Godot Setup for Windows+Mac

Engineering Ideas Clinic

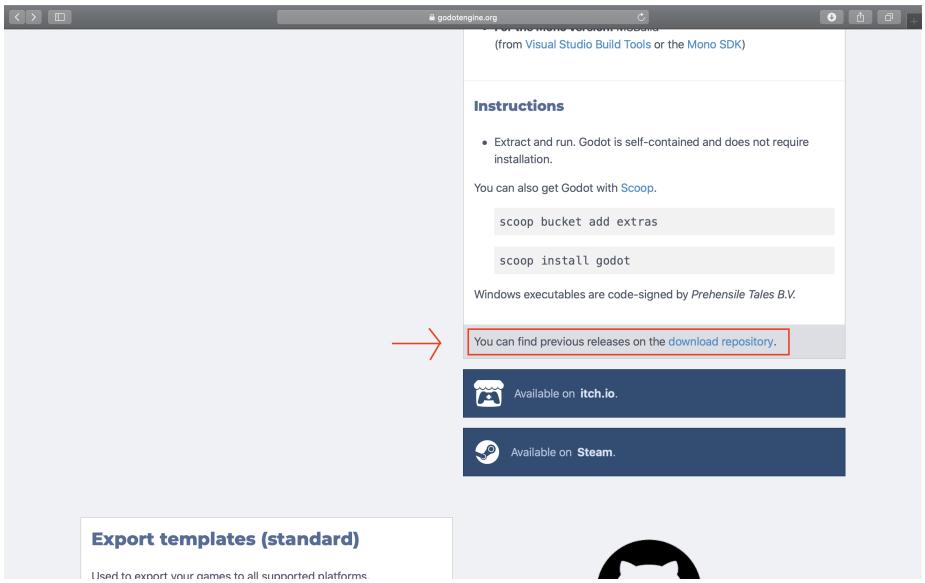
September 8, 2020

In order to complete this activity, we'll be using three pieces of software: the Godot game engine to simulate our spaceship sandbox, the VS Code text editor to write our code, and the Git version control system to share and merge code between teammates. This manual describes how to install these tools and connect them together on your local Windows/Mac development machine.

1 Setting up Godot

Godot is an open-source game engine that we will be using for this activity. The sandbox code for this activity was developed and tested using Godot version 3.2.1. While you're free to download a newer version of Godot, recognize that the activity staff will be less able to assist you should you encounter technical problems. At the very least, make sure your entire ship/team uses the same version of Godot.

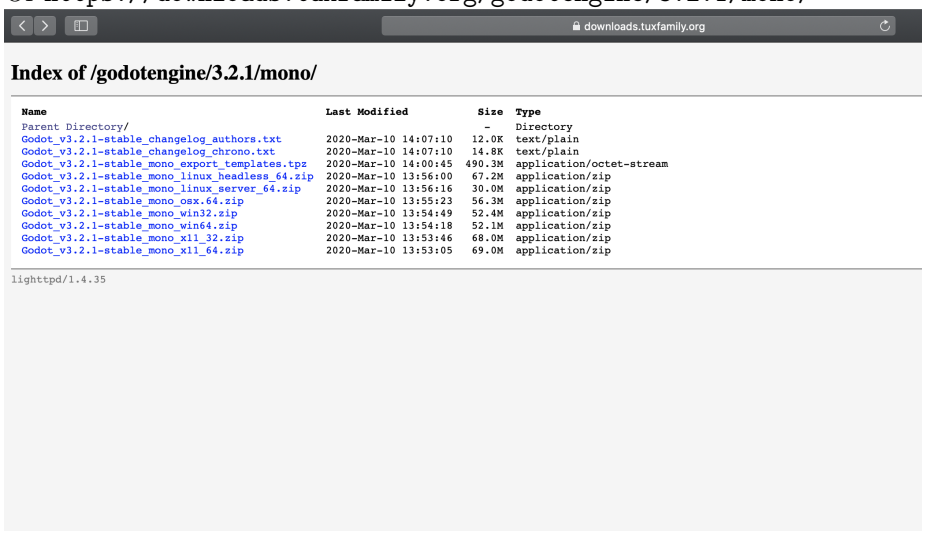
Go to
`https://godotengine.org/download/windows`



