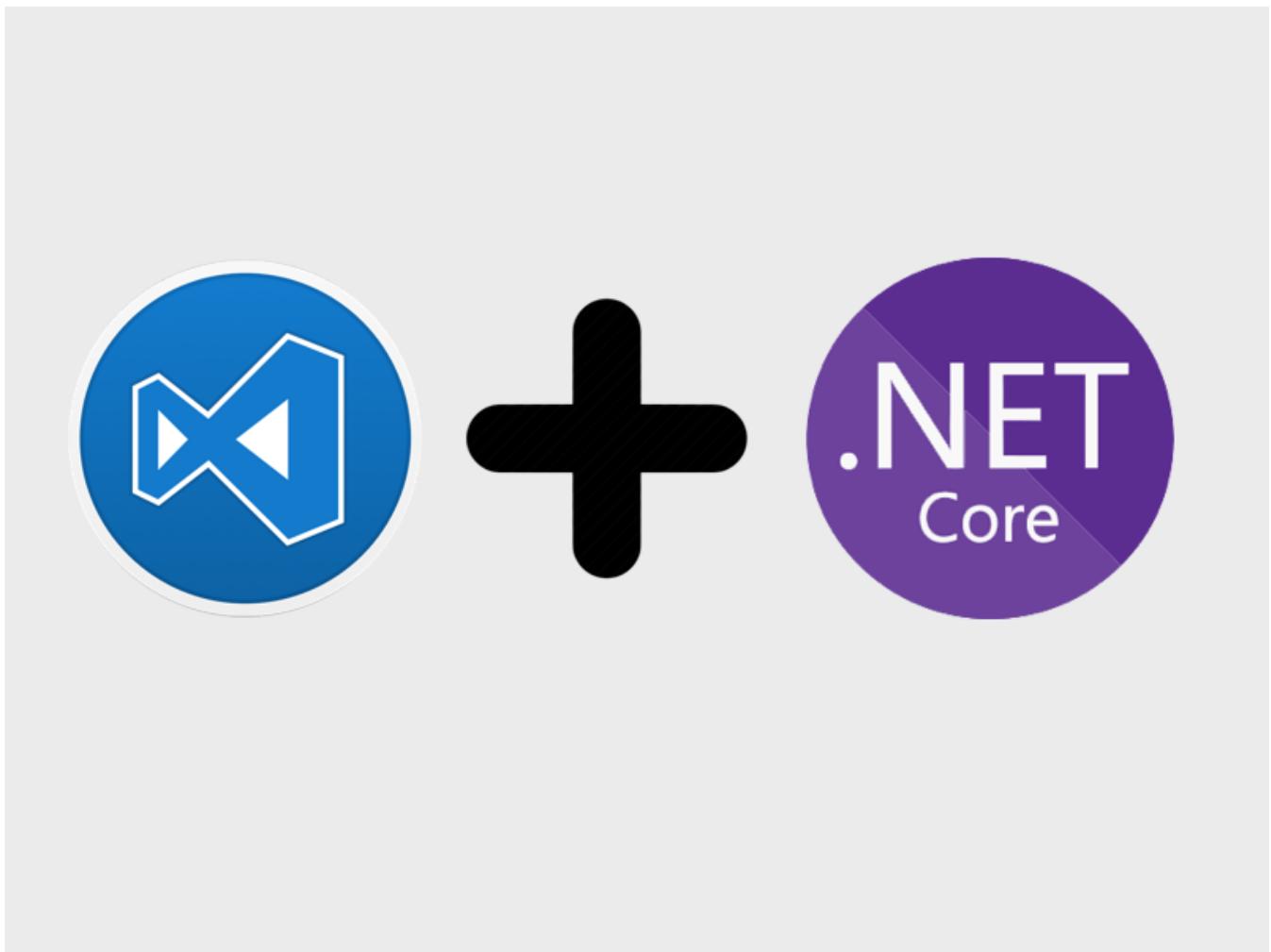


Vscode + ASP.NET Core: Setup development environment



Daniel Padua Ferreira
Jun 11, 2019 · 5 min read



For a long time Visual Studio has been the main development tool in the .NET world and recently in .NET Core. With time, many features were added making it even more heavier. I feel like I'm losing development agility lately for having to live with features that I do not use, and also, I feel like I'm in some sort of "comfort zone" using Visual Studio, mostly because it does a **lot** of things "under the hood" — this may cause a lack of understanding of the developer.

Taking advantage of the rise of Visual Studio Code and also of the open source culture that is increasing as a part of the .NET vein, I've decided to create this guide to show how quickly is to setup a development environment using vscode as well as using its extensions to gain productivity as good as provided by Visual Studio, but much lighter and without so many things that does stuff “automagically”.

• • •

Installing .NET Core SDK

There are a version of .NET Core SDK for Windows, Linux and macOS. To install it there are some possibilities, for example:

1. Download and install directly from Microsoft:

<https://dotnet.microsoft.com/download> selecting your OS

2. (Recommended) Using a package manager, as it follows:

- Windows

Yes, actually there **is** a package manager for Windows, and it's called: **Chocolatey**.

