Azure App Services Part I

CSCI E-94

Fundamentals of Cloud Computing - Azure

Joseph Ficara

Copyright © 2013-2025

Agenda

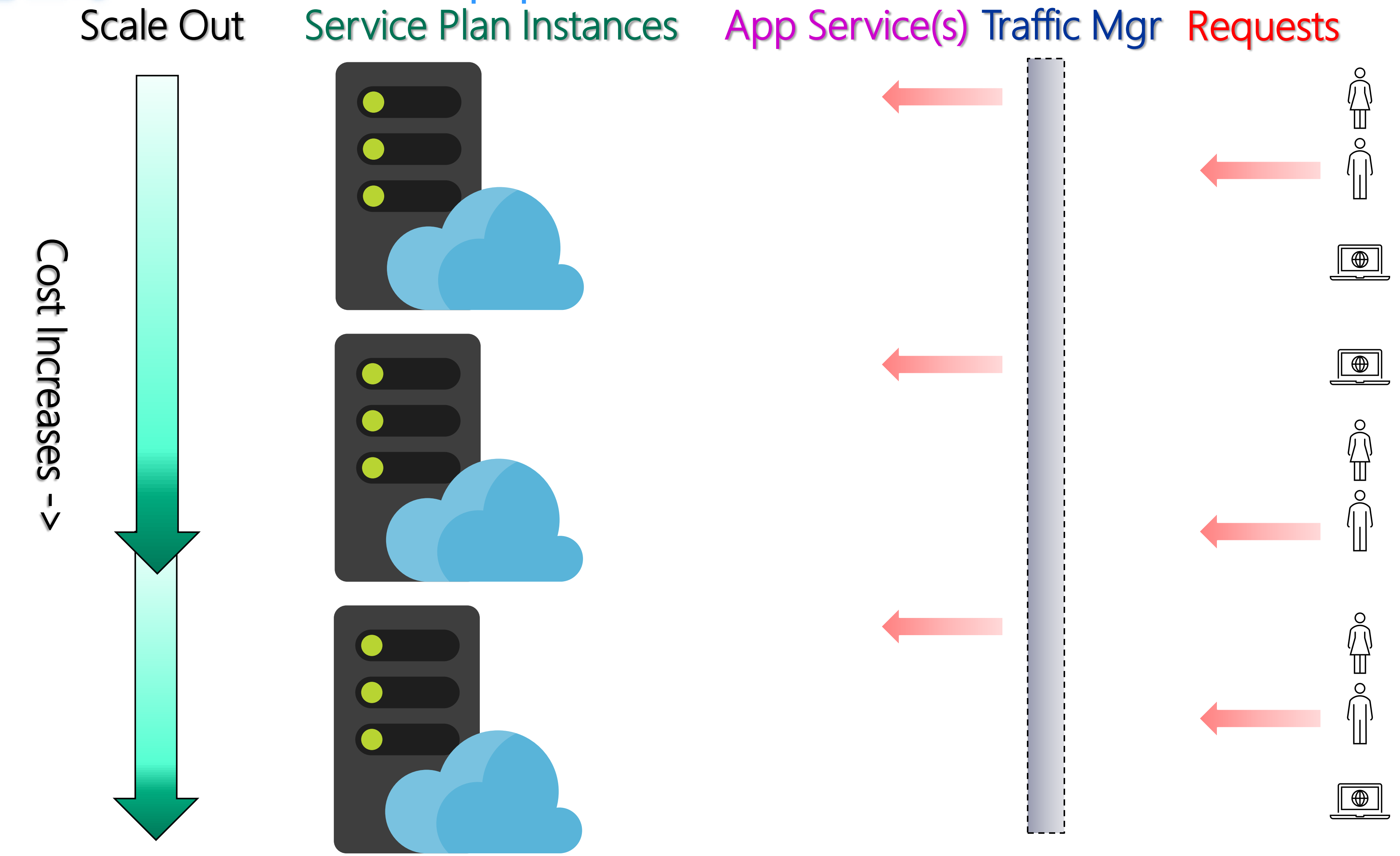
- Azure App Service
  - Overview
    - What is it?
    - Why do you care?
    - Essentials
  - ASP.NET Core Essentials
  - Weather Azure App Service
    - Supporting REST Documentation (Swagger)
  - Publishing to Azure

Overview

Azure App Service – Overview

- What is an Azure App Service?
  - A fully managed "VM"
    - With an HTTP server
    - Fully locked down
  - A place to host your
    - REST API and/or Website
  - A place to run
    - Background jobs (Webjobs)
  - What about the "server"?
    - It's defined by an App Service Plan

Azure App Service – Overview



Azure App Service – Overview

- Why do I care ?
  - Excellent for rich client applications
    - Web apps that rely on AJAX
      - Vue.js
      - KnockoutJS
      - Jquery
      - Etc…
    - Single Page Applications (SPA)
      - AngularJS
      - ReactJS
      - Etc…

Azure App Service – Overview

Why do I care ? …

- Excellent for clients needing cloud services
  - Internet of things IoT
  - Progressive Web Apps (PWA)
  - OS X applications
  - MAUI & Win32 Applications
  - Linux applications
  - Clients that need to use REST
- For gRPC support: Requires Linux
  - gRPC on .NET supported platforms
  - Configure gRPC on App Service

Azure App Service – Overview

Why do I care ? …

- SLA and Cost - App Service
  - Uptime
    - As of 10/15/2024 (Basic Tier and Above)
      - 99.95% -> App Services 100% Credit
      - See [SLA for App Service]
  - Cost
    - Check out the Azure calculator
      - See: [http://bit.ly/1Mc1JxQ]
      - Roughly – Configured for the Uptime above…

        Windows **$54.75** USD Per month

        Linux: **$12.41** USD Per Month

Azure App Service – Overview

Essentials

- Platform as a Service offering
  - Operating system security updates
    - Automatically handled
  - Fully managed
- Supports Windows & Linux
- Built in
  - Authentication support
  - Load balancing & Auto-scaling
  - Built in traffic manager

Azure App Service – Overview

Essentials …

- Custom Domain & SSL Certs
- Language agnostic
  - .NET / .NET Core
  - Node.JS
  - PHP
  - Java
  - Python
  - Ruby

Azure App Service – Overview

Essentials …

- DevOps
  - Continuous Deployment
    - Azure DevOps
    - GitHub
    - Docker Hub
    - Others …
  - Deployment slots facilitate
    - Live updating & rollback
    - Testing in production
    - Light weight A/B Testing

Azure App Service – Overview

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
    - Twitter
    - Azure Active Directory
    - …

ASP.NET Core – Essentials

Supports several authentication schemes …

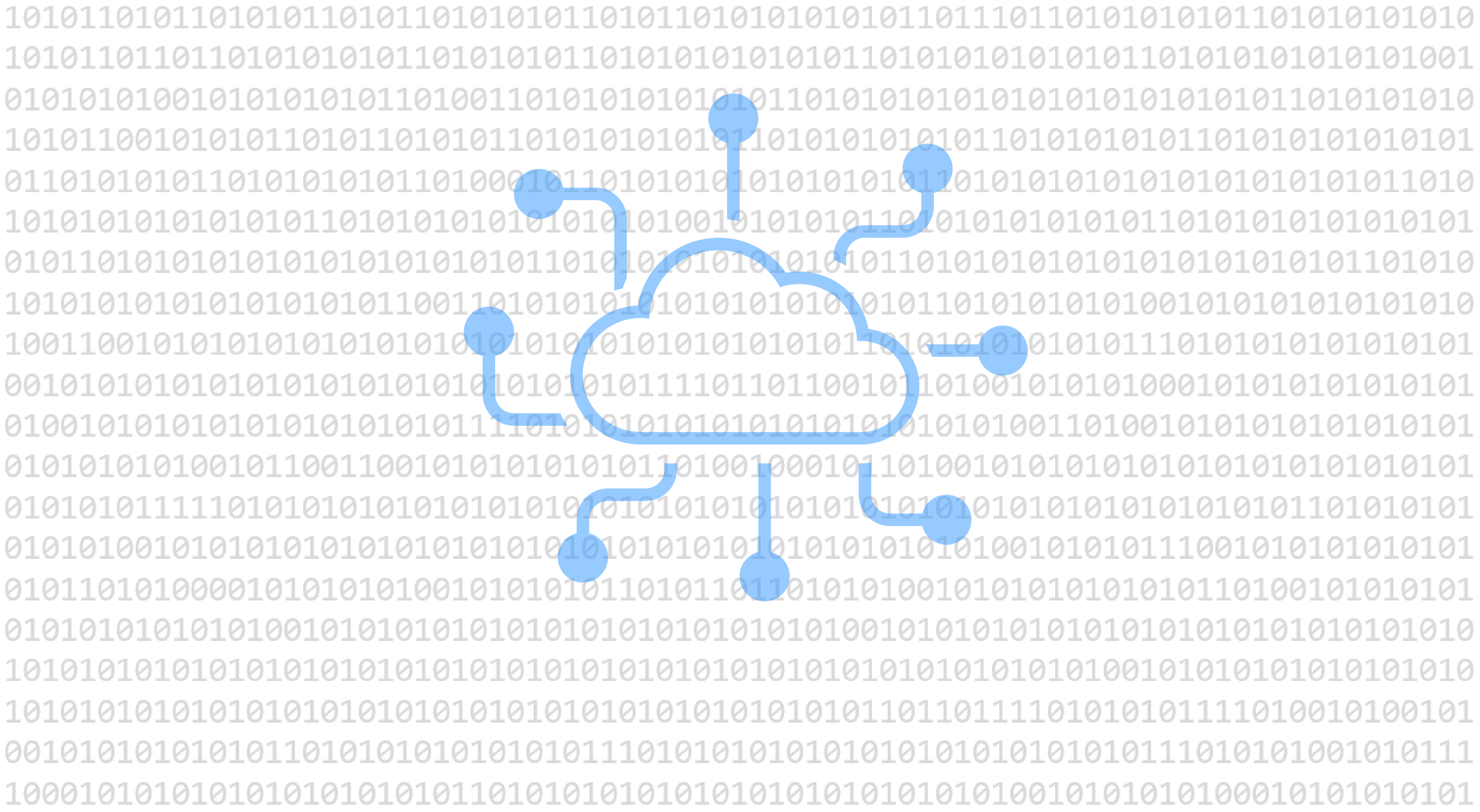
- Azure Active Directory B2C
- Basic Authentication
- Forms Authentication
- Integrated Windows Authentication
- OAUTH 2.0
- OpenID Connect
- …

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

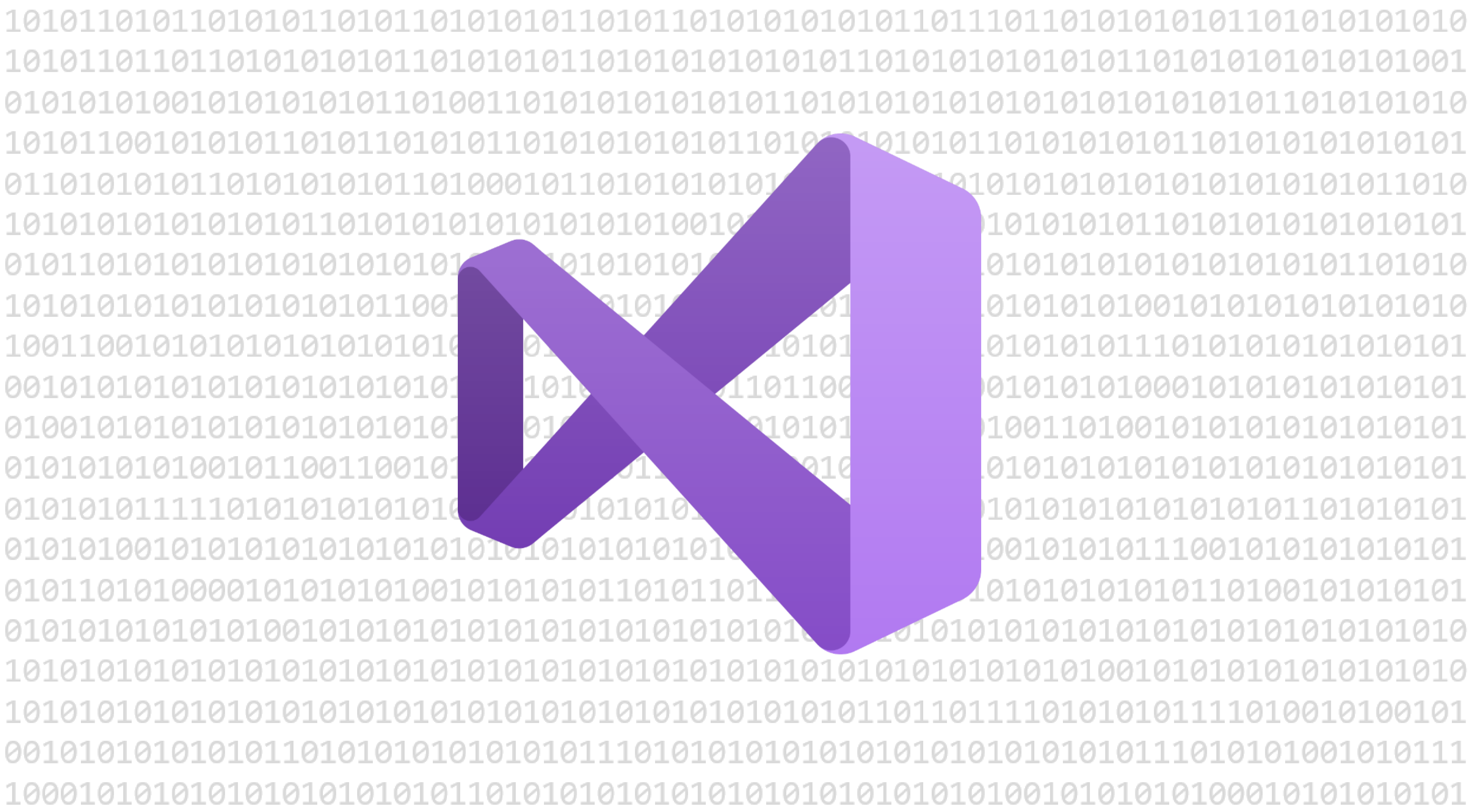
Let's Code!