Since we are downloading the Mono version of Godot version 3.2.1

From there, navigate to index /godotengine/3.2.1/mono/

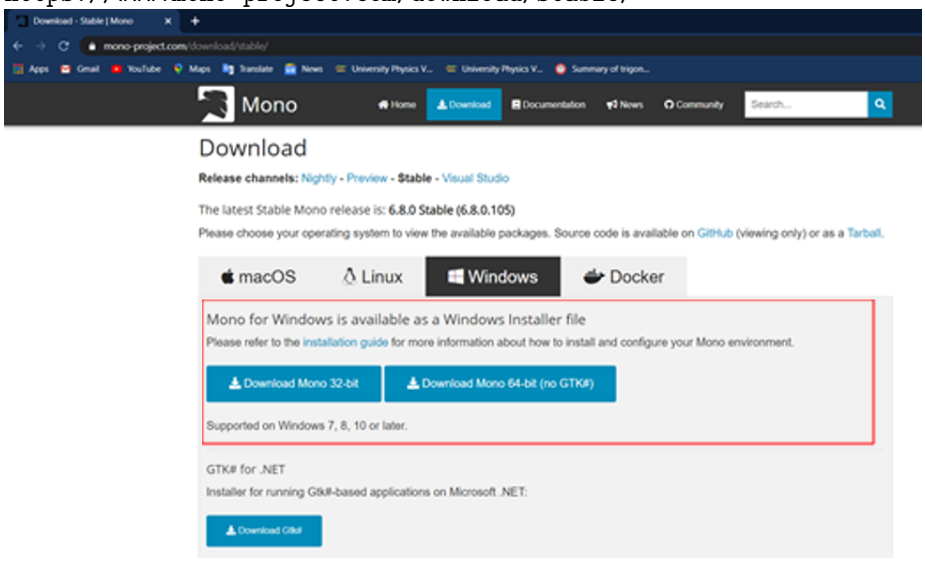
Or `https://downloads.tuxfamily.org/godotengine/3.2.1/mono/`



Select the correct download for your machine (OSX for Mac, win64 or win32 for Windows, etc.)

Godot comes as a self-contained .zip file. Once downloaded, extract Godot to a location on your computer. There is no installation step for the Godot editor.

Go to
`https://www.mono-project.com/download/stable/`



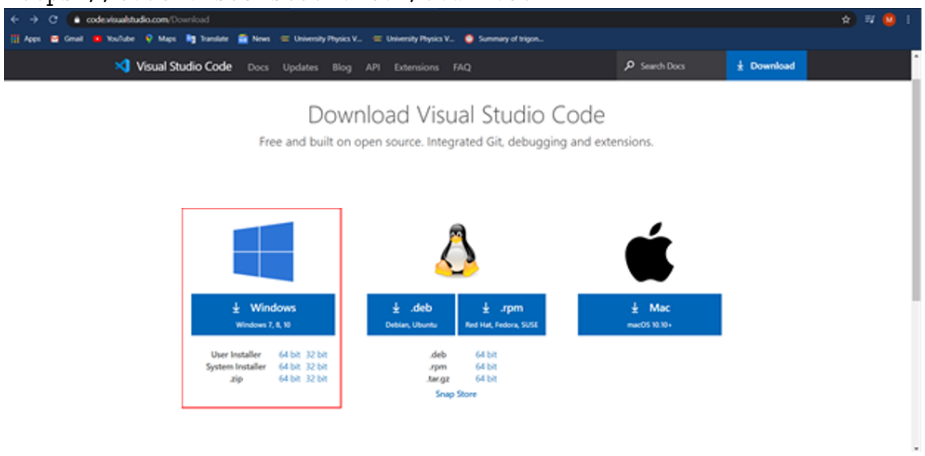
Download Mono SDK - Godot uses this to build the game executable in the background

Run the downloaded file to install the MonoSDK.