Follow this installation guide: <https://chocolatey.org/install> and install the dotnet-sdk using the **PowerShell** command:

```
choco install dotnetcore-sdk
```

- Linux

For those who are penguin lovers, you must already know N distros and N package managers, you only need to search for “*dotnet core sdk*”. For example, in Fedora you can use the following guide: <https://dotnet.microsoft.com/download/linux-package-manager/fedora28/sdk-2.1.301> to configure the repository and install it using:

```
sudo dnf install dotnet-sdk-2.1
```

- MacOS

To Apple users I recommend you to use Homebrew: <https://brew.sh> and install via cask, with the command line:

```
brew cask install dotnet-sdk
```

• • •

Installing Visual Studio Code

The instructions above are also valid when installing vscode, it can be installed using a package manager or downloading directly in <https://code.visualstudio.com/download>

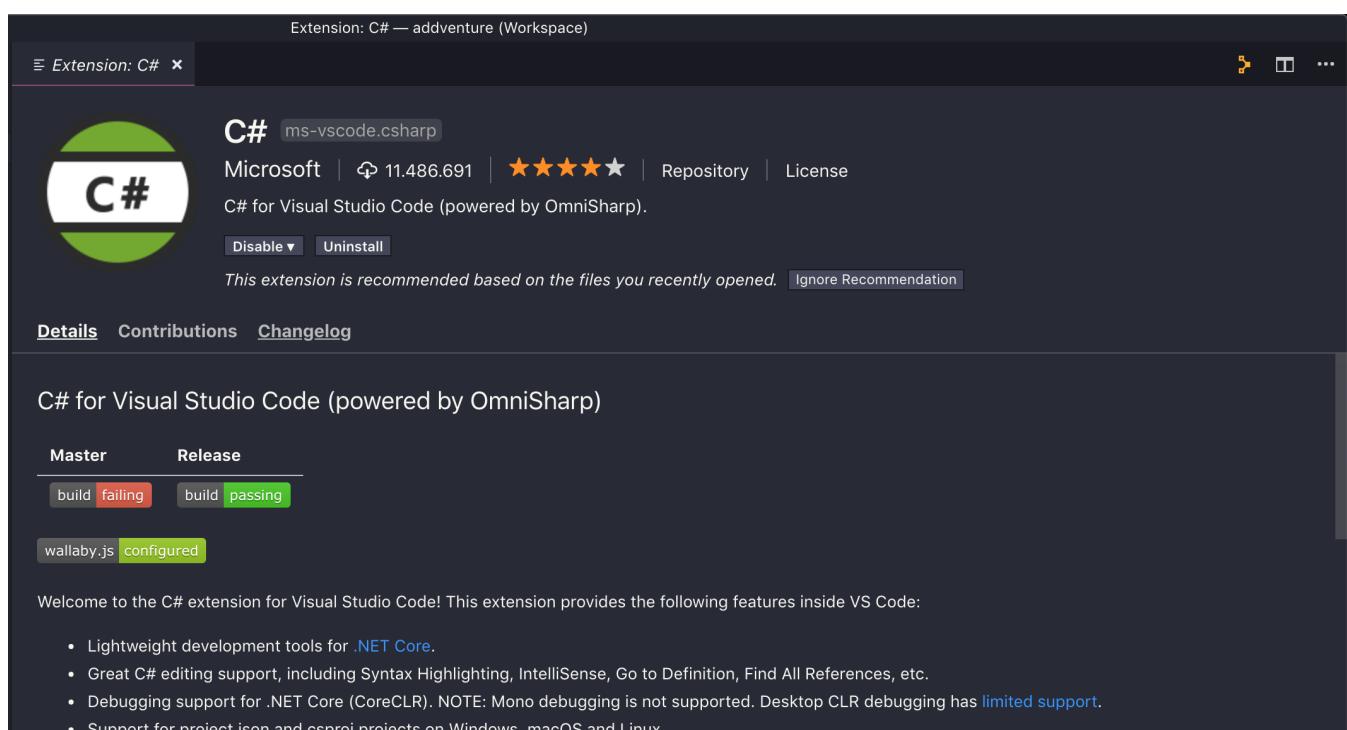
• • •

Configuring Visual Studio Code Extensions

To achieve some basic functionalities that we used to have in Visual Studio, for instance: debug, auto-complete, compiler etc, it will be necessary to install some extensions in vscode:

Essentials

- C# (Omnisharp)



The screenshot shows the Microsoft Store page for the "C# — addventure (Workspace)" extension. The extension is developed by Microsoft and has 11,486.691 installations. It has a 5-star rating. The description states: "C# for Visual Studio Code (powered by OmniSharp)". There are "Disable" and "Uninstall" buttons. A note says: "This extension is recommended based on the files you recently opened." Below the main section, there's a "Details" tab showing the extension's GitHub repository and a "Changelog" tab. The "Changelog" tab lists the following changes:

- C# for Visual Studio Code (powered by OmniSharp)
- Master** **Release**
- build failing build passing
- wallaby.js configured

Welcome to the C# extension for Visual Studio Code! This extension provides the following features inside VS Code:

- Lightweight development tools for .NET Core.
- Great C# editing support, including Syntax Highlighting, IntelliSense, Go to Definition, Find All References, etc.
- Debugging support for .NET Core (CoreCLR). NOTE: Mono debugging is not supported. Desktop CLR debugging has limited support.
- Support for project.json and csproj projects on Windows, macOS and Linux.

The C# extension is powered by [OmniSharp](#).

