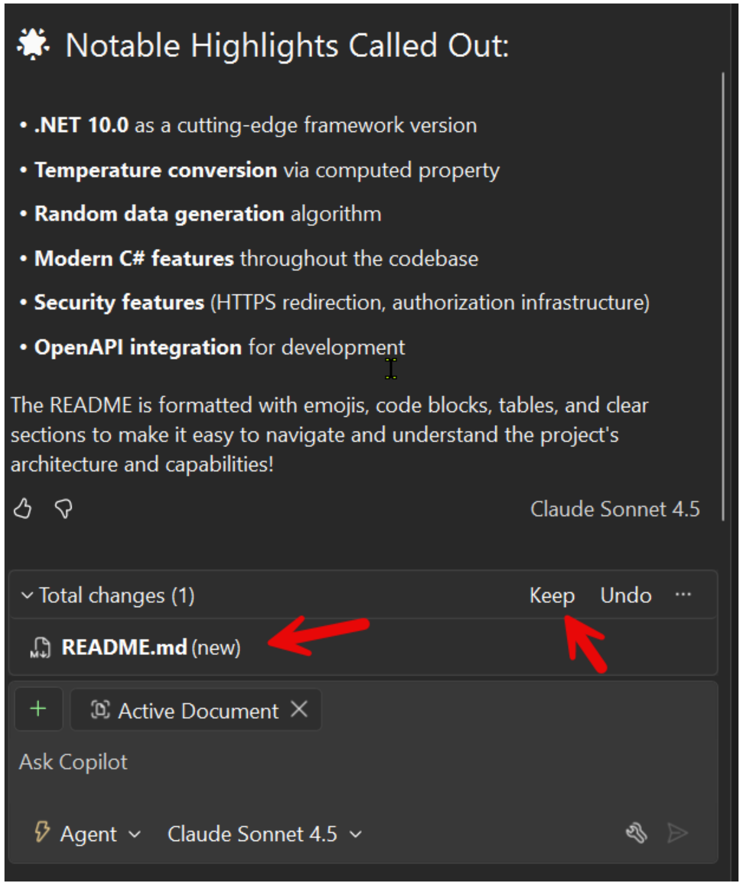
- GitHub Copilot can also create the readme.md file



Active Document | #WeatherForecast.csproj

Create a comprehensive readme.md file for the #WeatherForecast.csproj describing the architecture, design, and REST end points available highliting anything notable

Agent | Claude Sonnet 4.5



Notable Highlights Called Out:

- **.NET 10.0** as a cutting-edge framework version
- **Temperature conversion** via computed property
- **Random data generation** algorithm
- **Modern C# features** throughout the codebase
- **Security features** (HTTPS redirection, authorization infrastructure)
- **OpenAPI integration** for development

The README is formatted with emojis, code blocks, tables, and clear sections to make it easy to navigate and understand the project's architecture and capabilities!

Claude Sonnet 4.5

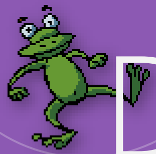
Total changes (1) Keep Undo

README.md (new)

Active Document

Ask Copilot

Agent | Claude Sonnet 4.5



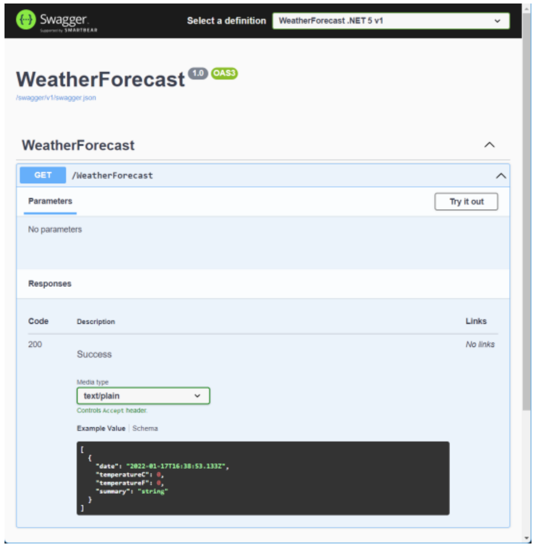
Demo

ASP.NET Core API Template Example

```
WeatherForecastSolution.sln
   WeatherForecast.csproj
```