2 Setting up Visual Studio Code

VS code is a lightweight code editor with useful extensions

Go to
`https://code.visualstudio.com/download`

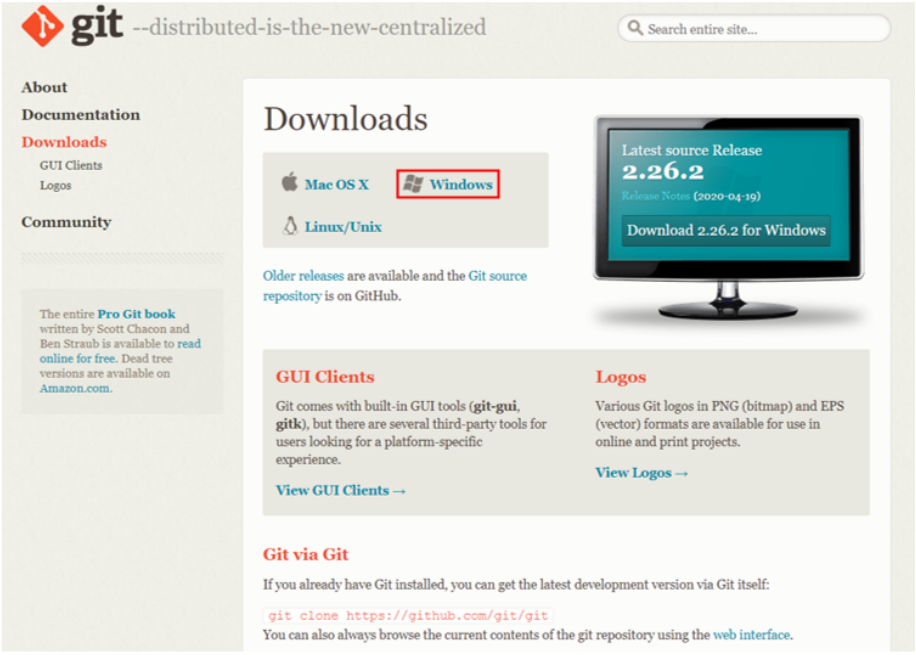


Download the appropriate .zip file for your system.

Once downloaded, extract the file to a desired location on your hard drive.

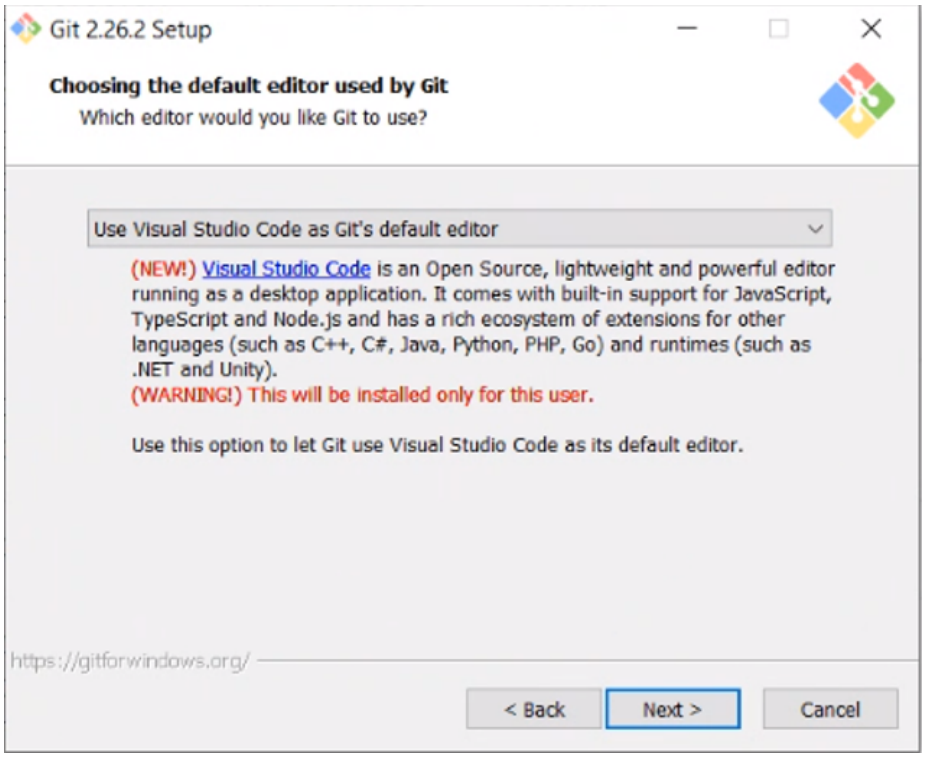
3 Setting Up Git

Go to
`https://git-scm.com/downloads`



Download Git for your operating system.