Get Started Writing C# in VS Code

- [Documentation](#)
- [Video Tutorial compiling with .NET Core](#)

What's new in 1.17.1

- Updated Razor language service to fix various Razor editing reliability issues. For details see <https://github.com/aspnet/Razor.VSCode/releases/tag/1.0.0-rc.1>

This is the official C# extension by Microsoft. Using only this, you'll already be able to debug, auto-complete and use the compiler without any trouble.

Optionals — The ones that I'm using/testing

- C# XML Documentation Comments

C# XML Documentation Comments k-kato.doccomment

Keisuke Kato | ⚡ 452.236 | ★★★★★ | Repository | License

Generate C# XML documentation comments for ///

[Disable](#) [Uninstall](#)

[Details](#) [Contributions](#) [Changelog](#)

XML Documentation Comments Support for Visual Studio Code

build passing Coverage Status license [MIT](#) Visual Studio Marketplace v0.1.6 installs 169.31K

Generate XML documentation comments for Visual Studio Code.

Usage

Type "///", it auto-generates an XML documentation comment like this:

```
X.cs
1 namespace N // "N:N"
2
3 public unsafe class X // "T:N.X"
4 {
5     public X() {} // "M:N.X.#ctor"
6     public X(int i) {} // "M:N.X.#ctor(System.Int32)"
7     public string q; // "F:N.X.q"
8     public const double PI = 3.14; // "F:N.X.PI"
9     public int f(){return 1;} // "M:N.X.f"
10    public int bb(string s, ref int y, void * z){return 1;} // "M:N.X.bb(System.String,System.Int32@,=System.Void*)"
11    public int gg(short[] array1, int[,] array){return 0;} // "M:N.X.gg(System.Int16[], System.Int32[0:,0:])"
12    public static X operator+(X x, X xx){return x;} // "M:N.X.op>Addition(N.X,N.X)"
13    public int prop {get{return 1;} set{}} // "P:N.X.prop"
14    public event D d; // "E:N.X.d"
```

This extension basically provides auto-complete when typing “///” that generates basic XML documentation template of the scope, like Visual Studio.

- ASP.NET Core Snippets

Extension: ASP.NET Core Snippets — addventure (Workspace)

Extension: ASP.NET Core Snippets [...](#)

ASP.NET Core Snippets rahulsahay.csharp-aspnetcore

This is a collection of asp.net core snippets for Visual Studio Code.

This extension contains some code snippets to auto-complete: methods, properties etc.

- **C# Extensions**

Welcome to C# Extensions. This VSCode extension provides extensions to the IDE that will hopefully speed up your development workflow.

Features

Add C# Class

Cookies

This extension provides some specific .NET Core context actions, like: create a C# class by clicking with the mouse's right button, instead of create a blank .cs file.

- Bracket Pair Colorizer

Extension: Bracket Pair Colorizer — addventure (Workspace)

Extension: Bracket Pair Colorizer x



Bracket Pair Colorizer coenraads.bracket-pair-colorizer

Coenraads | 3.488.277 | ★★★★★ | Repository | License

A customizable extension for colorizing matching brackets

[Disable ▾](#) [Uninstall](#)

[Details](#) [Contributions](#) [Changelog](#)

Bracket Pair Colorizer

Announcement: A new version is being developed at <https://github.com/CoenraadS/Bracket-Pair-Colorizer-2>

This extension allows matching brackets to be identified with colours. The user can define which characters to match, and which colours to use.

Screenshot:

```

22  public updateDocument(document: TextDocument) {
23    const documentDecoration = this.getDocumentDecorations(document);
24    if (documentDecoration) {
25      documentDecoration.triggerUpdateDecorations();
26    }
27  }

```

[Release Notes](#)

[Features](#)

User defined matching characters

By default (), [], and {} are matched, however custom bracket characters can also be configured.

This one set colors to opening and closing brackets, that eases code visualization.

- Path Intellisense

Extension: Path Intellisense — addventure (Workspace)

Extension: Path Intellisense x



Path Intellisense christian-kohler.path-intellisense

Christian Kohler | 1.790.529 | ★★★★★ | Repository | License

Visual Studio Code plugin that autocompletes filenames

[Disable ▾](#) [Uninstall](#)

[Details](#) [Contributions](#) [Changelog](#)

Path Intellisense

Visual Studio Code plugin that autocompletes filenames.

Installation

In the command palette (cmd+shift+p) select Install Extension and choose Path Intellisense.

Usage



Node packages intellisense

Use [npm intellisense](#)

Contributing

Something missing? Found a bug? - Create a pull request or an issue. [Github](#)

Are you a windows user?

There is an issue on windows with the period key. See Issue <https://github.com/ChristianKohler/NpmIntellisense/issues/12>

Add this to the keybinding:



This is a extremely useful extension. When typing in a string, like: “..” it lists your project files to auto-complete.

- Prettier

Extension: Prettier - Code formatter — addventure (Workspace)

Extension: Prettier - Code formatter x

 **Prettier - Code formatter** esbenp.prettier-vscode

Esben Petersen | ⚡ 7.567.639 | ★★★★★ | Repository | License

VS Code plugin for prettier/prettier

Disable ▾ Uninstall

[Details](#) [Contributions](#) [Changelog](#)

Prettier formatter for Visual Studio Code

VS Code package to format your JavaScript / TypeScript / CSS using [Prettier](#).