Adding REST Documentation

- Web API supports documentation
  - UI Needs to be added
    - Via Swashbuckle nuget package
      `Swashbuckle.AspNetCore (10.x)`



WeatherForecast
/swagger/v1/swagger.json

WeatherForecast

/WeatherForecast

Parameters

No parameters

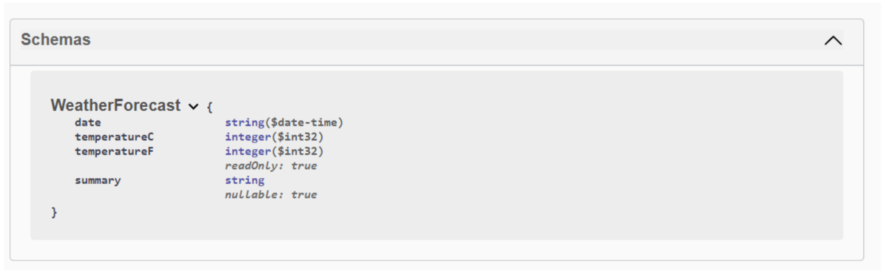
Responses

Code	Description	Links
200	Success	*No links*

Media type: text/plain
Controls Accept header.

Example Value | Schema

```
[
  {
    "date": "2022-01-17T16:38:53.133Z",
    "temperatureC": 0,
    "temperatureF": 0,
    "summary": "string"
  }
]
```