Once it is downloaded, run the file and click next/install to install it (the default settings will work), but you may choose to set Visual Studio Code as Git's default browser for convenience


Git 2.26.2 Setup
Choosing the default editor used by Git
Which editor would you like Git to use?
Use Visual Studio Code as Git's default editor
(NEW!) Visual Studio Code is an Open Source, lightweight and powerful editor running as a desktop application. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity).
(WARNING!) This will be installed only for this user.
Use this option to let Git use Visual Studio Code as its default editor.
https://gitforwindows.org/
< Back
Next >
Cancel

4 Cloning a repository

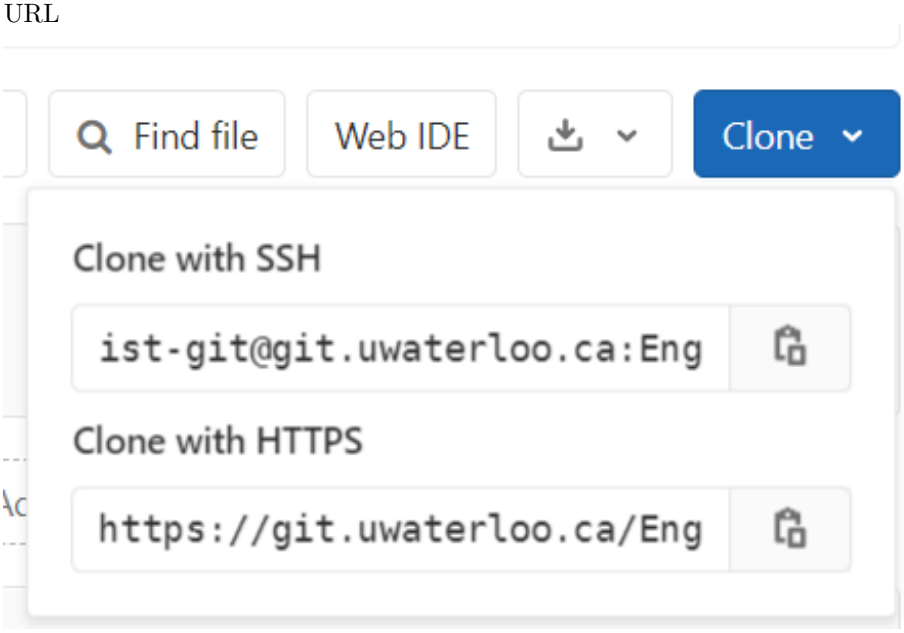
With Git installed, you can now clone a repository (repo). In this manual, we'll focus on cloning from the UWaterloo GitLab server. We have created an example repo for you to clone and test whether you've installed Git and Godot correctly. This example repo can be found at:
`https://git.uwaterloo.ca/EngineeringIdeasClinic/hellogodot-examplerepo`
When your local Git client communicates with a remote server, there are two common protocols for authenticating yourself to the server and proving you have permission to download a particular repo: HTTPS and SSH.

4.1 Cloning with HTTPS

Cloning a GitLab repo via HTTPS is akin to logging in to GitLab using your web browser. You'll be prompted to input your WatIAM username and password.

Go to
`https://git.uwaterloo.ca`
Sign in

Go to the desired repository's browser page and copy the "Clone with HTTPS" URL



Open Git Bash and navigate to the file location that you want to clone the repository to

In Git Bash, type

```
git clone <insert repo URL here>
```

inserting the appropriate repo HTTPS URL you copied previously.

When prompted, enter your waterloo credentials (your WatIAM credentials not your email).

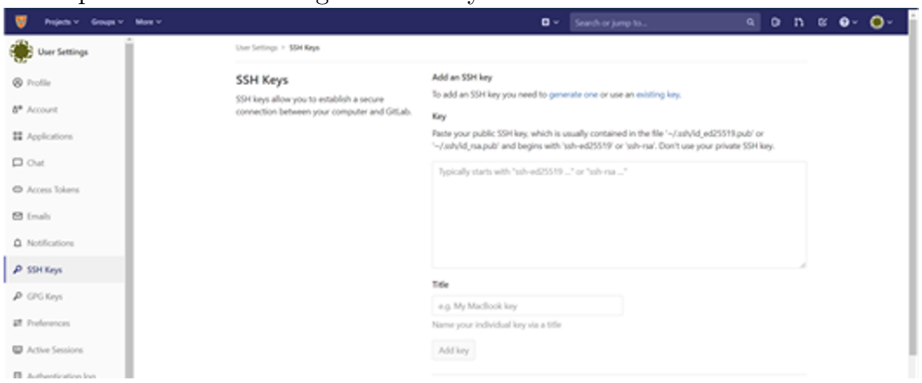
4.2 Clone with SSH

If the HTTPS method of authentication does not work for you, follow the following steps for SSH authentication (you can skip these steps if the waterloo credentials were successful). With this approach, you will generate a private/public pair of encryption keys and upload the public key to the UWaterloo GitLab server. Your local Git client will then use the private key (stored on your local machine)to authenticate you against that publiclly stored key.