Installation

Install through VS Code extensions. Search for **Prettier – Code formatter**

[Visual Studio Code Market Place: Prettier – Code formatter](#)

Can also be installed in VS Code: Launch VS Code Quick Open (Ctrl+P), paste the following command, and press enter.

```
ext install esbenp.prettier-vscode
```

⚠ A word of warning-if you have any other code formatting extensions installed such as for example hugely popular `HookyQR.beautify` or `taichi.react-beautify` they might take precedence and format your code instead of Prettier leading to unexpected results.

Usage

Using Command Palette (CMD/CTRL + Shift + P)

1. CMD + Shift + P → Format Document
- OR
1. Select the text you want to Prettify
2. CMD + Shift + P → Format Selection

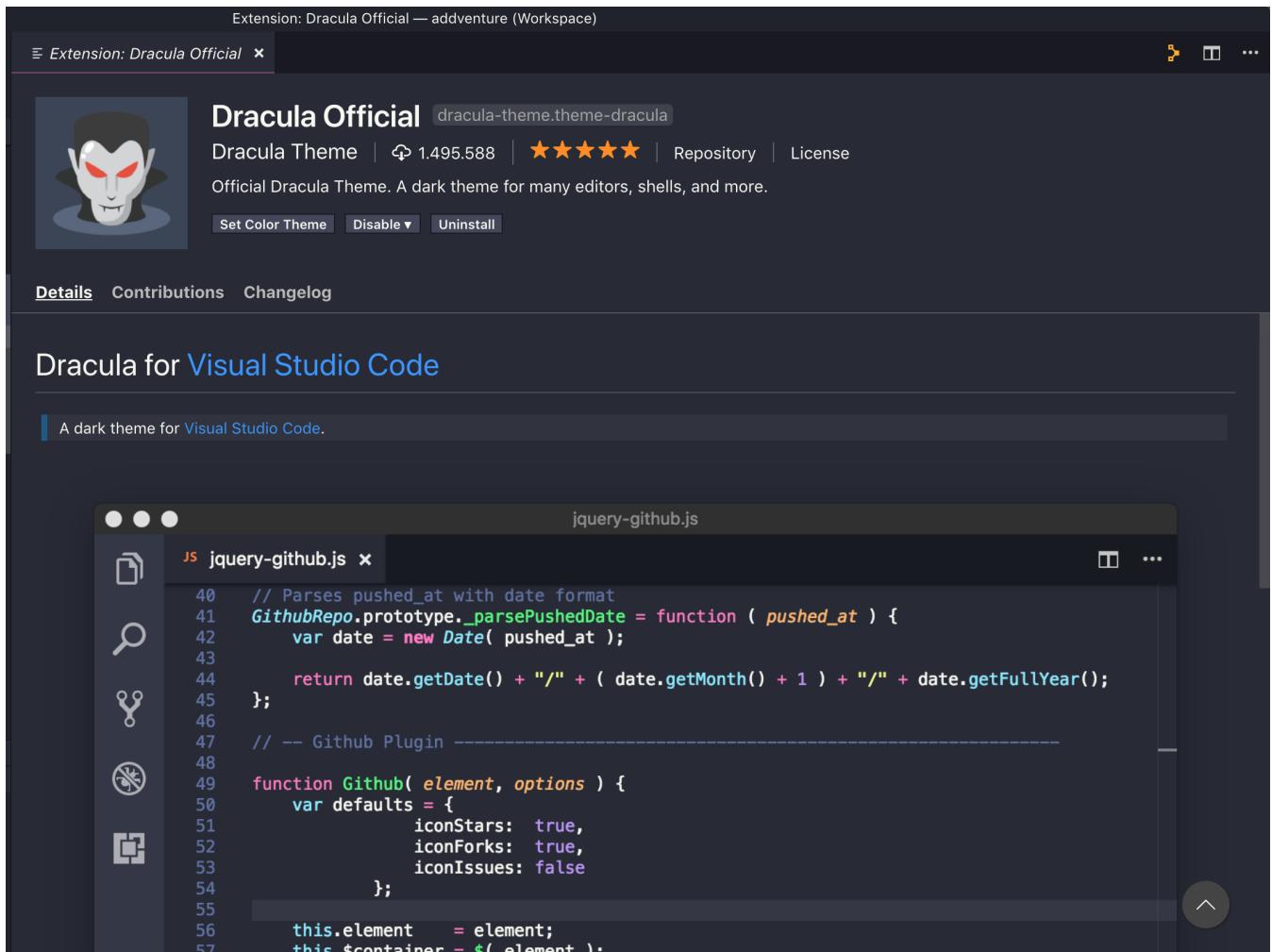


Up arrow icon

Extension used to format and indent code, and it can be used for multiple languages.

Bonus: Theme

- Dracula Official



This is the theme that I've best adapted.