Weather Azure App Service

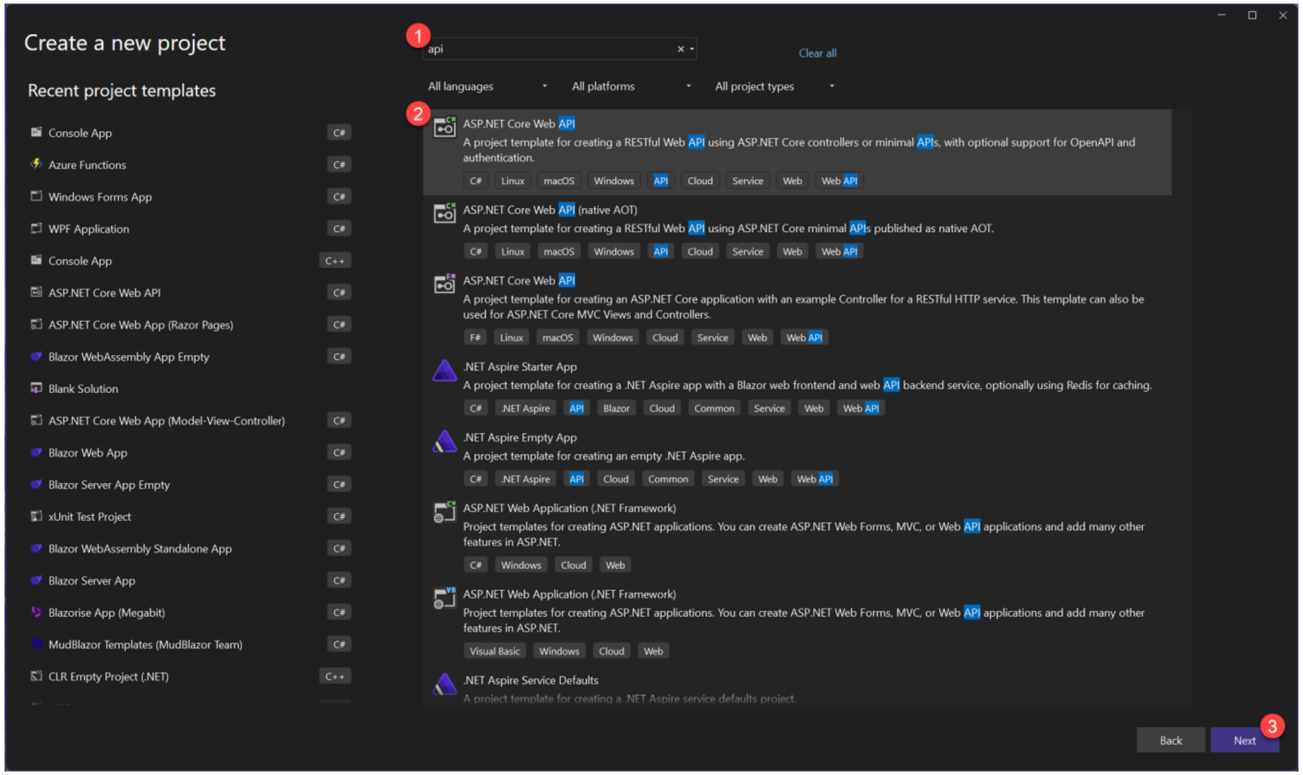
- Several templates available for .NET 9
  - ASP.NET Core Empty
  - ASP.NET Core Web App
  - ASP.NET Core Web API
  - ASP.NET Core Web API (native AOT)
  - ASP.NET Core Web App (Model-View-Controller)
  - ASP.NET Core gRPC Service
  - ASP.NET Core Web App (Razor Pages)
  - …

Visual Studio 2022



Weather Azure App Service

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



Create a new project

Recent project templates

Console App	C#
Azure Functions	C#
Windows Forms App	C#
WPF Application	C#
Console App	C++
ASP.NET Core Web API	C#
ASP.NET Core Web App (Razor Pages)	C#
Blazor WebAssembly App Empty	C#
Blank Solution	
ASP.NET Core Web App (Model-View-Controller)	C#
Blazor Web App	C#
Blazor Server App Empty	C#
xUnit Test Project	C#
Blazor WebAssembly Standalone App	C#
Blazor Server App	C#
Blazorise App (Megabit)	C#
MudBlazor Templates (MudBlazor Team)	C#
CLR Empty Project (.NET)	C++

1 api

Clear all

All languages · All platforms · All project types

2

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.
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 with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
F# · Linux · macOS · Windows · Cloud · Service · Web · Web API

.NET Aspire Starter App
A project template for creating a .NET Aspire app with a Blazor web frontend and web API backend service, optionally using Redis for caching.
C# · .NET Aspire · API · Blazor · Cloud · Common · Service · Web · Web API

.NET Aspire Empty App
A project template for creating an empty .NET Aspire app.
C# · .NET Aspire · API · Cloud · Common · Service · Web · Web API

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
C# · Windows · Cloud · Web

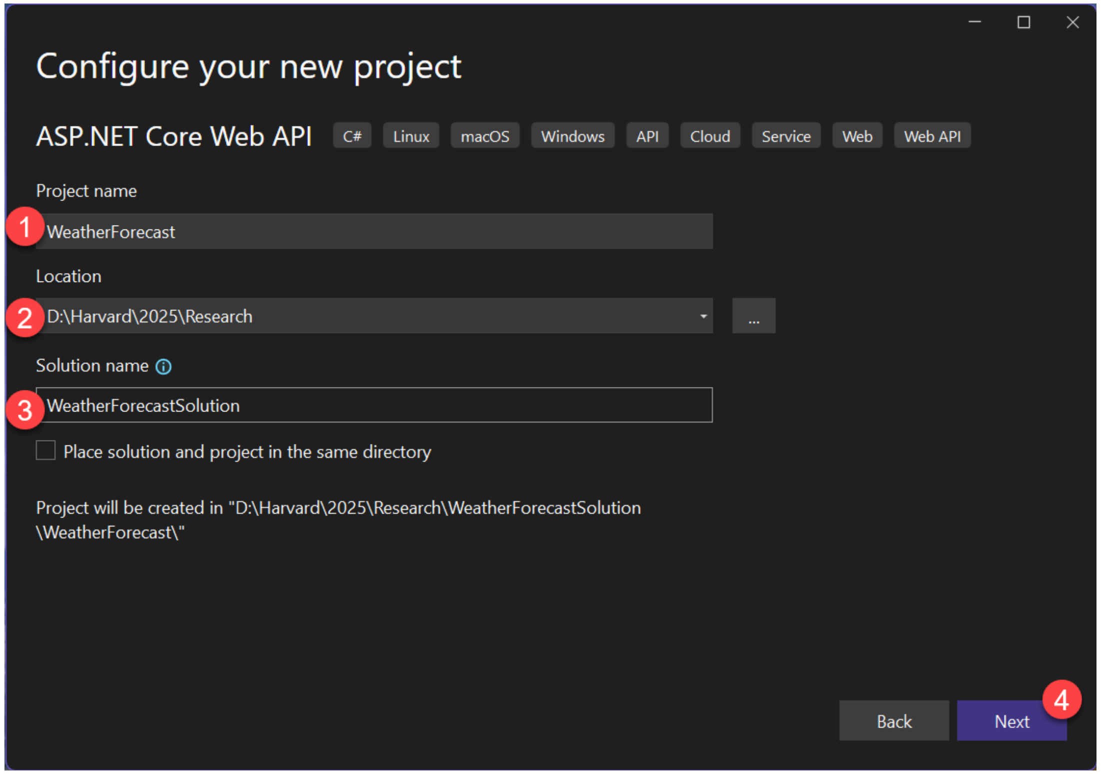
ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
Visual Basic · Windows · Cloud · Web

.NET Aspire Service Defaults
A project template for creating a .NET Aspire service defaults project.

Back · Next 3

Weather Azure App Service

- Additional Information
  - Name your project and solution



Configure your new project

ASP.NET Core Web API

Project name: WeatherForecast

Location: D:\Harvard\2025\Research

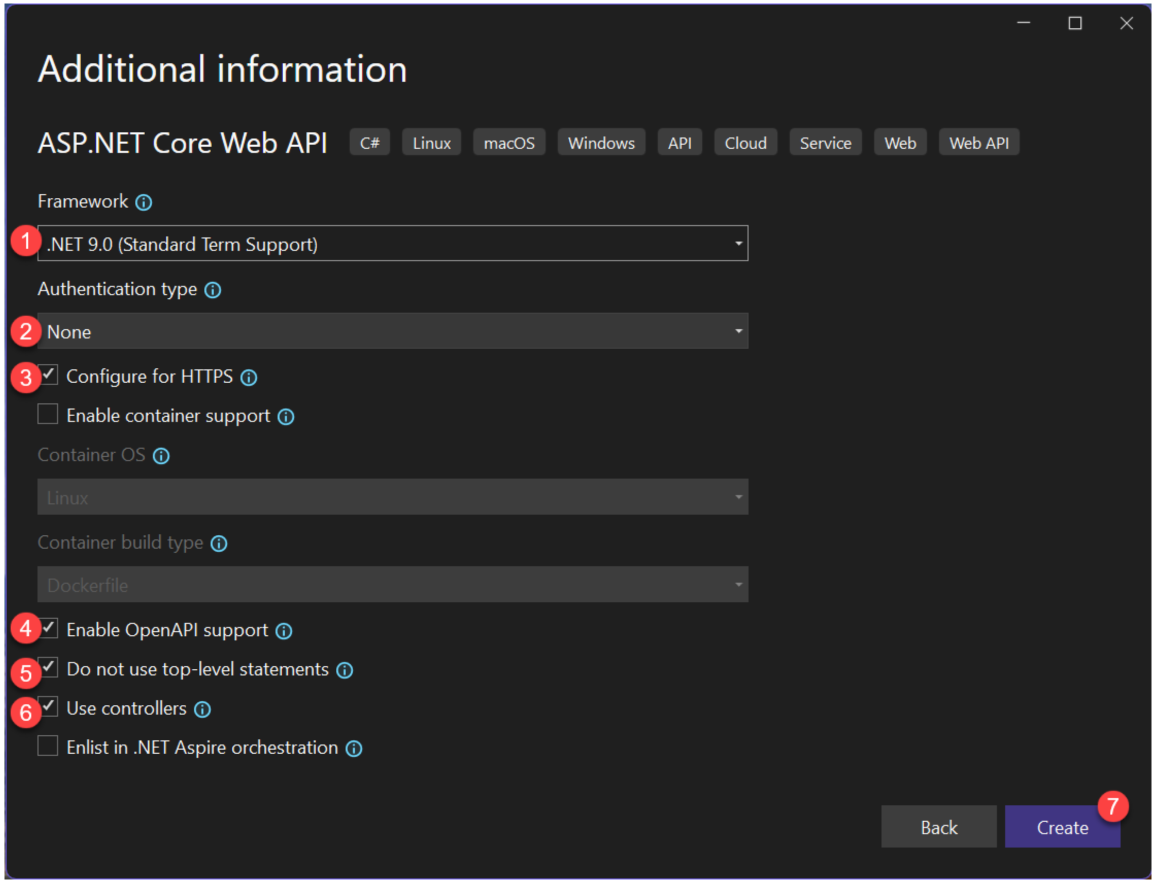
Solution name: WeatherForecastSolution

[ ] Place solution and project in the same directory

Project will be created in "D:\Harvard\2025\Research\WeatherForecastSolution\WeatherForecast\"

Weather Azure App Service

- Additional Information



Additional information

ASP.NET Core Web API

Framework

1 .NET 9.0 (Standard Term Support)