Go to
`https://git.uwaterloo.ca/help/ssh/README#generating-a-new-ssh-key-pair`
Follow the steps for "generate a new SSH key pair" and generate a ED25519 SSH key

Go to `https://git.uwaterloo.ca`
Sign in

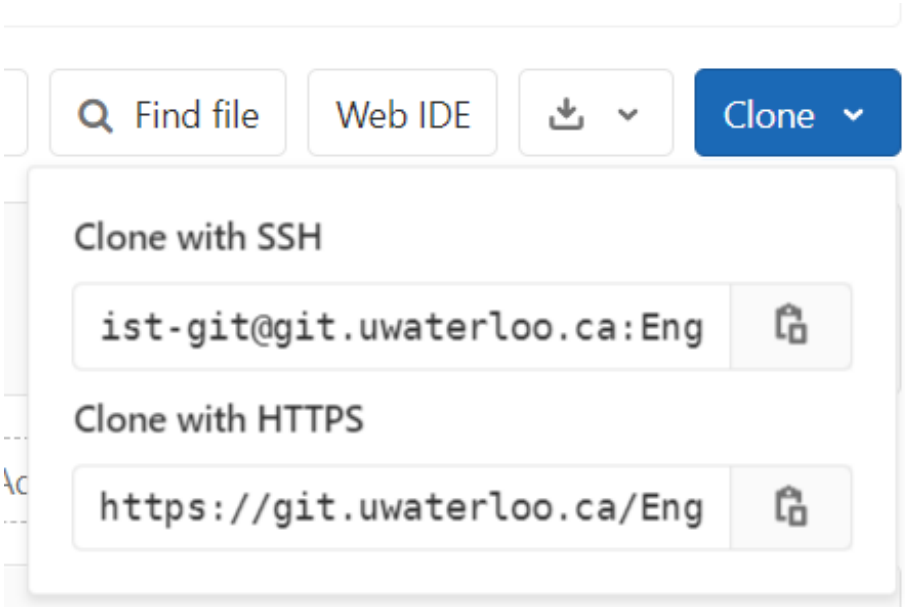
Go to profile icon → settings → SSH keys



Following the instructions under "Key", go to the file where the key was saved.

Copy the key text and paste into the text box to add the key to your profile.

With your public key successfully saved to your GitLab profile, go to the desired repository's browser page and copy the "Clone with SSH" URL.



In Git Bash, navigate to the file location that you want to clone the repository to.

In Git Bash, type

```
git clone <insert repo SSH URL here>
```

making sure to insert the SSH URL you just copied from the GitLab web page

If you set a password when you were generating your SSH keys (highly recommended), you will be prompted to enter the password during the clone operation. If you do not create passwords with your SSH keys, anyone who acquires a copy of your private keys can pretend to be you.