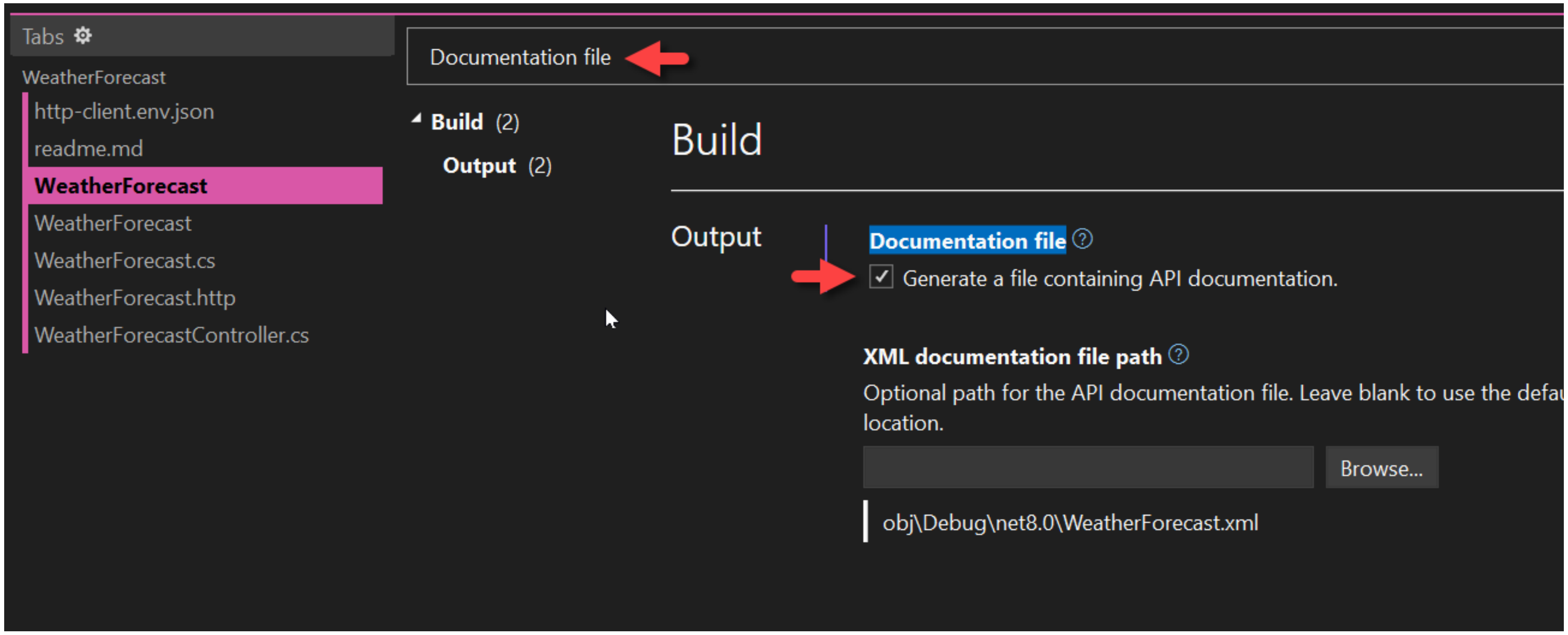


Schemas

```
WeatherForecast ⌄ {
   date               string($date-time)
   temperatureC       integer($int32)
   temperatureF       integer($int32)
                      readOnly: true
   summary            string
                      nullable: true
}
```

Adding REST Documentation

- Suppress warnings
  - Right click on the project & choose properties
  - Search for **documentation file**



Tabs

WeatherForecast

http-client.env.json

readme.md

WeatherForecast

WeatherForecast

WeatherForecast.cs

WeatherForecast.http

WeatherForecastController.cs

Documentation file

Build (2)

Output (2)

Build

Output

Documentation file

☑ Generate a file containing API documentation.

XML documentation file path

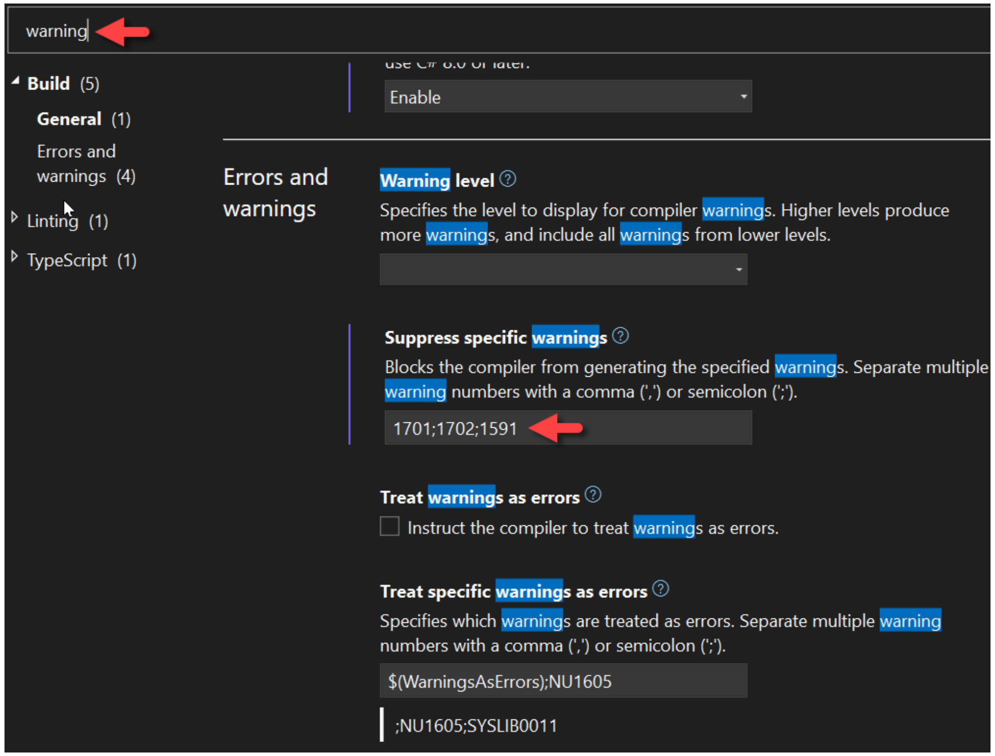
Optional path for the API documentation file. Leave blank to use the defa
location.

Browse...

obj\Debug\net8.0\WeatherForecast.xml

Adding REST Documentation

- Suppress warnings
  - Right click on the project and choose properties
  - Search for **warning**, add 1591 to suppress comment warnings



warning

Build (5)
General (1)
Errors and warnings (4)
Linting (1)
TypeScript (1)

use C# 8.0 or later.

Enable

Errors and warnings

Warning level

Specifies the level to display for compiler warnings. Higher levels produce more warnings, and include all warnings from lower levels.