Authentication type

2 None

3 [✓] Configure for HTTPS

[ ] Enable container support

Container OS

Linux

Container build type

Dockerfile

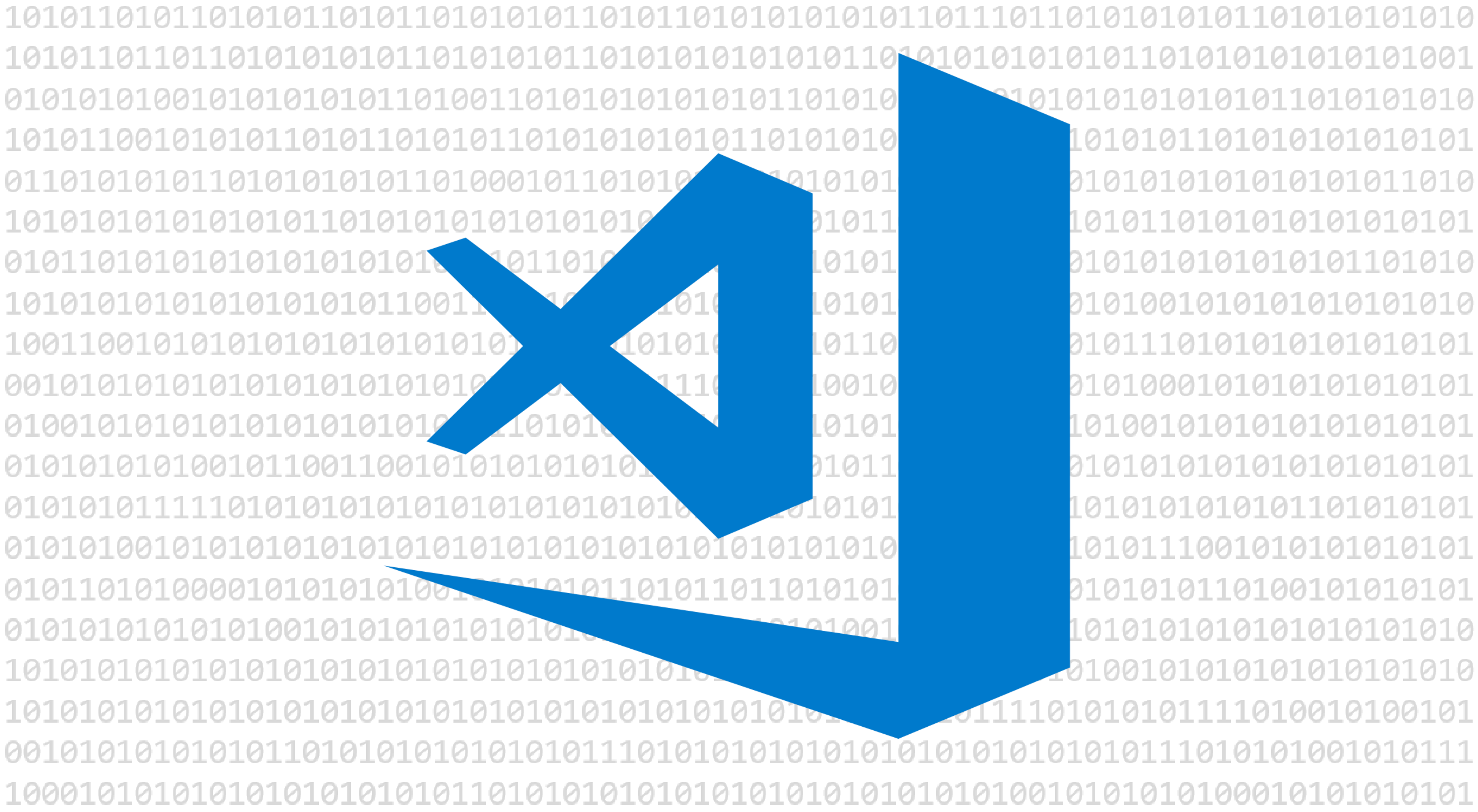
4 [✓] Enable OpenAPI support

5 [✓] Do not use top-level statements

6 [✓] Use controllers

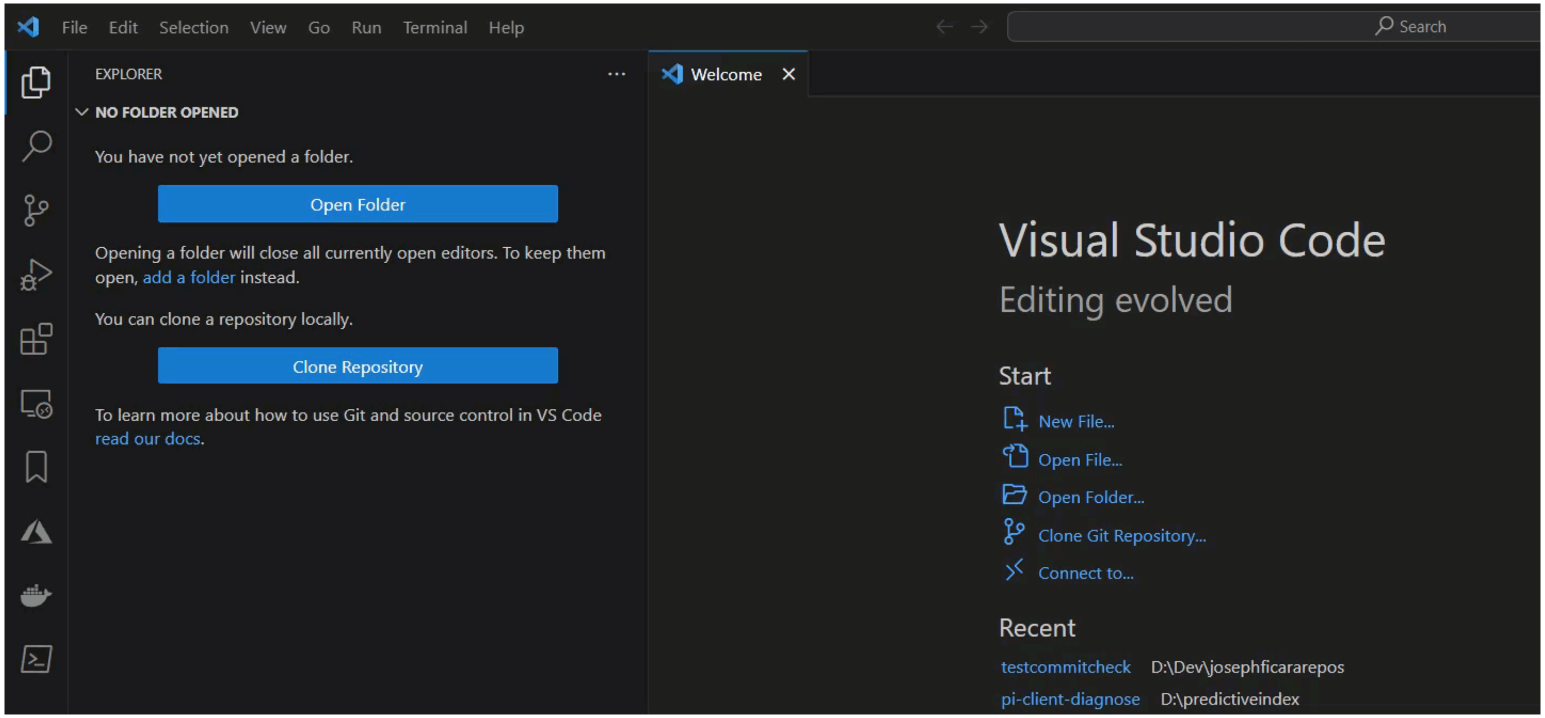
[ ] Enlist in .NET Aspire orchestration

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

Start

New File...

Open File...

Open Folder...

Clone Git Repository...

Connect to...

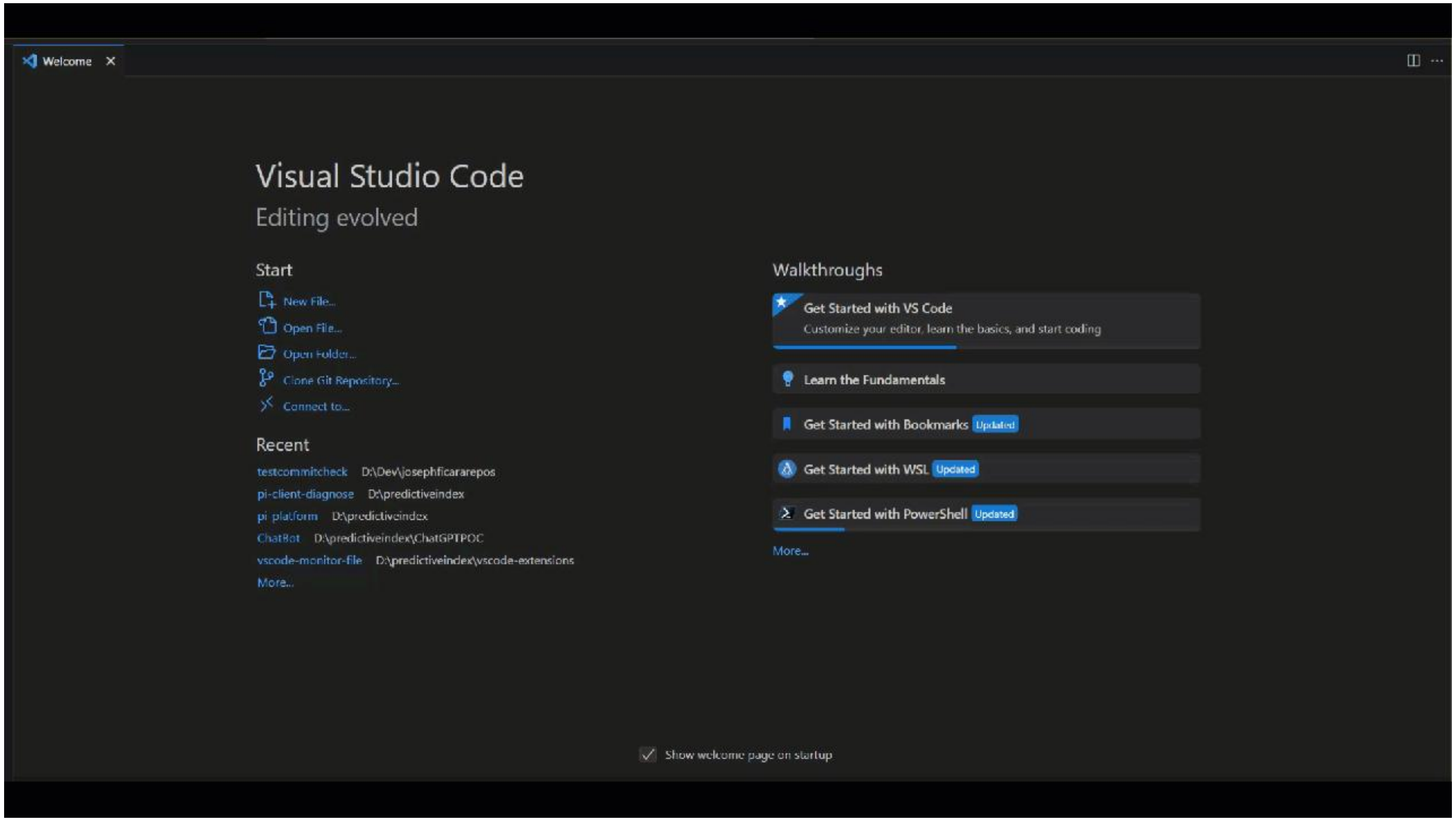
Recent

testcommitcheck D:\Dev\josephficararepos

pi-client-diagnose D:\predictiveindex

Weather Azure App Service
VSCode

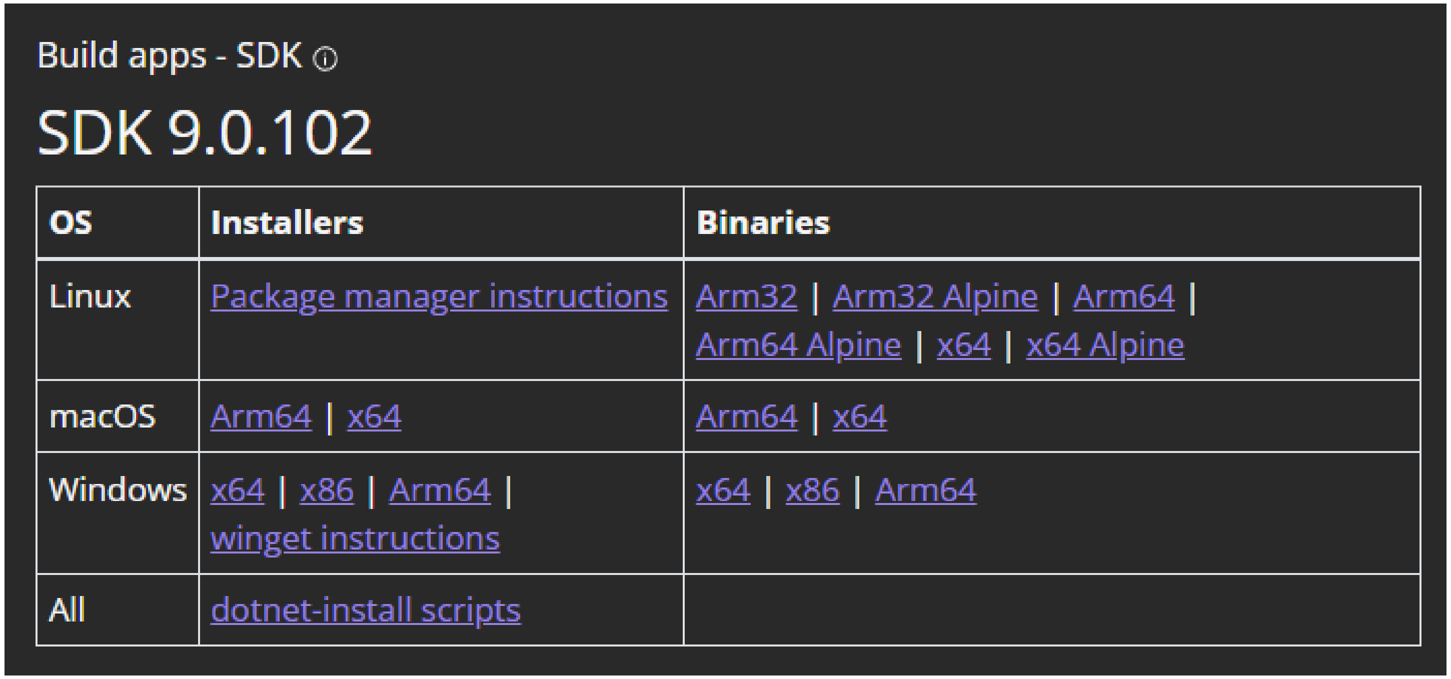
- Follow steps Getting Started with C# Dev Kit



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

Weather Azure App Service
VSCode

- .NET 9 SDK Download & Install
  - Download .NET 9.0



Build apps - SDK ⓘ

SDK 9.0.102

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

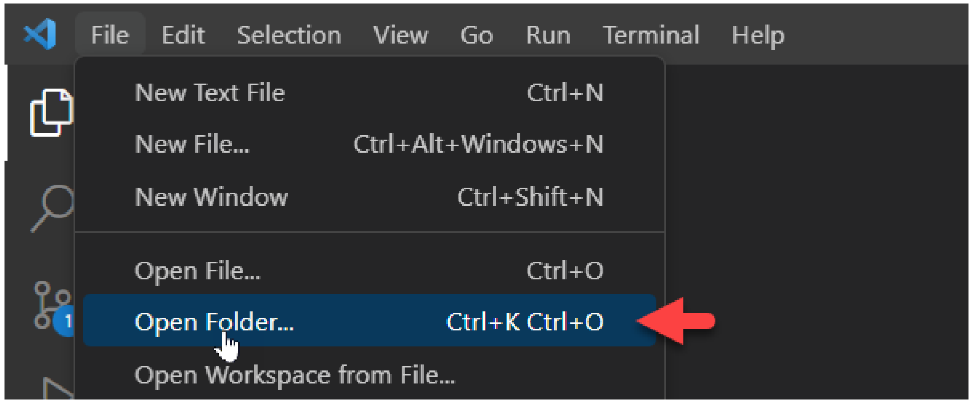
- Verify .NET 9 SDK is Installed

```
Joseph  ~  ✓                                                    in pwsh at 16:49:17
```