Creating a test project

Finally, the time has come to create a project and check if the setup is correct. Open vscode's integrated terminal (ctrl+`) and follow instructions below:

1. Create a directory for your project using the command line:

```
mkdir ~/project-test
```

2. Navigate to the created directory, unless you're not already in it:

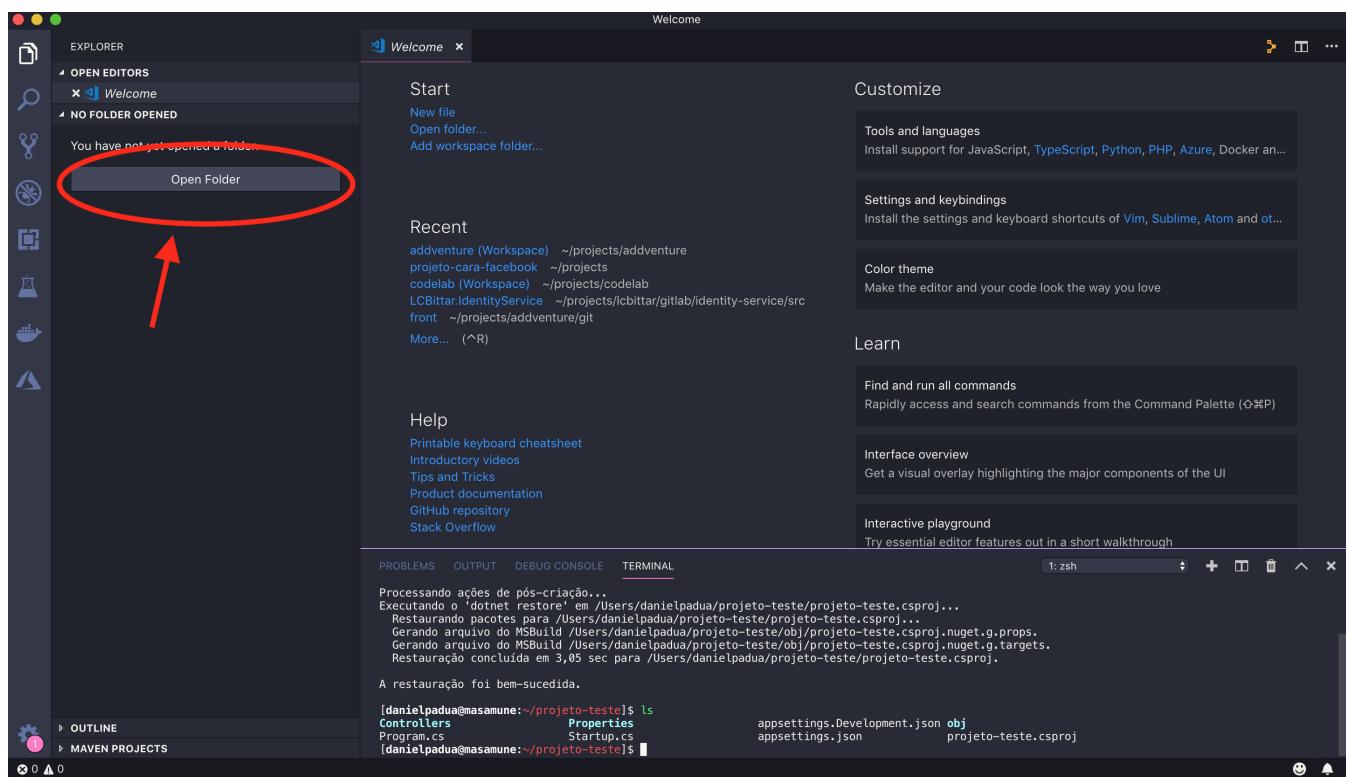
```
cd ~/project-test
```

3. Create a project using **.NET Core CLI** (more of CLI's info is in <https://docs.microsoft.com/en-us/dotnet/core/tools/?tabs=netcore2x>):