Suppress specific warnings

Blocks the compiler from generating the specified warnings. Separate multiple warning numbers with a comma (',') or semicolon (';').

1701;1702;1591

Treat warnings as errors

☐ Instruct the compiler to treat warnings as errors.

Treat specific warnings as errors

Specifies which warnings are treated as errors. Separate multiple warning numbers with a comma (',') or semicolon (';').

$(WarningsAsErrors);NU1605

;NU1605;SYSLIB0011

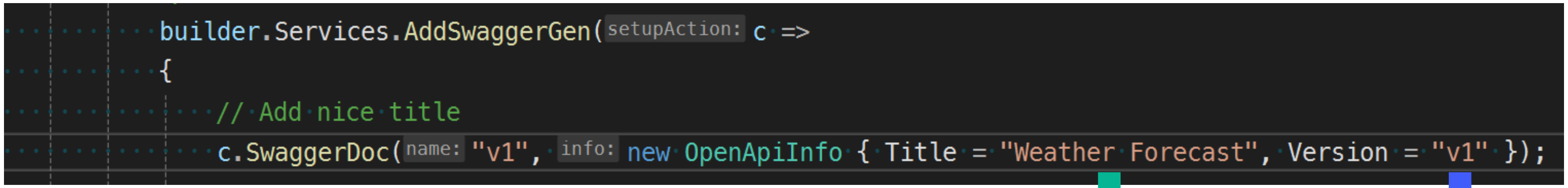
Adding REST Documentation

Swagger Initialization code

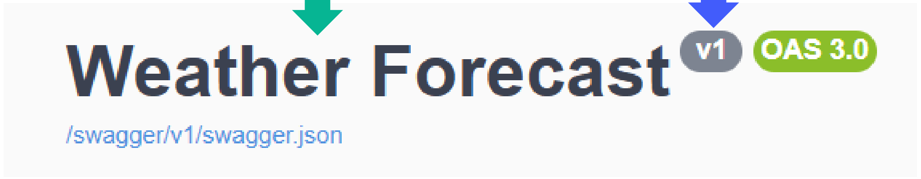
```
var builder = WebApplication.CreateBuilder(args);

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

// Add Swashbuckle Swagger generation
builder.Services.AddSwaggerGen(c =>
{
    // Add nice title
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "WeatherForecast Testing", Version
= "v1" });

    // Add documentation via C# XML Comments
    var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
    var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);
    c.IncludeXmlComments(xmlPath);

});
```

Adding REST Documentation

- Title and Version Example



```
builder.Services.AddSwaggerGen(setupAction: c =>
{
    // Add nice title
    c.SwaggerDoc(name: "v1", info: new OpenApiInfo { Title = "Weather Forecast", Version = "v1" });
```