Weather Azure App Service

VSCode

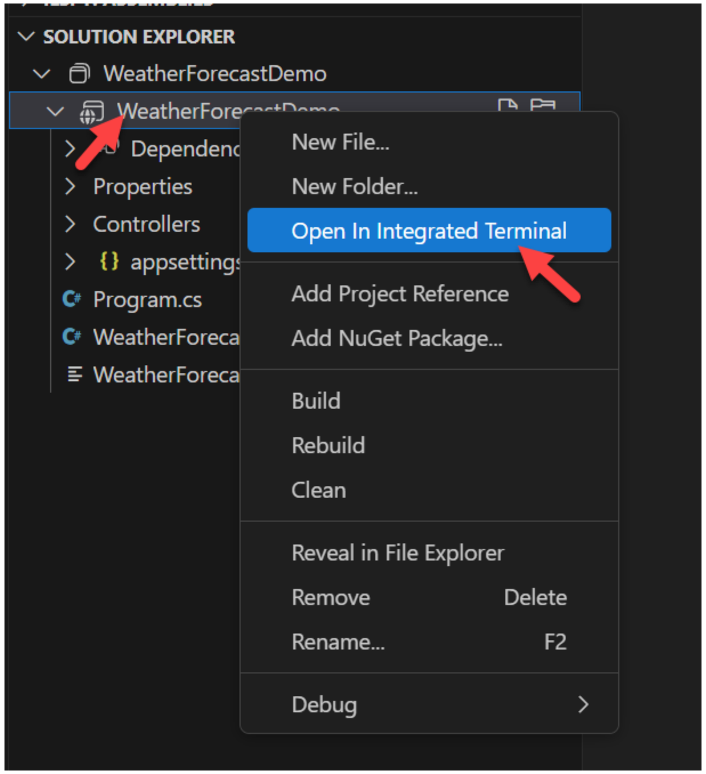
- Select an empty folder in VS Code



- Use the command line to create your project
  - dotnet new webapi --framework net9.0 --use-controllers --use-program-main -n <project name>

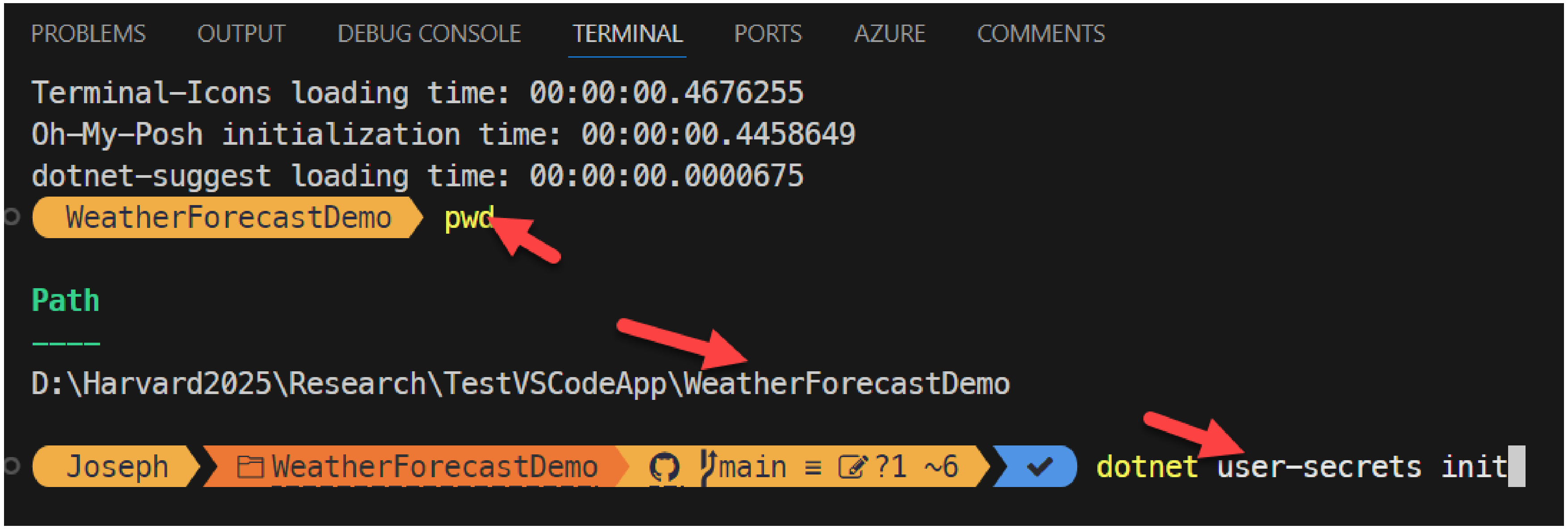
Weather Azure App Service
VSCode

- 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**

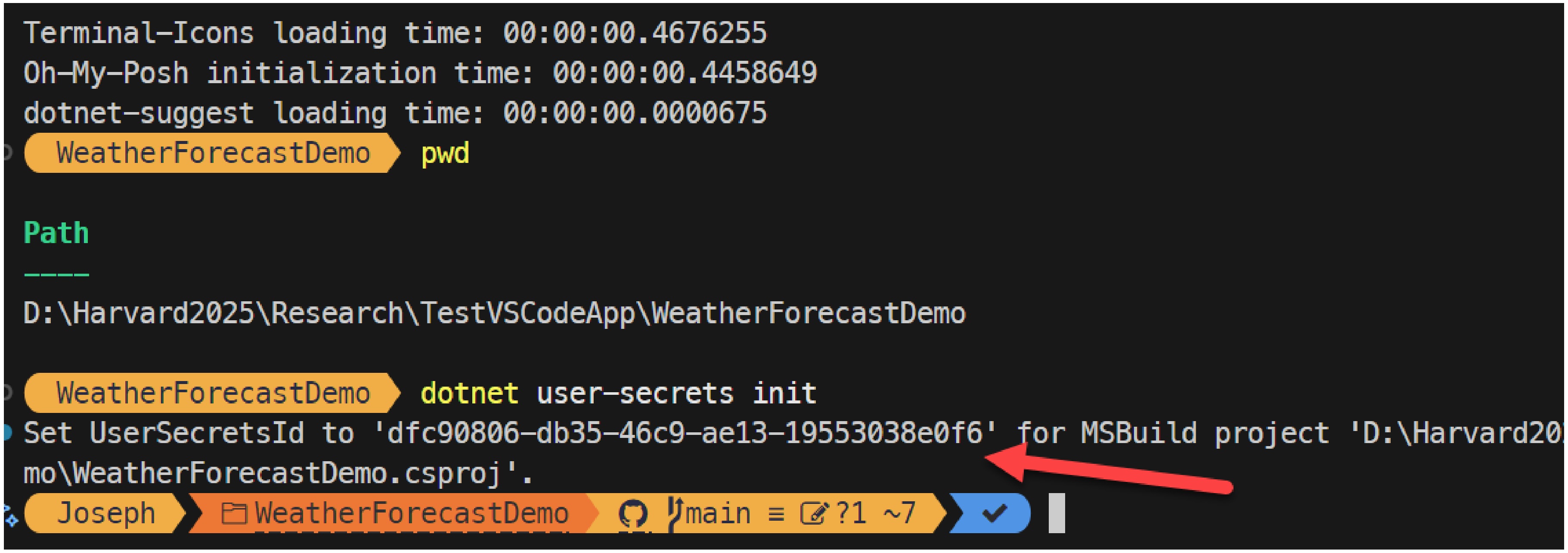


```
Terminal-Icons loading time: 00:00:00.4676255
Oh-My-Posh initialization time: 00:00:00.4458649
dotnet-suggest loading time: 00:00:00.0000675
 WeatherForecastDemo  pwd

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

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

Weather Azure App Service

VSCode

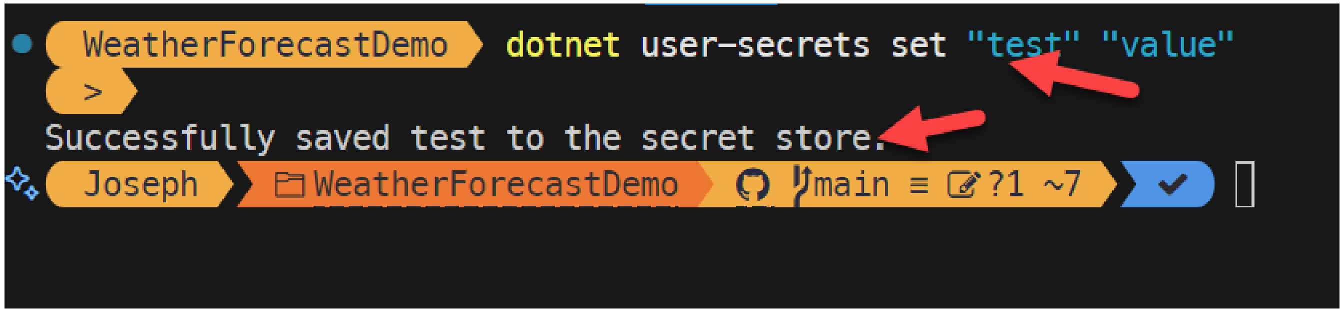
- Generate your **secrets.json** file …
  - Result should look like this



```
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

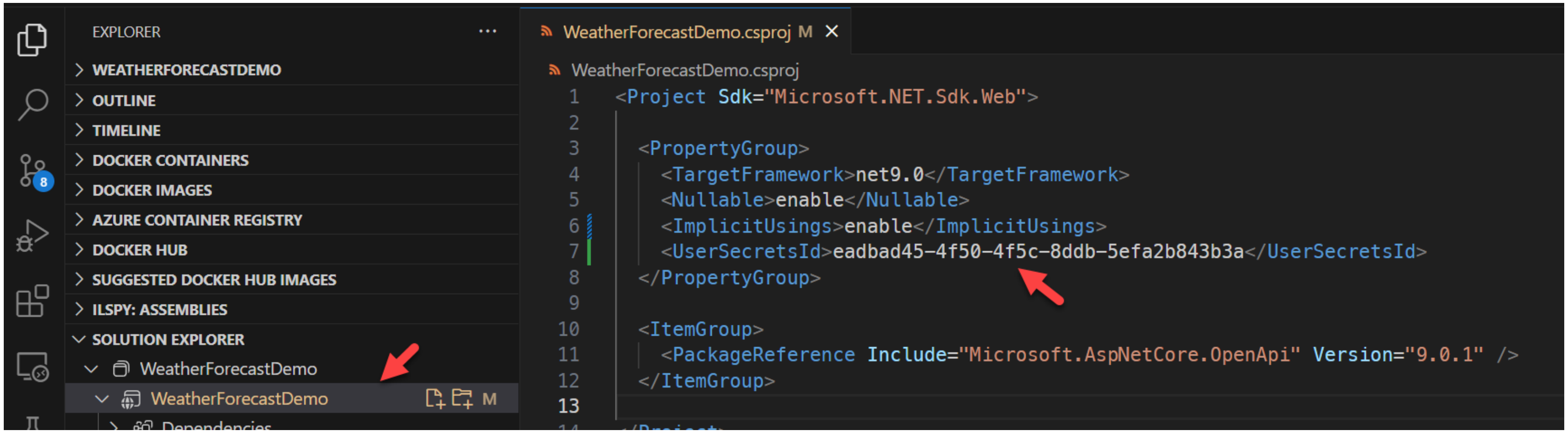


```
WeatherForecastDemo  dotnet user-secrets set "test" "value"
>
Successfully saved test to the secret store.
Joseph  WeatherForecastDemo  main ≡ ?1 ~7  ✔
```

Weather Azure App Service

VSCode

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

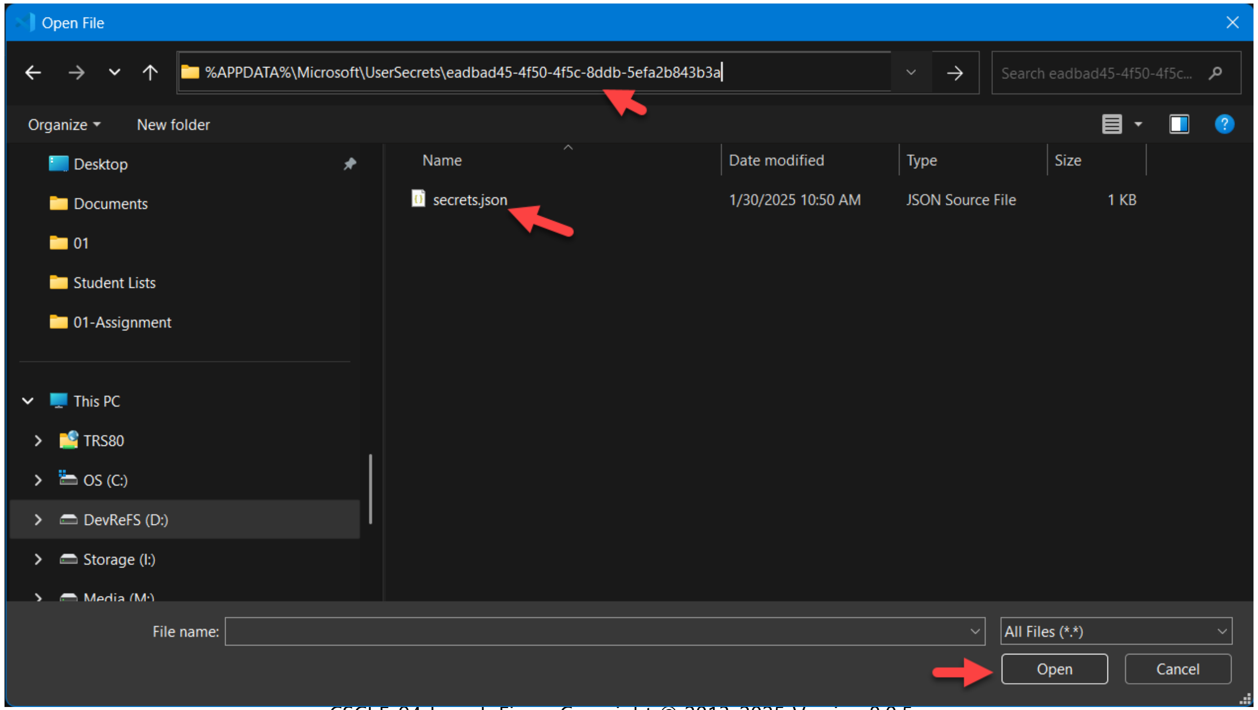


```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <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>

```

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


Open File

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

Search eadbad45-4f50-4f5c...