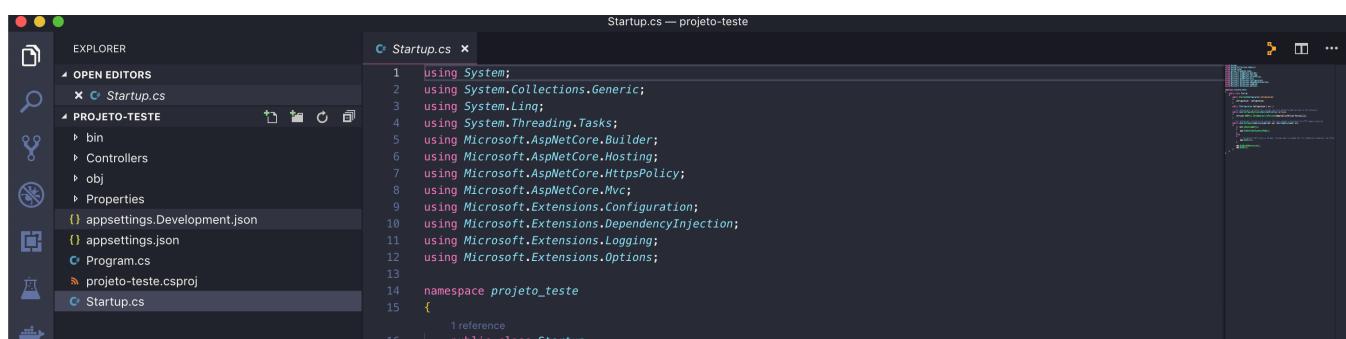
```
dotnet new webapi
```

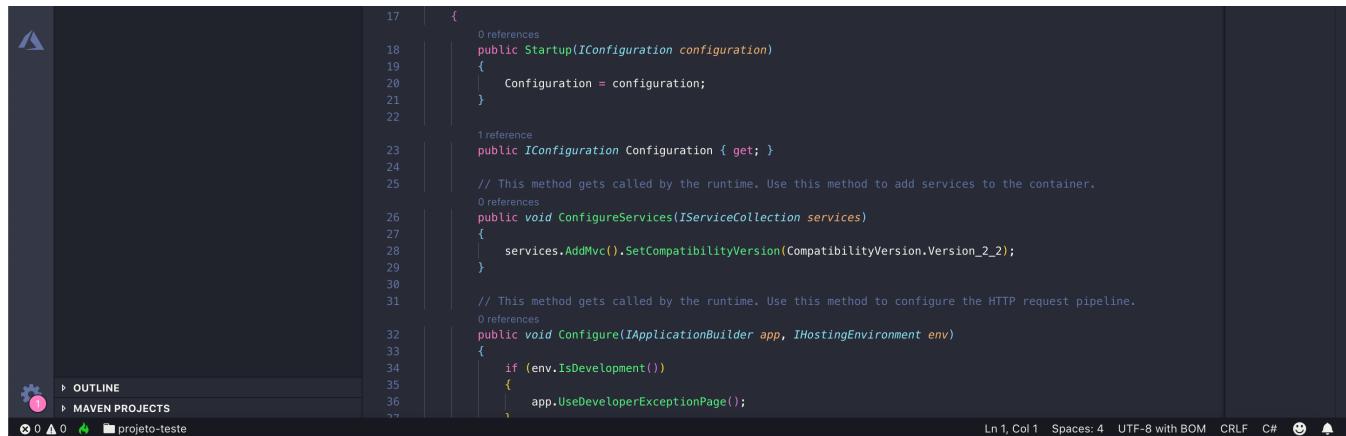
At this moment, the project structure was created in “~/project-test” (~ means your user’s home folder). You only have to associate the created project in vscode, as it follows:

Click in the “Open Folder” button



Navigate and open: ~/project-test





```

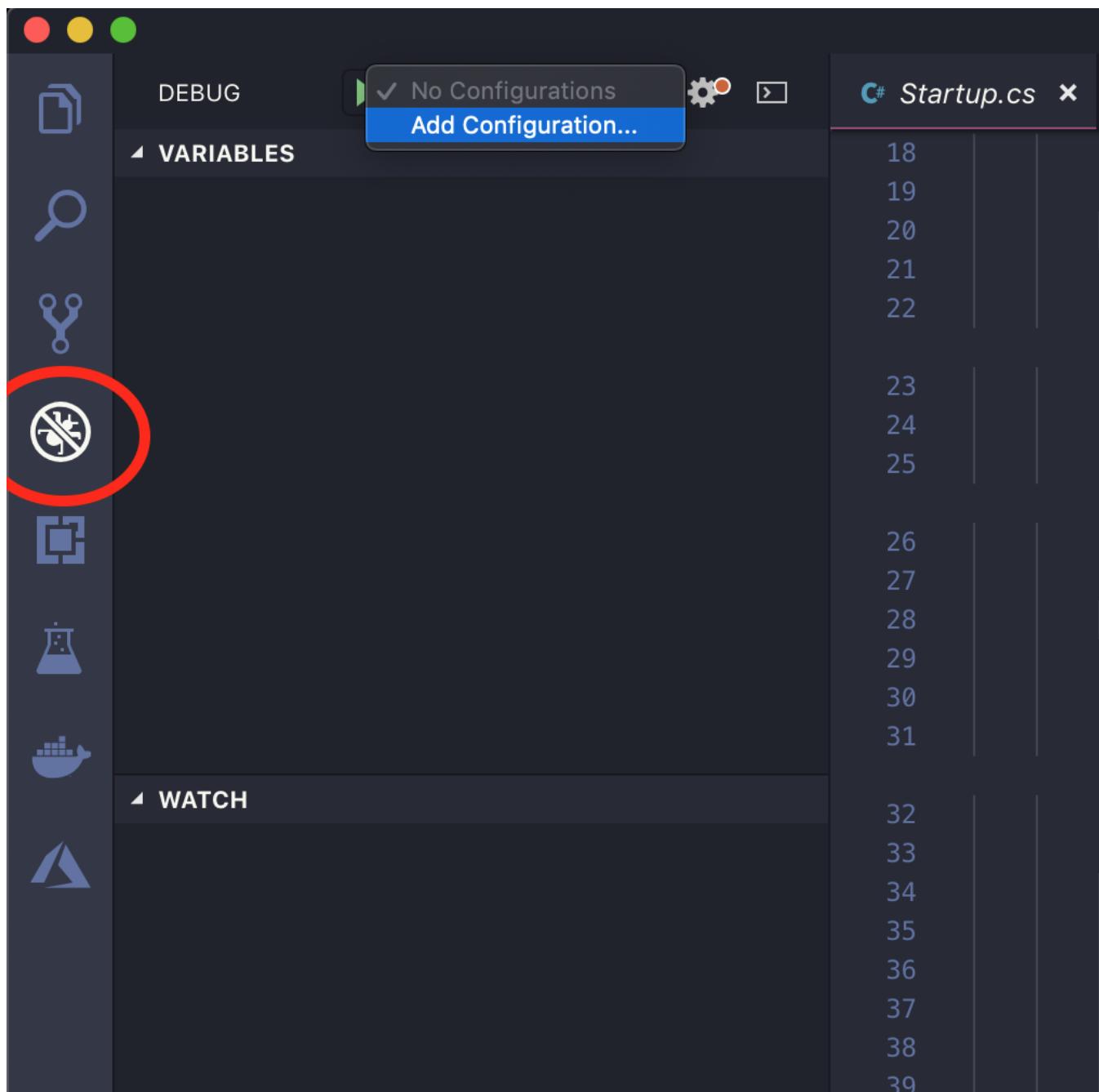
17  {
18      0 references
19      public Startup(IConfiguration configuration)
20      {
21          Configuration = configuration;
22      }
23      1 reference
24      public IConfiguration Configuration { get; }
25
26      // This method gets called by the runtime. Use this method to add services to the container.
27      0 references
28      public void ConfigureServices(IServiceCollection services)
29      {
30          services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
31      }
32
33      // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
34      0 references
35      public void Configure(IApplicationBuilder app, IHostingEnvironment env)
36      {
37          if (env.IsDevelopment())
38          {
39              app.UseDeveloperExceptionPage();

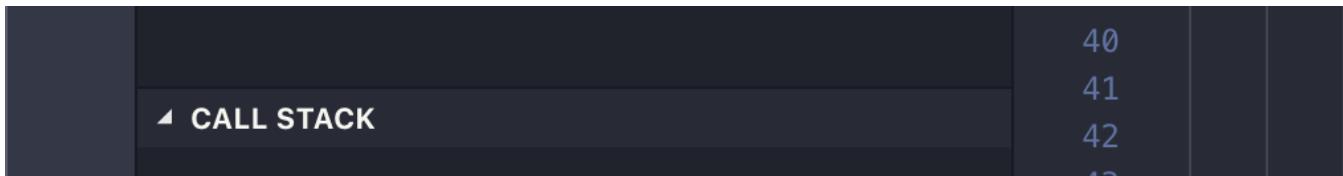
```