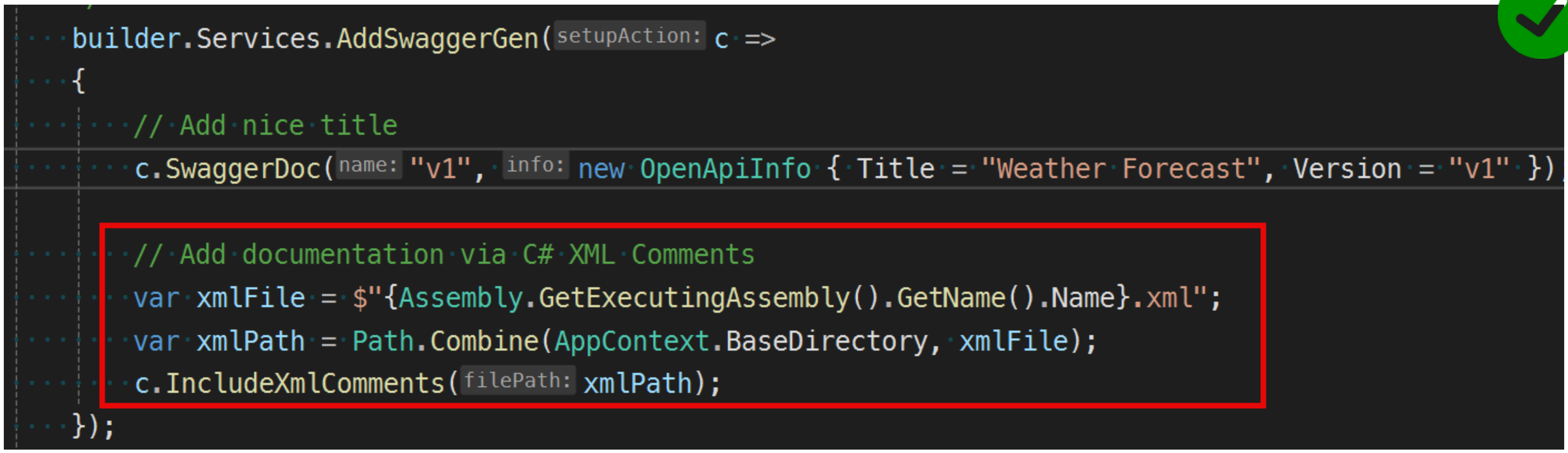


Weather Forecast v1 OAS 3.0

/swagger/v1/swagger.json

Adding REST Documentation

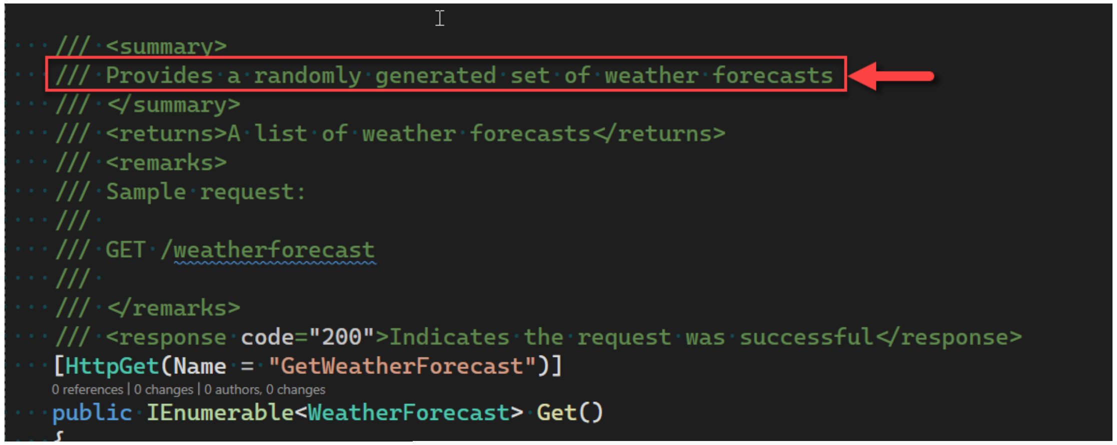
- Don't forget add the code to include the xml file
  - Why?
    - To see the C# XML API Comments, you added



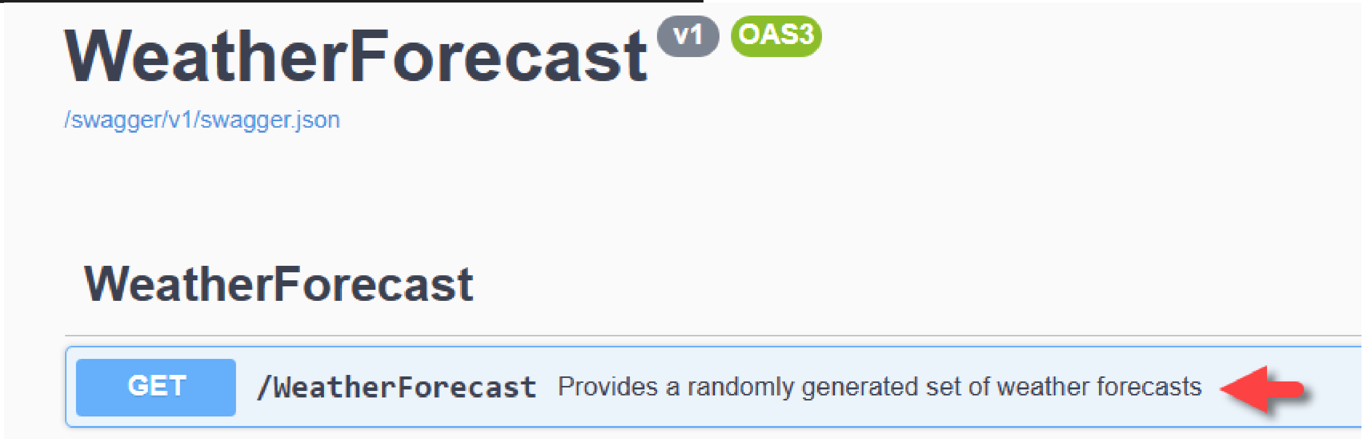
```
    builder.Services.AddSwaggerGen(setupAction: c =>
    {
        // Add nice title
        c.SwaggerDoc(name: "v1", info: new OpenApiInfo { Title = "Weather Forecast", Version = "v1" })

        // Add documentation via C# XML Comments
        var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
        var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);
        c.IncludeXmlComments(filePath: xmlPath);
    });
```