Organize | New folder

Desktop
Documents
01
Student Lists
01-Assignment
This PC
TRS80
OS (C:)
DevReFS (D:)
Storage (I:)
Media (M:)

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

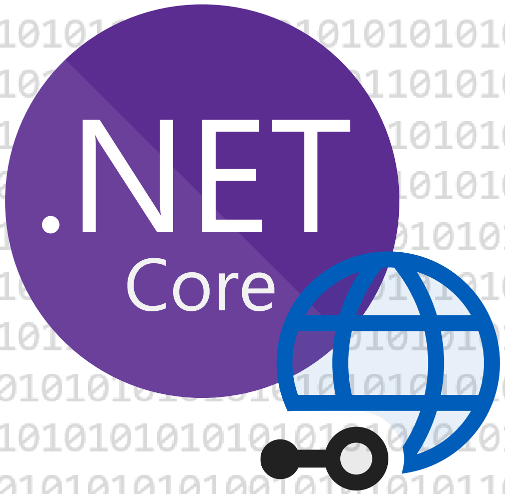
File name: | All Files (*.*)

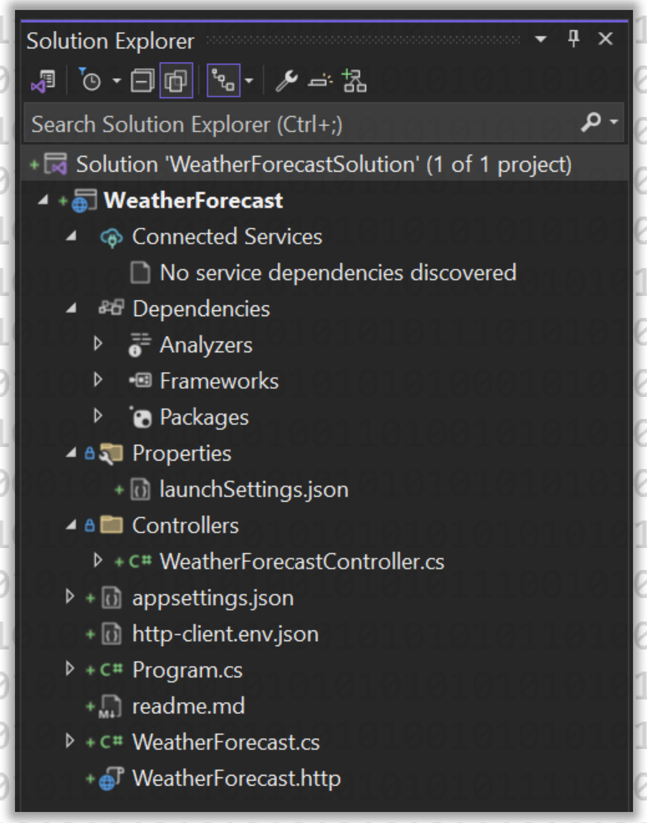
Open | Cancel

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)
  WeatherForecast
    Connected Services
      No service dependencies discovered
    Dependencies
      Analyzers
      Frameworks
      Packages
    Properties
      launchSettings.json
    Controllers
      WeatherForecastController.cs
    appsettings.json
    http-client.env.json
    Program.cs
    readme.md
    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 …

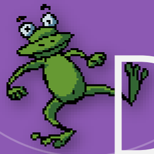
- Project structure …
  - Controllers
    - Classes that handle HTTP requests go here
    - Http Verbs automatically routed to methods
      - HTTP Verb GET routes to a method called Get()
    - *Clearer to use the C# Attribute* [HttpGet]
    - Controllers/WeatherForcastController.cs
      - Sample code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



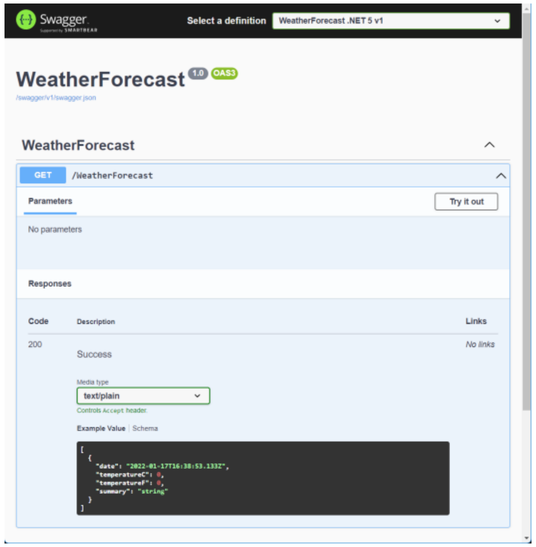
Demo

ASP.NET Core API Template Example

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

Adding REST Documentation

- Web API supports documentation
  - UI Needs to be added manually
    - Via Swashbuckle nuget package
      `Swashbuckle.AspNetCore (7.x)`



WeatherForecast

/swagger/v1/swagger.json

WeatherForecast

GET /WeatherForecast

Parameters

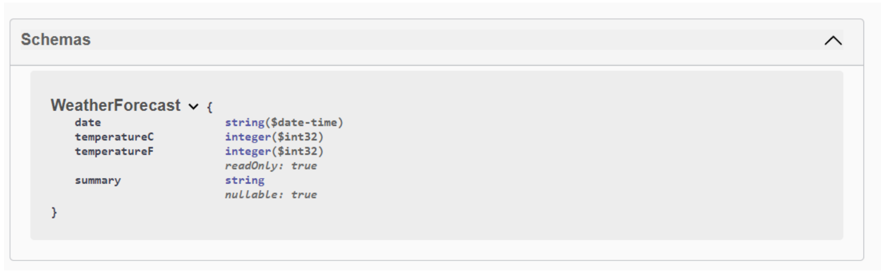
No parameters

Responses

Code	Description	Links
200	Success	*No links*

Controls Accept header.

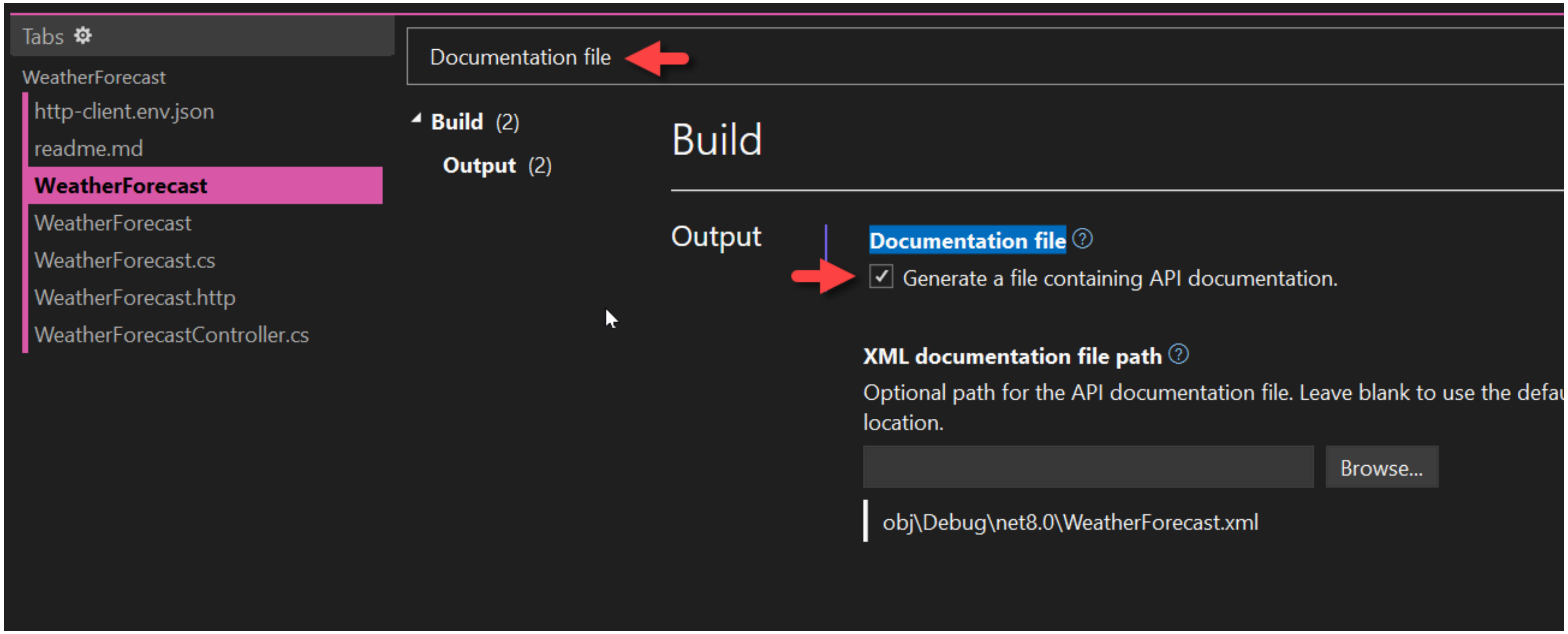
```
[
  {
    "date": "2022-01-17T16:38:53.133Z",
    "temperatureC": 0,
    "temperatureF": 0,
    "summary": "string"
  }
]
```



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

Adding REST Documentation

- Generate suppress warnings
  - Right click on the project and 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

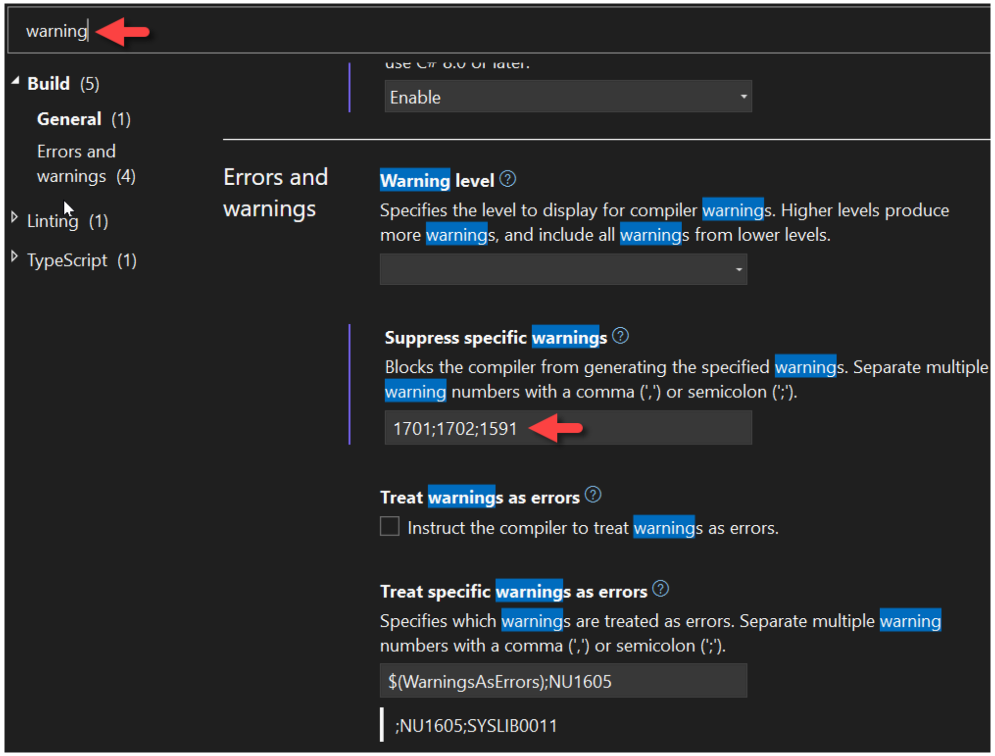
Optional path for the API documentation file. Leave blank to use the defau
location.

Browse...

obj\Debug\net8.0\WeatherForecast.xml

Adding REST Documentation

- Generate 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)

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

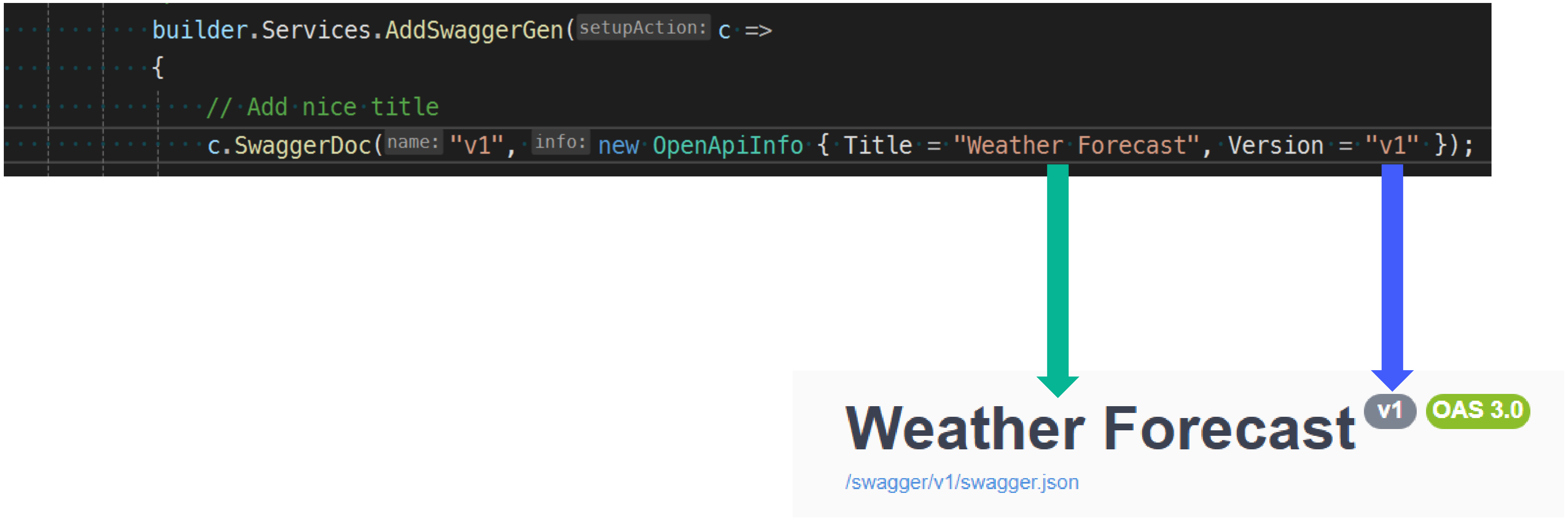
ASP.NET Core Web API App supports documentation …