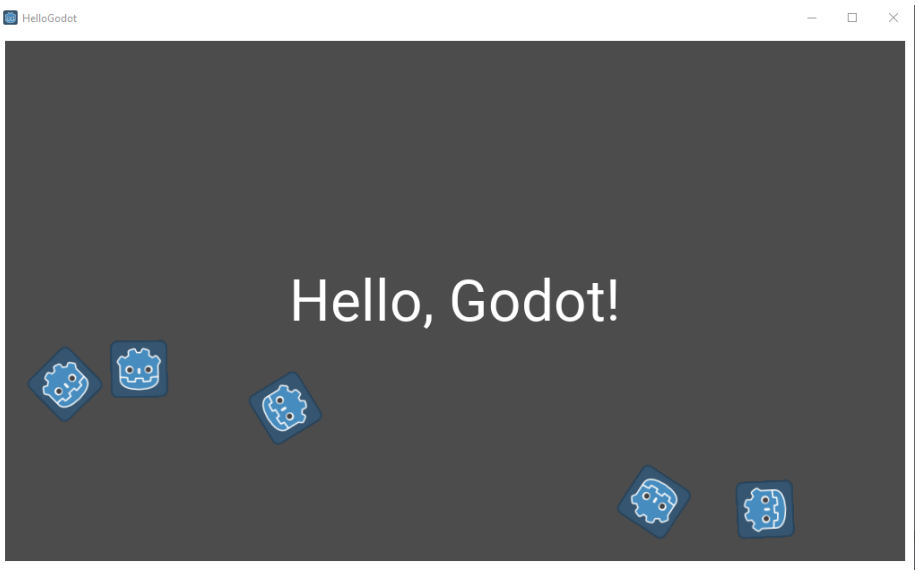
4.3 Testing Godot

Once you have cloned the example repo, open the Godot editor.
You should be presented with the Godot Project Manager.

Select Import and open the project.godot file from the example repo you cloned.

Godot will open the example project.

Press F5 (or click on the Play button in the top right) in order to play the starting scene.



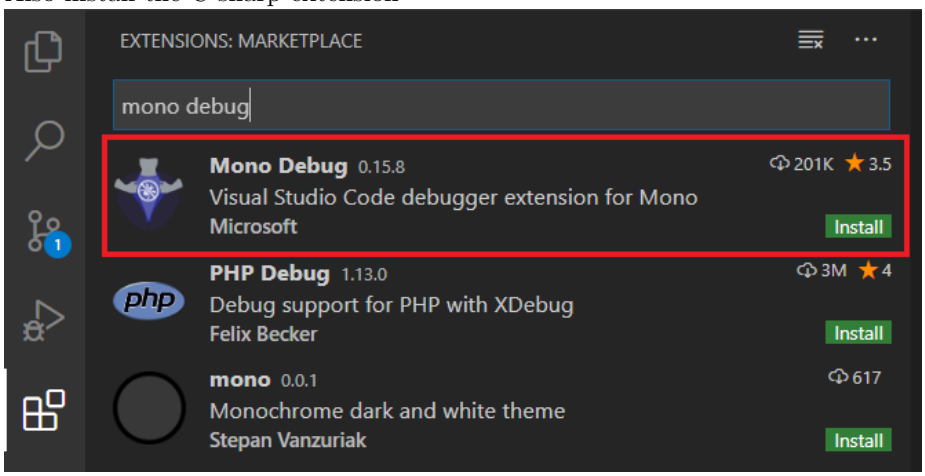
A new window will open showing the running game process. If you see a set of bouncing Godot logos and the words "Hello, Godot!" you've installed all the necessary tools correctly!

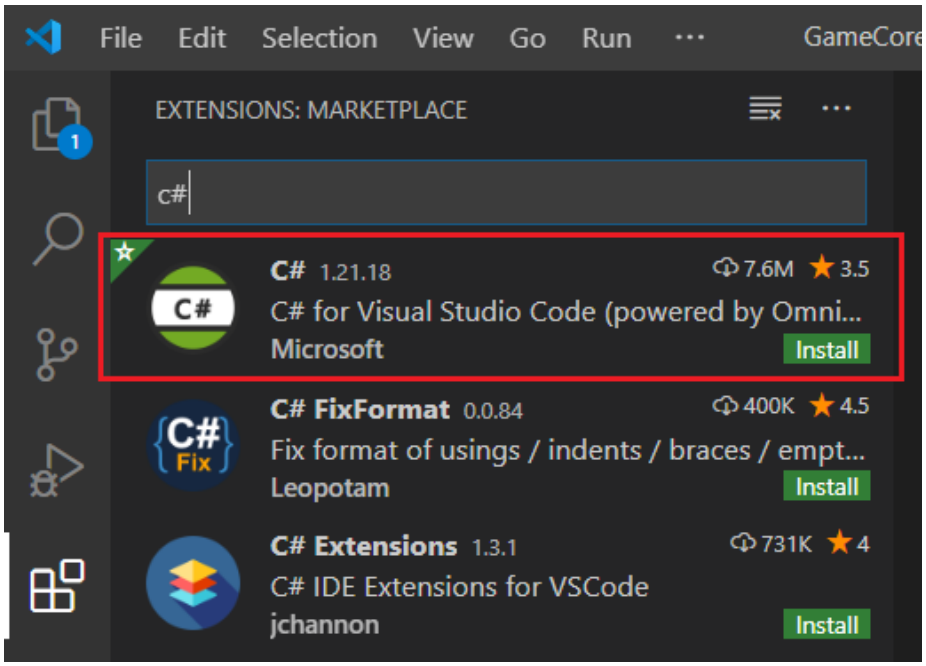
5 Connecting VS Code with Godot

Not only can VS Code be used to write code that will be compiled into game logic, VS Code can also be used to debug running Godot processes and help track down errors/bugs. This section describes the steps necessary to connect VS Code to Godot for debugging and to provide intellisense/autocomplete in order to make coding easier.