Adding REST Documentation

- XML Comments Example



```
/// <summary>
/// Provides a randomly generated set of weather forecasts
/// </summary>
/// <returns>A list of weather forecasts</returns>
/// <remarks>
/// Sample request:
///
/// GET /weatherforecast
///
/// </remarks>
/// <response code="200">Indicates the request was successful</response>
[HttpGet(Name = "GetWeatherForecast")]
0 references | 0 changes | 0 authors, 0 changes
public IEnumerable<WeatherForecast> Get()
{
```



WeatherForecast v1 OAS3

/swagger/v1/swagger.json

WeatherForecast

GET /WeatherForecast Provides a randomly generated set of weather forecasts

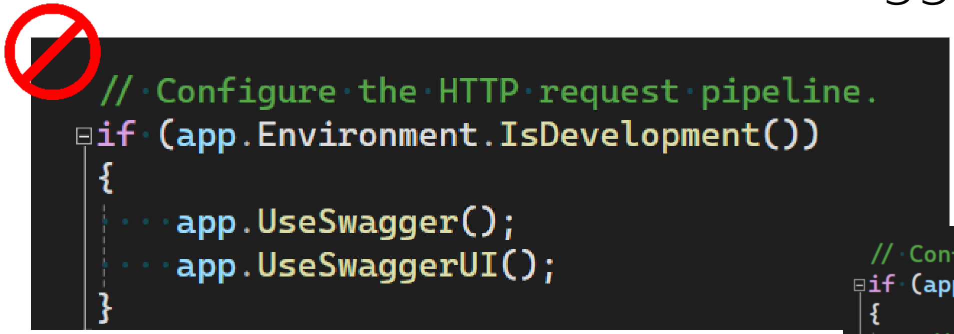
Adding REST Documentation

Swagger Initialization code

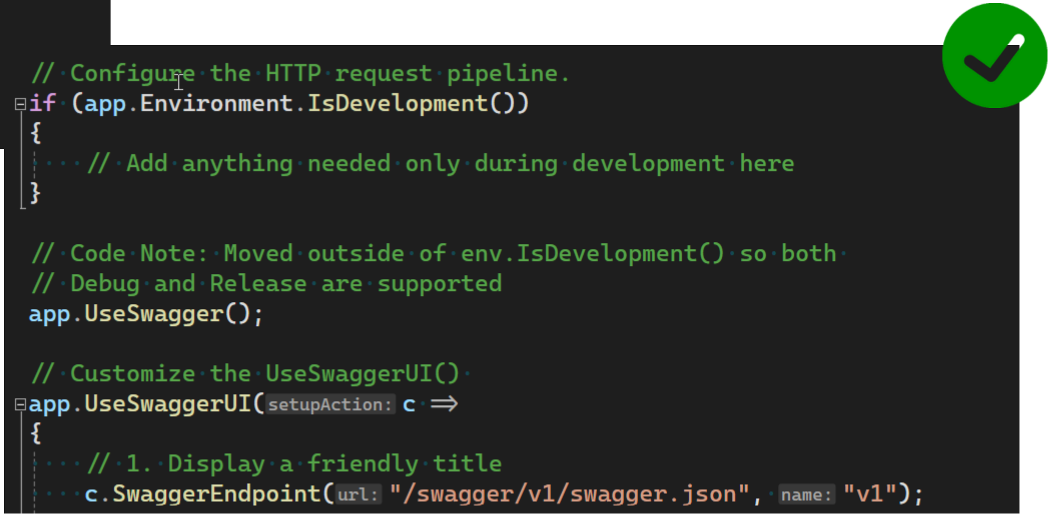
```
…
// Code Note: Moved outside of env.IsDevelopment() so both
// Debug and Release are supported
app.UseSwagger();

// Customize the UseSwaggerUI()
app.UseSwaggerUI(c =>
{
    // 1. Display a friendly title
    c.SwaggerEndpoint("/swagger/v1/swagger.json", "v1");

    // Code Note:
    // Launch the Swagger UI by default
    // Serving the Swagger UI at the app's root
    // (http://localhost:<port>)
    c.RoutePrefix = string.Empty;
});
…
```