Ln 1, Col 1 Spaces: 4 UTF-8 with BOM CRLF C# 😊 🔔

Put a breakpoint in any line of “Startup.cs” and then we’ll create a debug profile.

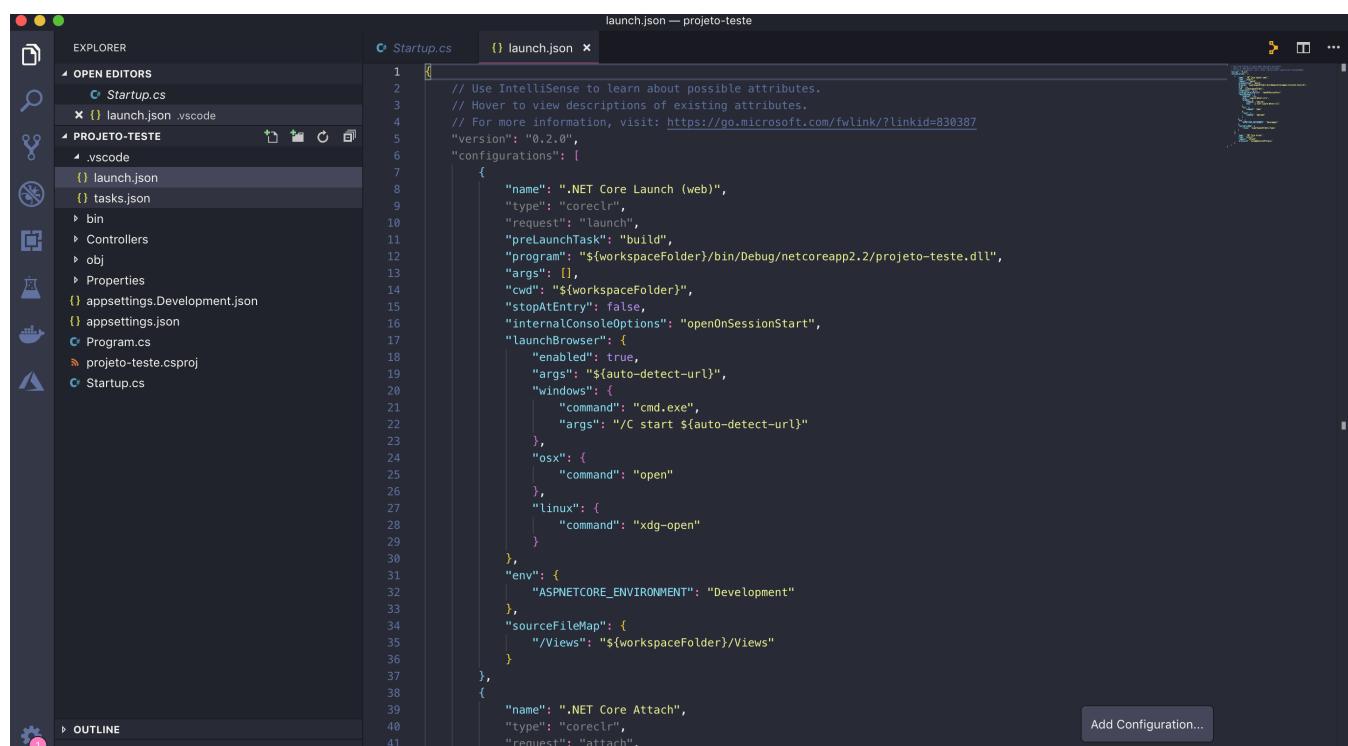
Click on the debug icon at the sidebar and then in “No Configuration” and select “Add Configuration...”