Swagger Initialization code

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

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

// Learn more about configuring OpenAPI at https://aka.ms/aspnet/openapi
builder.Services.AddOpenApi();
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

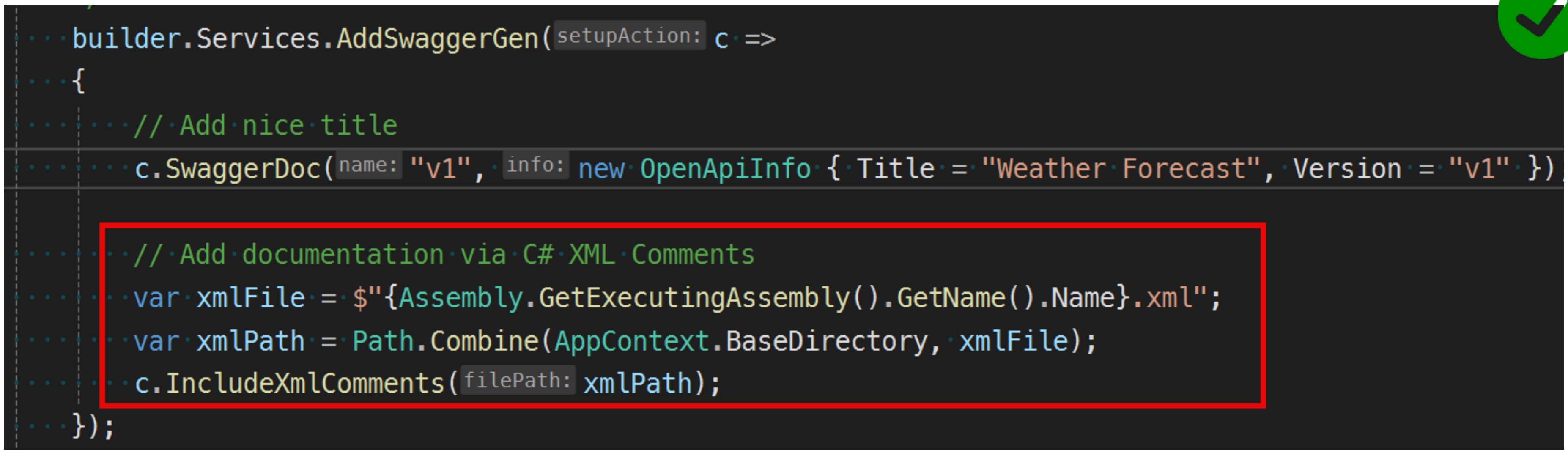
- Title and Version Example



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

Adding REST Documentation

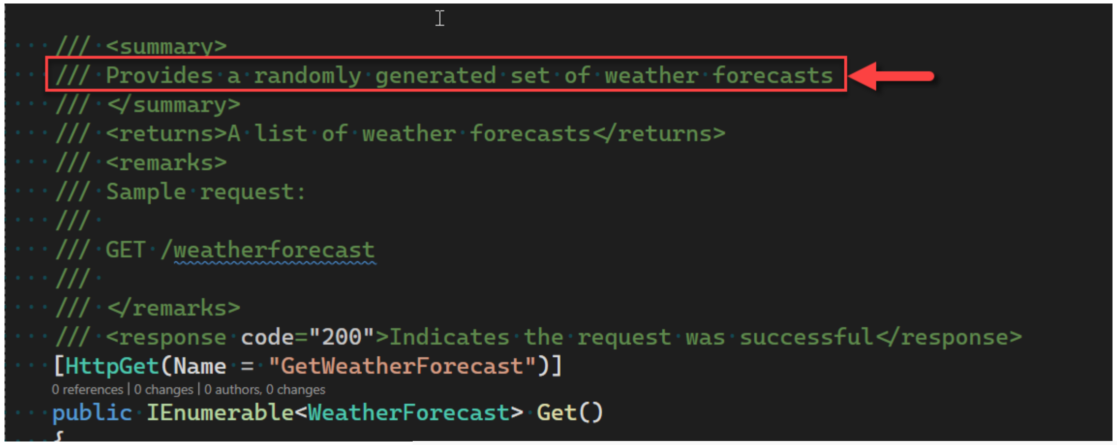
- Don't forget add the code to include the xml file
  - Why?
    - To see the C# XML API Comments, you added



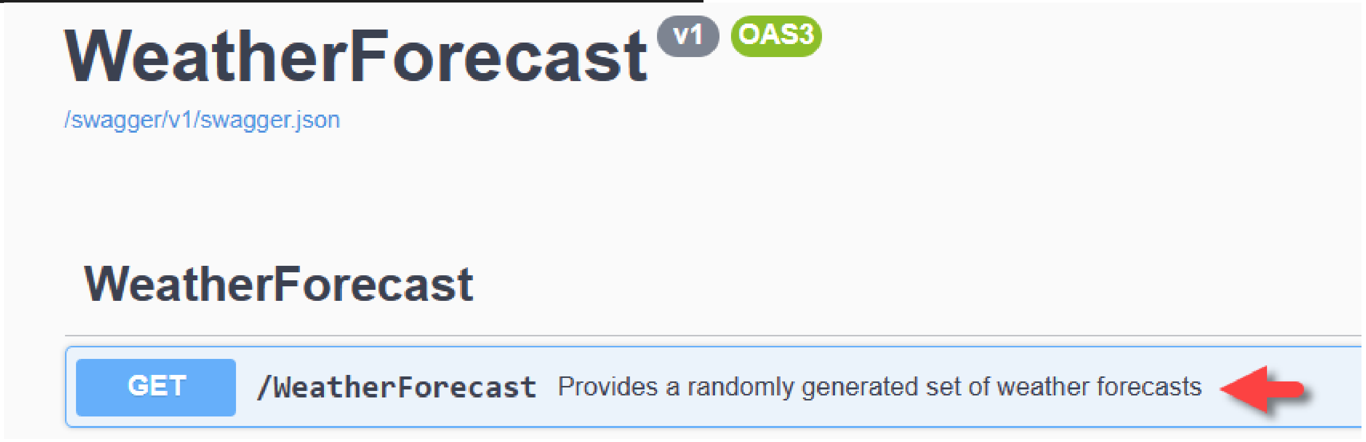
```
builder.Services.AddSwaggerGen(setupAction: c =>
{
    // Add nice title
    c.SwaggerDoc(name: "v1", info: new OpenApiInfo { Title = "Weather Forecast", Version = "v1" })

    // 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

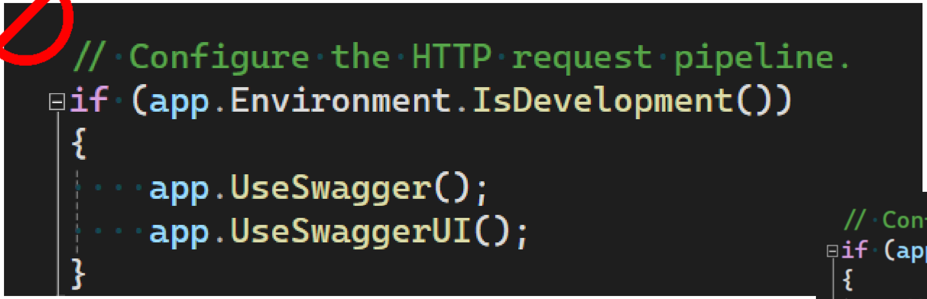
ASP.NET Core Web API App supports documentation …

Swagger Initialization code

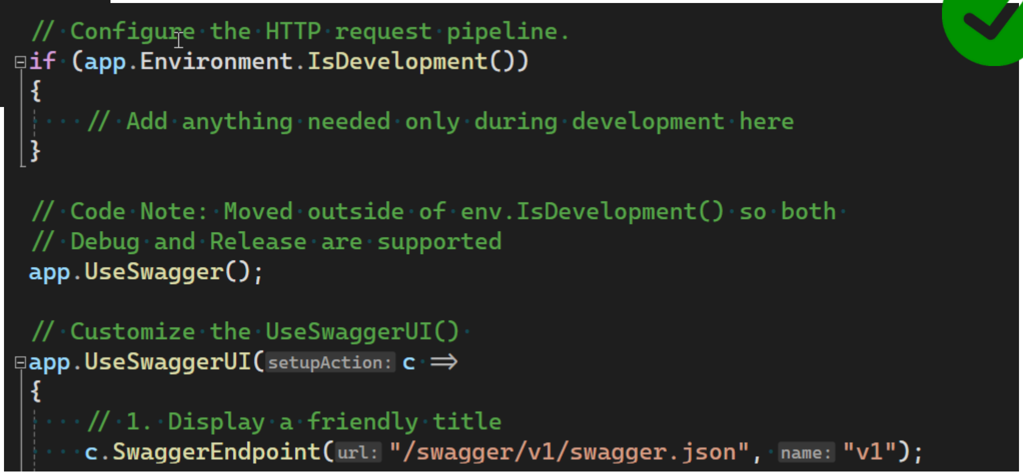
```
…
// Code Note: Moved outside of env.IsDevelopment() so both
// Debug and Release are supported
app.UseSwagger();