Adding REST Documentation

- Don't forget to move out of **`IsDevelopment()`**
  - Why?
    - Won't see swagger UI when deployed to Azure



```
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
```



```
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    // Add anything needed only during development here
}

// Code Note: Moved outside of env.IsDevelopment() so both
// Debug and Release are supported
app.UseSwagger();

// Customize the UseSwaggerUI()
app.UseSwaggerUI(setupAction: c =>
{
    // 1. Display a friendly title
    c.SwaggerEndpoint(url: "/swagger/v1/swagger.json", name: "v1");
```

Demo

Adding REST Documentation

```
Extending WeatherForecast API App with Swagger Doc
WeatherForecastSolution.sln
  WeatherForecast.csproj
```

Agenda

- Essential REST Verbs
  - GET
    - Retrieve a single item by Id
    - Retrieve a list of items when no Id is provided
  - POST
    - Create resource, server generates Id
  - PUT
    - Update a resource by replace its content
    - Create a resource using Id provided by caller

Agenda

- Essential REST Verbs
  - PATCH
    - Update a resource by replace parts of its content
  - DELETE
    - Delete a resource

Demo

Essential REST

```
GET, POST, PUT, PATCH, DELETE
WeatherForecastTestingSolution.sln
  WeatherTestingForecast.csproj
```

Questions

Best Practices

- Be stateless
- Be asynchronous
  - Execute I/O operations on non request thread
- Measure then optimize
- Cache as close to the wire as possible
  - Think carefully about your caching policy
- Servers shall be expendable
  - **They will fail, plan for it in your design**

Further Reading

- Azure for Developers: 3rd Edition
  - Author: Kamil Mrzygłód
  - ISBN: 978-1836203513
  - Chapter 1

Further Reading

- Optional Book: C# 14 and .NET 10 – Modern Cross-Platform Development Fundamentals
  - Chapter 15 Building and Consuming Web Services
  - Author: Mark J. Price
  - ISBN: 978-1836206637

Further Reading

- Building Cloud Apps with Microsoft Azure
  - Authors: Scott Guthrie, Mark Simms, Tom Dkystra, Rick Anderson, Mike Wasson
  - ASIN: B00LXAAMSG
  - Chapters: 4, 9, 11

Links

- Azure App Services (API and Web)
  - App Service documentation
- ASP.NET Core
  - ASP.NET documentation
- Create a web API with ASP.NET Core and Visual Studio for Windows
  - Tutorial: Create a web API with ASP.NET Core
  - Generate OpenAPI documents | Microsoft Learn

Links

- Publish an ASP.NET Core app to Azure with Visual Studio
- Publish an ASP.NET Core app to Azure with Visual Studio Code
- Publish an ASP.NET Core web app with CLI tools

Links

- Visual Studio 2026
  - Built in support for .http files
- Visual Studio Code Extensions
  - Azure App Service
  - Azure Developer CLI
  - Azure Resources
  - Azure Tools
  - Bicep
  - C#
  - C# Dev Kit
  - REST Client

REST Utilities

- Postman
  - Make REST calls from a richly featured UI
    - https://www.getpostman.com/
- Nightengale
    - https://nightingale.rest/
- cURL
  - Included in Windows 11
  - Rest calls from the command line

REST Utilities

- Fiddler
  - Make REST Calls
  - Examine / Debug request/response
  - http://www.telerik.com/fiddler
- Firefox extension:
  - RESTClient