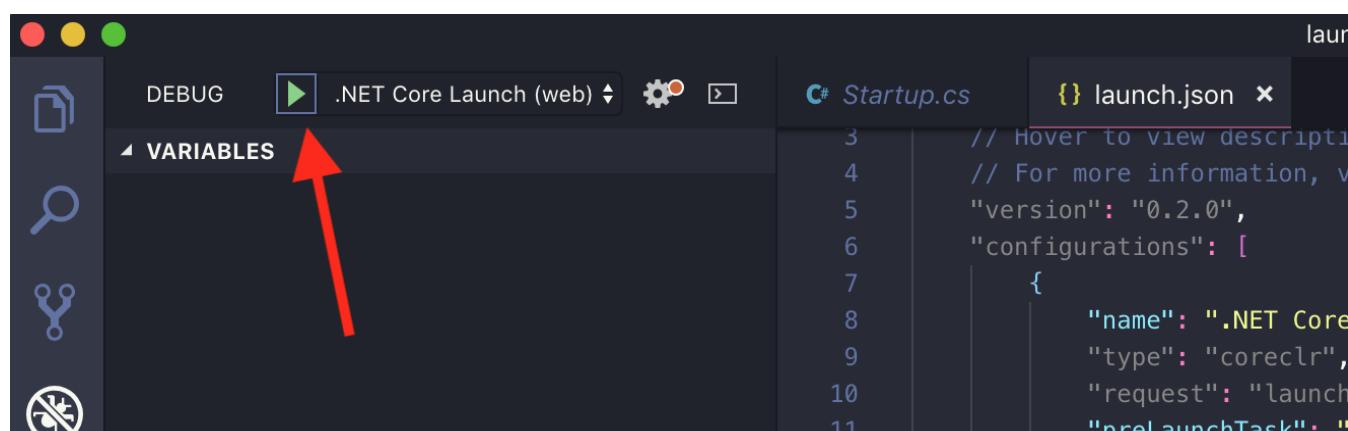
vscode will load the extension, so maybe you'll have to wait for a while. When it's ready, select “.NET Core” in the main bar.

At this moment, you may realize that vscode created a hidden folder inside your project, called: “.vscode” and inside, 2 files: launch.json and tasks.json:



These files tells the extension, the tasks that have to be executed to compile your project, and how the debugger can attach correctly to your app's process.

The template configuration will do for this test project, so basically you only have to hit run button!



And voilá, the debugger is on:

```

Startup.cs — projeto-teste
31   // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
32   public void Configure(IApplicationBuilder app, IHostingEnvironment env)
33   {
34     if (env.IsDevelopment())
35     {
36       app.UseDeveloperExceptionPage();
37     }
38     else
39     {
40       // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://
41       app.UseHsts();
42     }
43
44     app.UseHttpsRedirection();
45     app.UseMvc();
46   }
47
48 }

```

DEBUG .NET Core Launch (web) ⚙️

VARIABLES

- Locals
- Watch

WATCH

CALL STACK

- <No Name> PAUSED ON BREAKPOINT
- [External Code] Unknown Source
- projeto-teste.dll!projeto_teste.Startup.Configure [External Code] Unknown Source
- <No Name> PAUSED
- <No Name> PAUSED

BREAKPOINTS

- All Exceptions
- User-Unhandled Exceptions
- Startup.cs

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Loaded '/usr/local/share/dotnet/shared/Microsoft.NETCore.App/2.2.2/System.Collections.dll'. Module was built without symbols.
Loaded '/usr/local/share/dotnet/shared/Microsoft.AspNetCore.App/2.2.2/Microsoft.Extensions.Configuration.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.
Loaded '/usr/local/share/dotnet/shared/Microsoft.AspNetCore.App/2.2.2/Microsoft.Extensions.Configuration.EnvironmentVariables.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.
Loaded '/usr/local/share/dotnet/shared/Microsoft.AspNetCore.App/2.2.2/Microsoft.Extensions.Primitives.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.
Loaded '/usr/local/share/dotnet/shared/Microsoft.NETCore.App/2.2.2/System.Threading.Tasks.dll'. Module was built without symbols.
Loaded '/usr/local/share/dotnet/shared/Microsoft.NETCore.App/2.2.2/System.Runtime.Extensions.dll'. Module was built without symbols.
Loaded '/usr/local/share/dotnet/shared/Microsoft.NETCore.App/2.2.2/System.Linq.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

Ln 45, Col 26 (13 selected) Spaces: 4 UTF-8 with BOM CRLF C# ☺ 🔍

With this we finish successfully the .NET Core + Visual Studio Code development setup. Remember that vscode can also be used to program in almost any language... So, despite it's lightweight, it is extremely powerful and versatile.

I recommend using vscode by personal experience. I feel like I was able to obtain more knowledge about .NET Core after I migrated from Visual Studio.

I hope this was useful to you too. See you soon!

Visual Studio Code

Vscode

Dotnet Core

Csharp

English

Medium

About Help Legal