// 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
```

Publishing to Azure

Publishing to Azure

- There are two basic steps in publishing
  - Creating the Azure resources
    - Resource Group
    - App Service Plan
    - App Service
  - Publishing
    - The .NET Core app to the App Service
- Both can be done from Visual Studio
- Several other options …

Publishing to Azure

- Create the App Service using
  - Portal & PowerShell
  - Azure Resource Manager (ARM) Templates
  - Bicep
  - REST Interface
  - And more…
- Publish from
  - Git
  - CICD pipeline
  - And more …

Publishing to Azure

- Some key items in Azure
  - Resource groups [Manage Azure resources]
    - Collections of resources
      - Resources are services/resources in azure:
        - App Services
        - Azure SQL
        - Azure Storage
        - Etc…
    - Group associated services with the same lifetime
    - Useful to organize and manage resources
      - Note: Location of the resource group defines:
        - Where metadata resides, not where resources reside

Publishing to Azure

Some key items in Azure …

- App Service Plan
  Azure App Service plan overview
  - This is **what you are paying** for
    - Defines size of the "**server**" used for app services
      - Run many app services on the same **app service plan**
        - **Load them up!**
        - Billed by the hour **on app service plan** *instances*
          - **Not the app service**
          - **Not the VMs running in the instance(s)**
      - Scale **up** the app Is Development service plan:
        - **Bigger** "server"
      - Scale **out** the service plan:
        - **More** "server instances"

Publishing to Azure

Some key items in Azure …

- App Service
  App Service documentation
  - PaaS instance software is deployed to
    - Runs your code
  - Code is isolated from other App Services
    - In the same app service plan
    - Not isolated from noisy neighbors

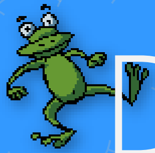
Publishing to Azure

Some key items in Azure …

- Storage Account
  Azure Storage documentation
  - Resource used for storage
    - Blobs, Files, Tables, Queues
  - Typical uses
    - Uploaded files
    - Producer / Consumer pattern with Queues
    - NoSql support with Tables
    - SDK Dependencies for WebJobs
      - Specifically, the dashboard

Publishing to Azure

- Let's walk through a live demo
  - Portal to create
    - Resource Group
    - App Service Plan
    - App Service
  - Pin resources to custom dashboard
  - Visual Studio to deploy
    - Software to the App Service

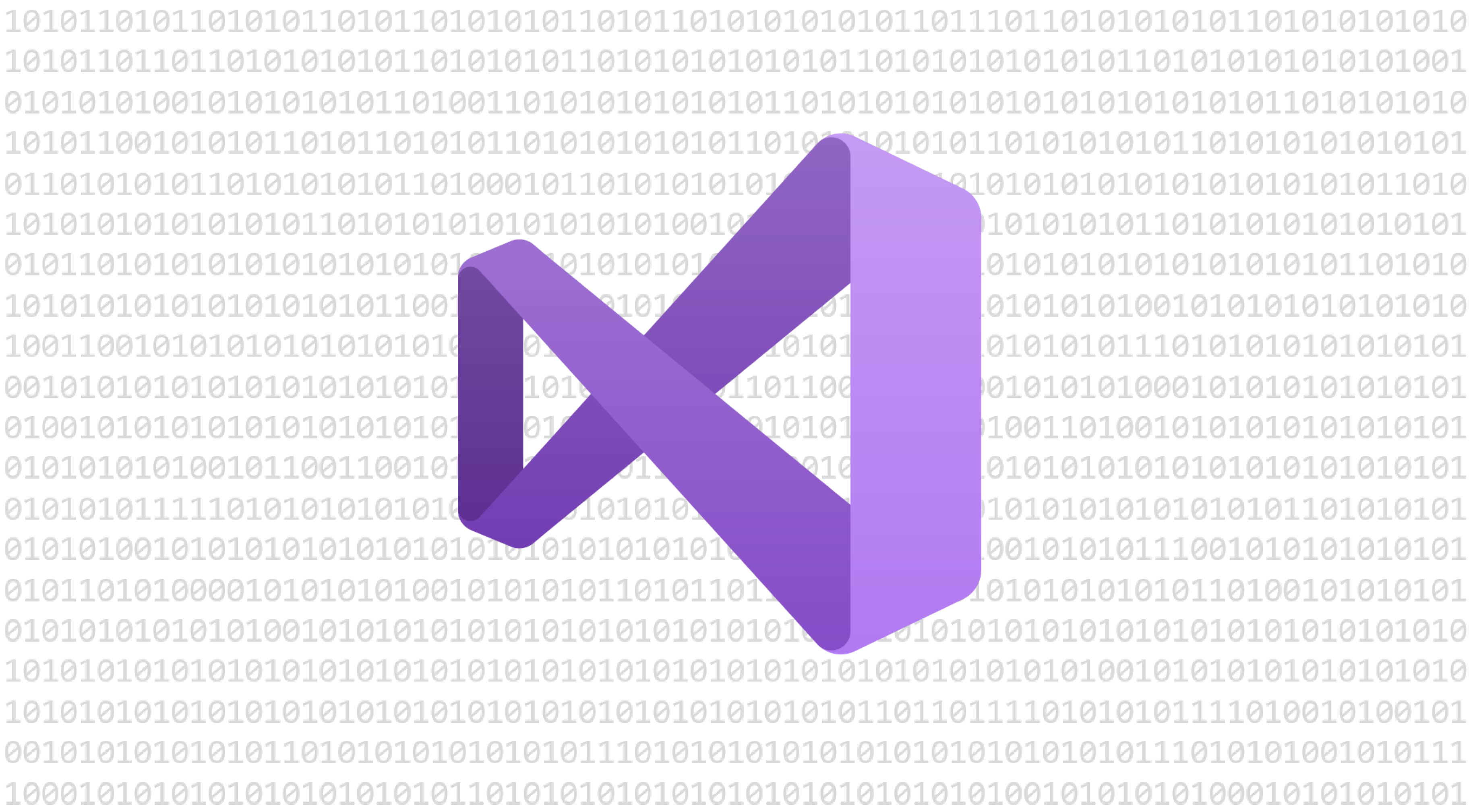


Demo

Publishing to Azure

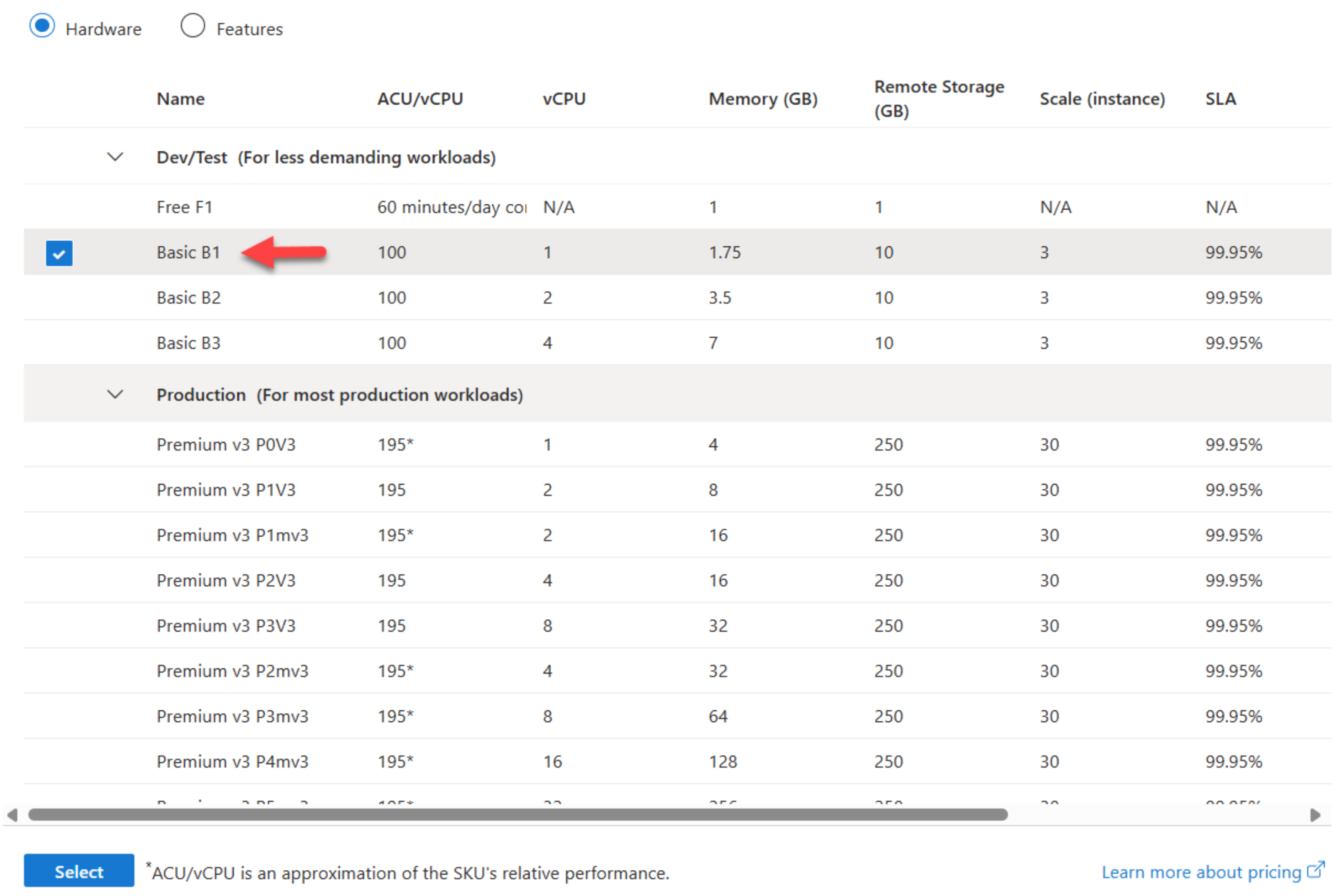
```
Portal + Visual Studio
WeatherForecastSwaggerSolution
    WeatherForecastSwagger
```

Visual Studio 2022



Visual Studio - Don't Forget

- Use the Linux Basic B1



[✓ Hardware] Features

	Name	ACU/vCPU	vCPU	Memory (GB)	Remote Storage (GB)	Scale (instance)	SLA
	Dev/Test (For less demanding workloads)						
	Free F1	60 minutes/day co	N/A	1	1	N/A	N/A
✓	Basic B1	100	1	1.75	10	3	99.95%
	Basic B2	100	2	3.5	10	3	99.95%
	Basic B3	100	4	7	10	3	99.95%
	Production (For most production workloads)						
	Premium v3 P0V3	195*	1	4	250	30	99.95%
	Premium v3 P1V3	195	2	8	250	30	99.95%
	Premium v3 P1mv3	195*	2	16	250	30	99.95%
	Premium v3 P2V3	195	4	16	250	30	99.95%
	Premium v3 P3V3	195	8	32	250	30	99.95%
	Premium v3 P2mv3	195*	4	32	250	30	99.95%
	Premium v3 P3mv3	195*	8	64	250	30	99.95%
	Premium v3 P4mv3	195*	16	128	250	30	99.95%

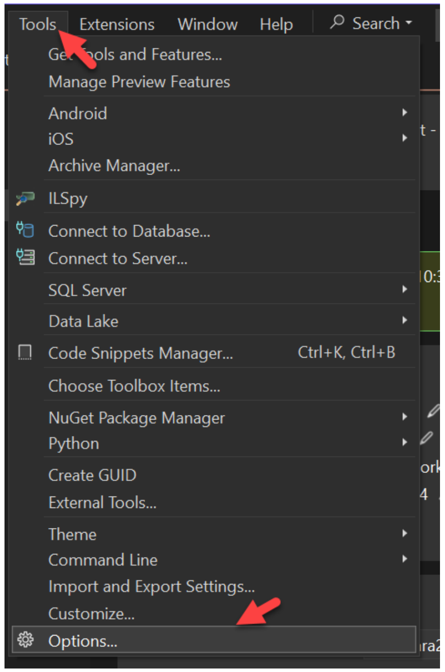
Select

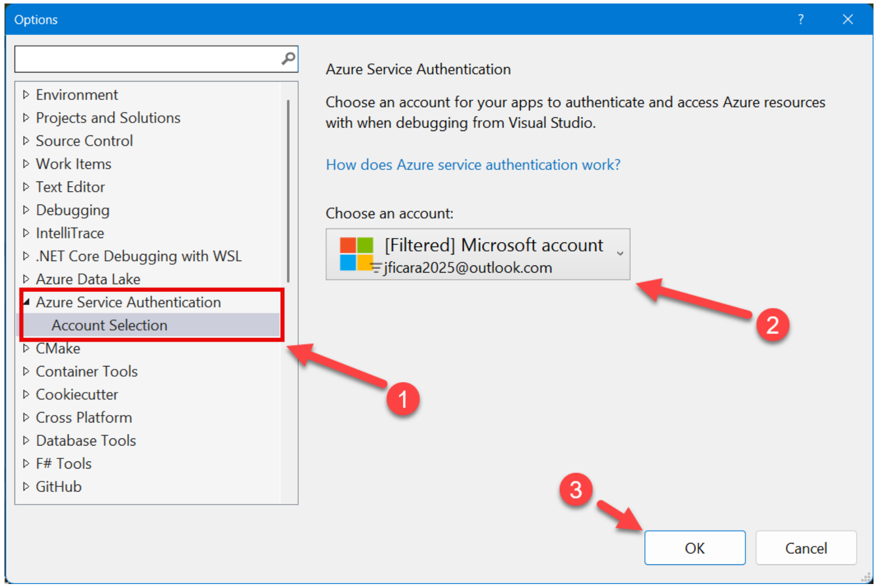
*ACU/vCPU is an approximation of the SKU's relative performance.

Learn more about pricing

Visual Studio - Don't Forget

- Ensure correct account is used
  - In Visual Studio





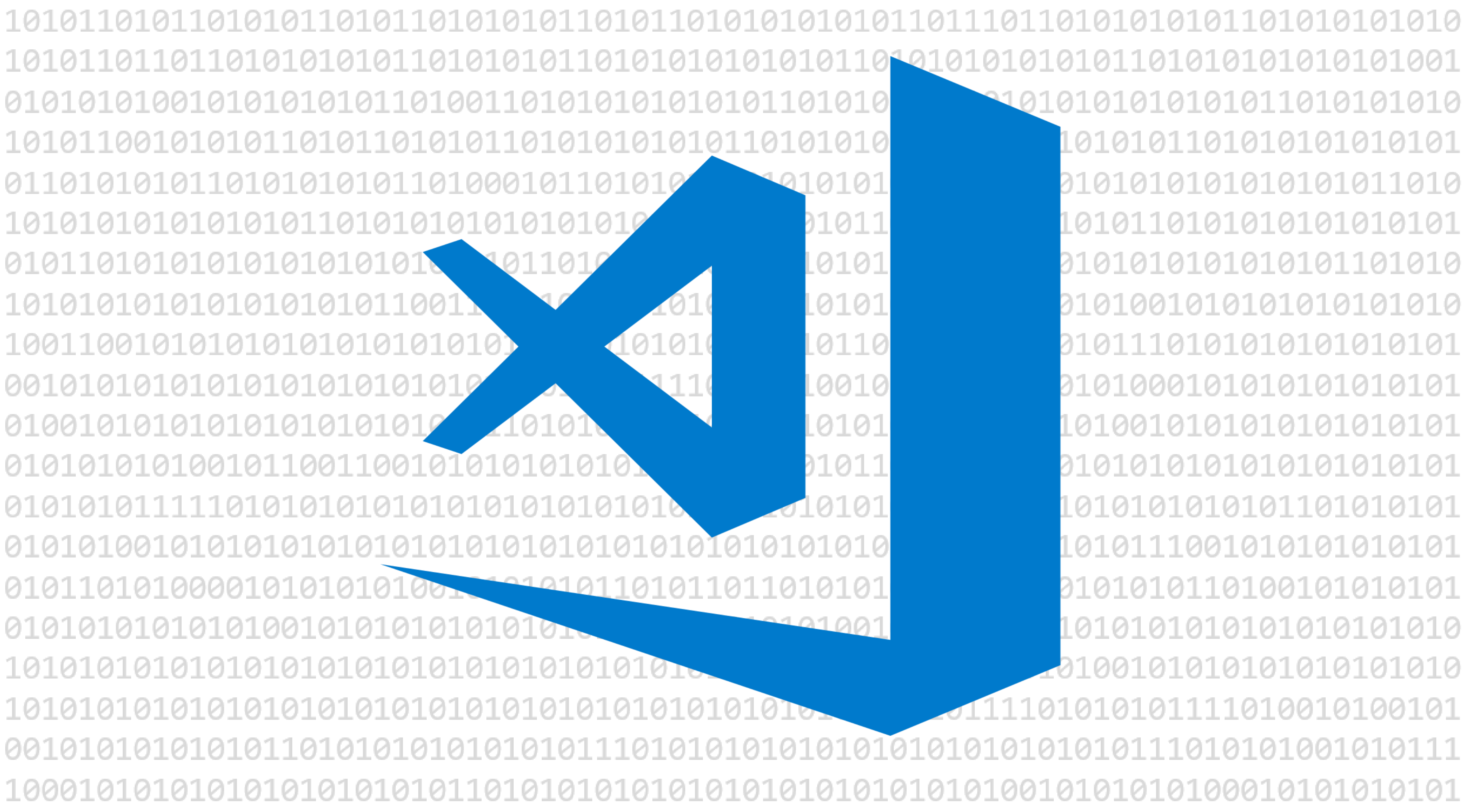
Azure Service Authentication

Choose an account for your apps to authenticate and access Azure resources with when debugging from Visual Studio.

How does Azure service authentication work?

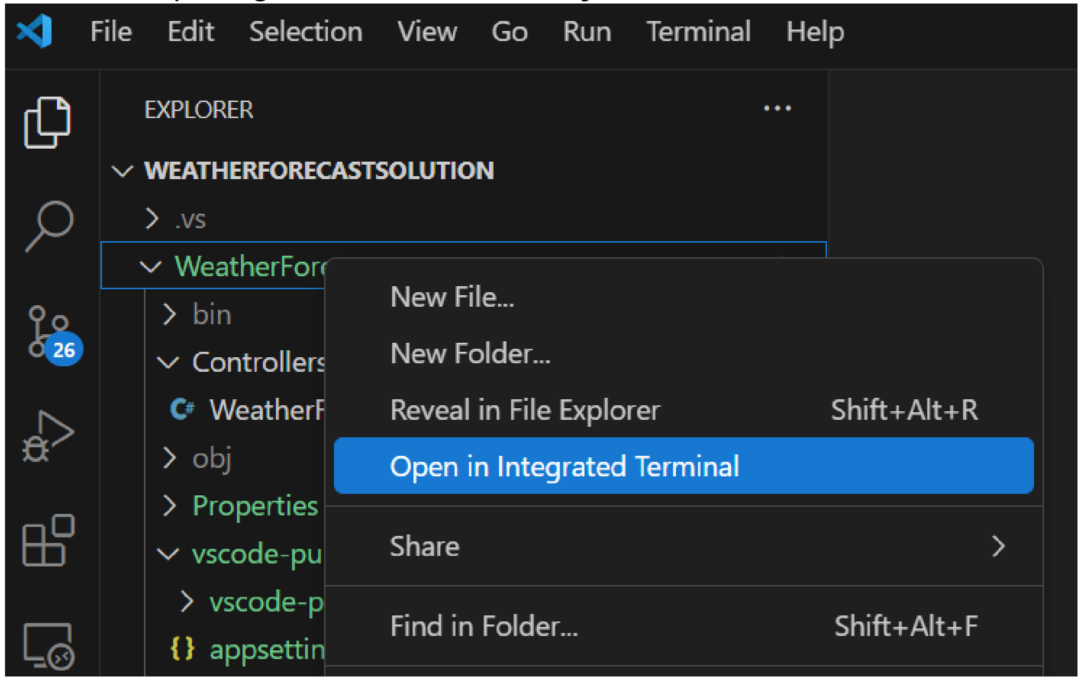
Choose an account: [Filtered] Microsoft account jficara2025@outlook.com

VS Code



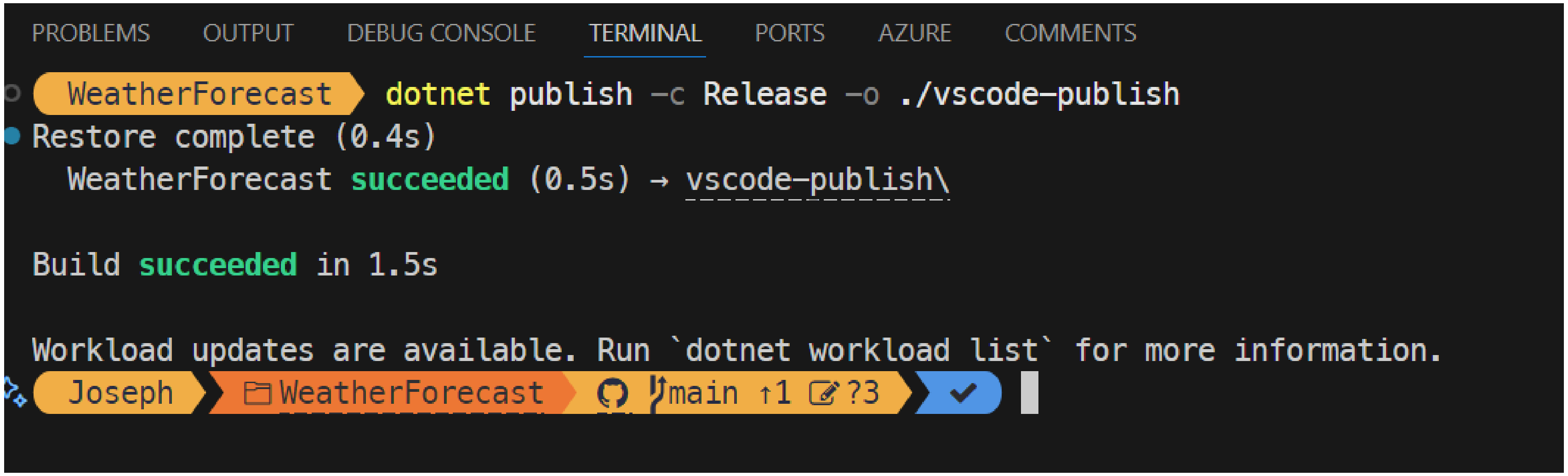
VS Code Notes

- To Publish from VS Code
  - Open an integrated terminal
    - In the project directory



VS Code Notes

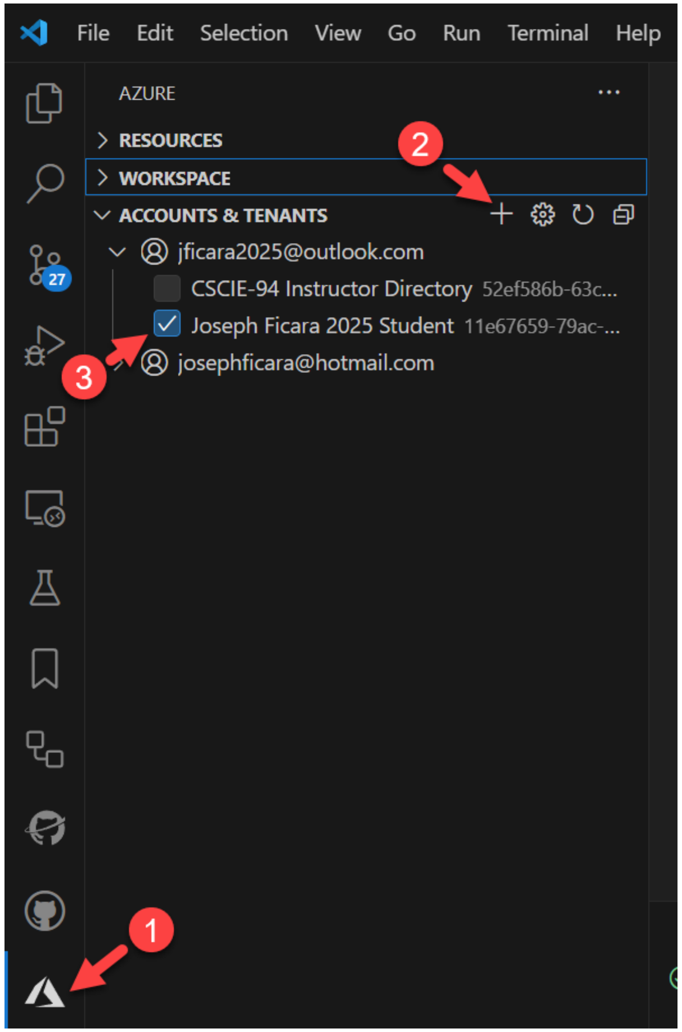
- Publish the code **locally**
  - dotnet publish -c Release -o ./vscode-publish



```
WeatherForecast  dotnet publish -c Release -o ./vscode-publish
Restore complete (0.4s)
  WeatherForecast succeeded (0.5s) → vscode-publish\

Build succeeded in 1.5s

Workload updates are available. Run `dotnet workload list` for more information.
Joseph  WeatherForecast  main ↑1 ?3  ✓
```

VS Code - Don't Forget

1. Select Azure
2. Add **your account**
   3. Select **your directory**

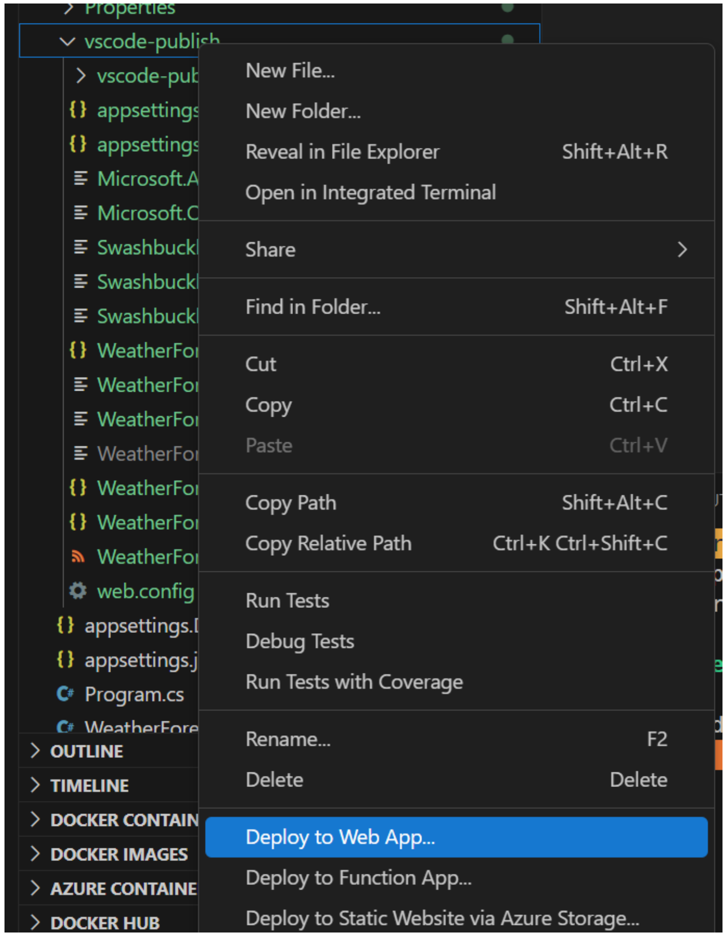


jficara2025@outlook.com
- CSCIE-94 Instructor Directory 52ef586b-63c...
- ✓ Joseph Ficara 2025 Student 11e67659-79ac-...

josephficara@hotmail.com

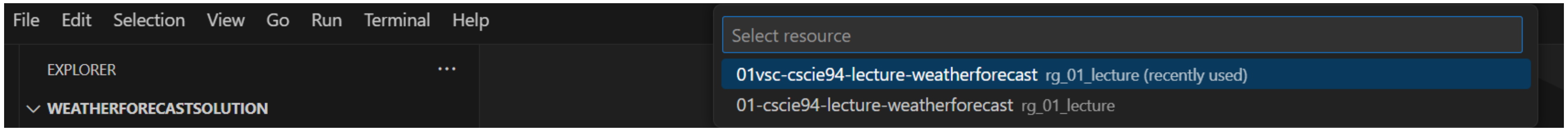
VS Code Notes

- Deploy to the web app
  - Right click **parent vscode-publish** folder
    - Choose deploy to Web App …



VS Code Notes

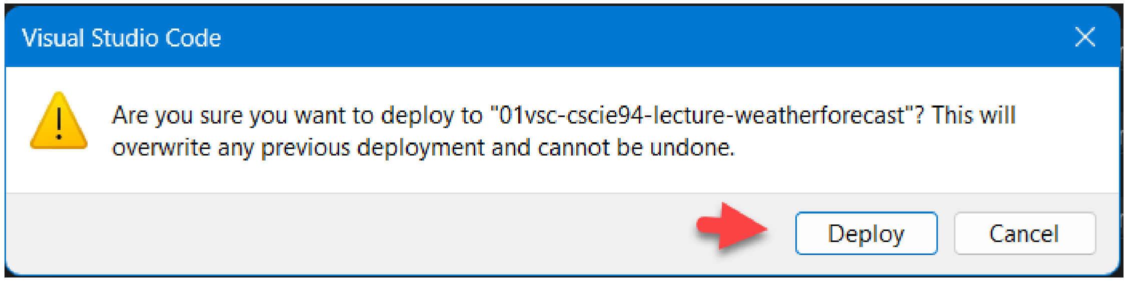
- Choose the Web App to deploy to



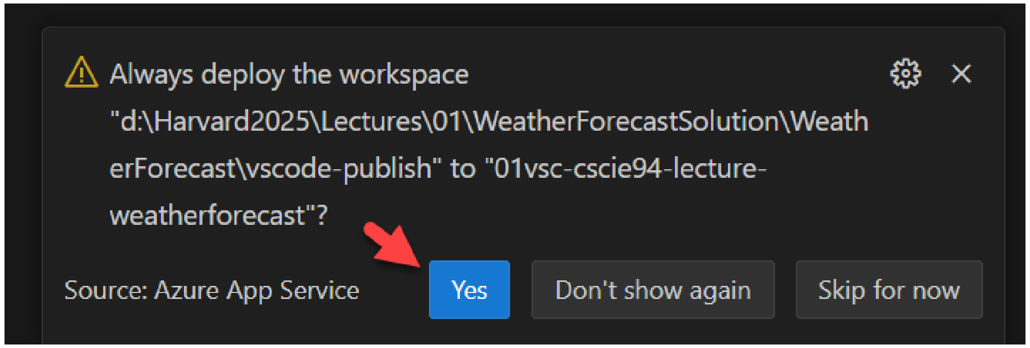
Select resource

01vsc-cscie94-lecture-weatherforecast rg_01_lecture (recently used)

01-cscie94-lecture-weatherforecast rg_01_lecture



Are you sure you want to deploy to "01vsc-cscie94-lecture-weatherforecast"? This will overwrite any previous deployment and cannot be undone.

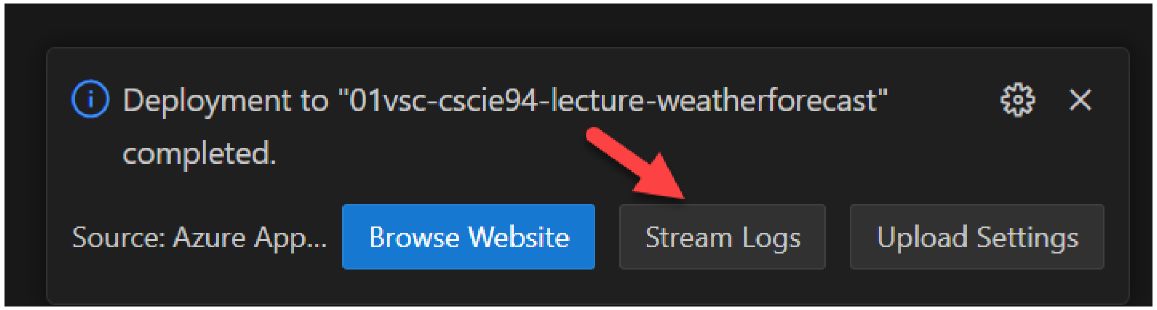


Always deploy the workspace "d:\Harvard2025\Lectures\01\WeatherForecastSolution\WeatherForecast\vscode-publish" to "01vsc-cscie94-lecture-weatherforecast"?

Source: Azure App Service

VS Code Notes

- Stream logs locally
  - To see diagnostic data from your web app
    - In vscode



Deployment to "01vsc-cscie94-lecture-weatherforecast" completed.

General Notes

- You may get gateway errors
  - Just wait a bit for the website to start
  - Why does this happen?
    - Typically, due to low resources
      - When using a smaller tier such as **Basic B1**

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: Implement rich Azure PaaS ecosystems using containers, serverless services, and storage solutions, 2nd Edition
  - Author: Kamil Mrzygłód
  - ISBN: 978-1803240091
  - Chapter 1

Further Reading

- Pro ASP.NET Core 6
  - Author: Adam Freeman
  - ISBN: 978-1484279564
  - Chapter 19

  -- *OR* --

- Pro ASP.NET Core 7
  - Author: Adam Freeman
  - ISBN: 1633437825
  - Chapter 19

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 2022
  - Built in support for .http files
- Visual Studio Code
  - 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