Open VS Code and navigate to the extensions tab, search for mono debug extension, and click install

Also install the C sharp extension

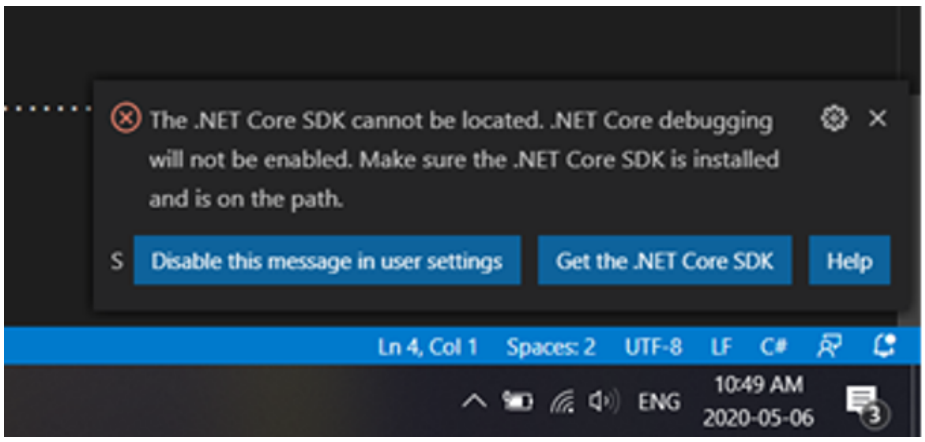




note: when installing the C sharp extensions for VS Code or compiling code, you may encounter this error

```
The reference assemblies for .NETFramework,Version=v4.7 were not found
```

or



in which case go to
`https://dotnet.microsoft.com/download/dotnet-framework/net47`
and download the Developer Pack and make sure to download version 4.7 and **not** the latest version (4.8)

For Mac users go to
`https://dotnet.microsoft.com/download/dotnet-core`
and download .NET Core 3.1

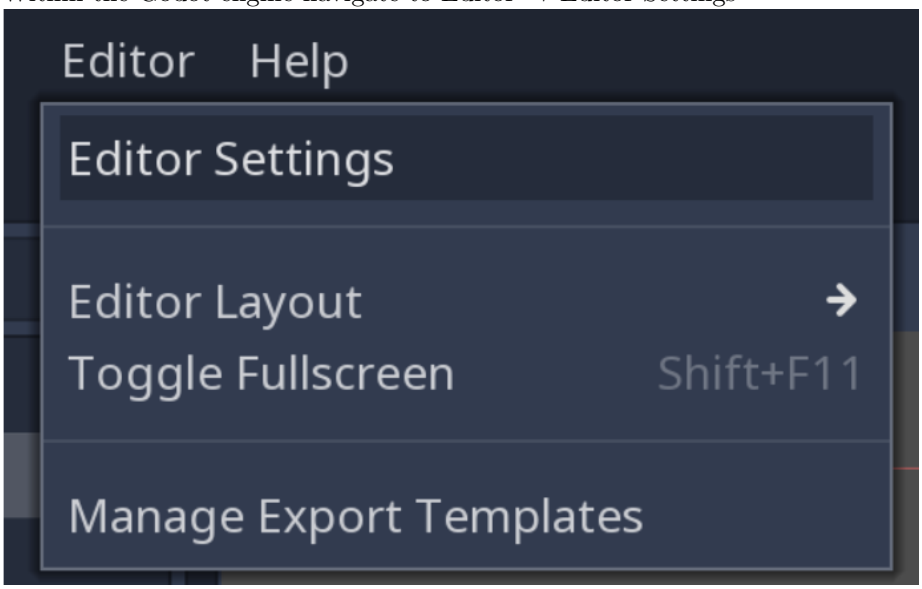
Reboot your computer after installing

Navigate to the repository you cloned and open the project.godot file with the godot engine

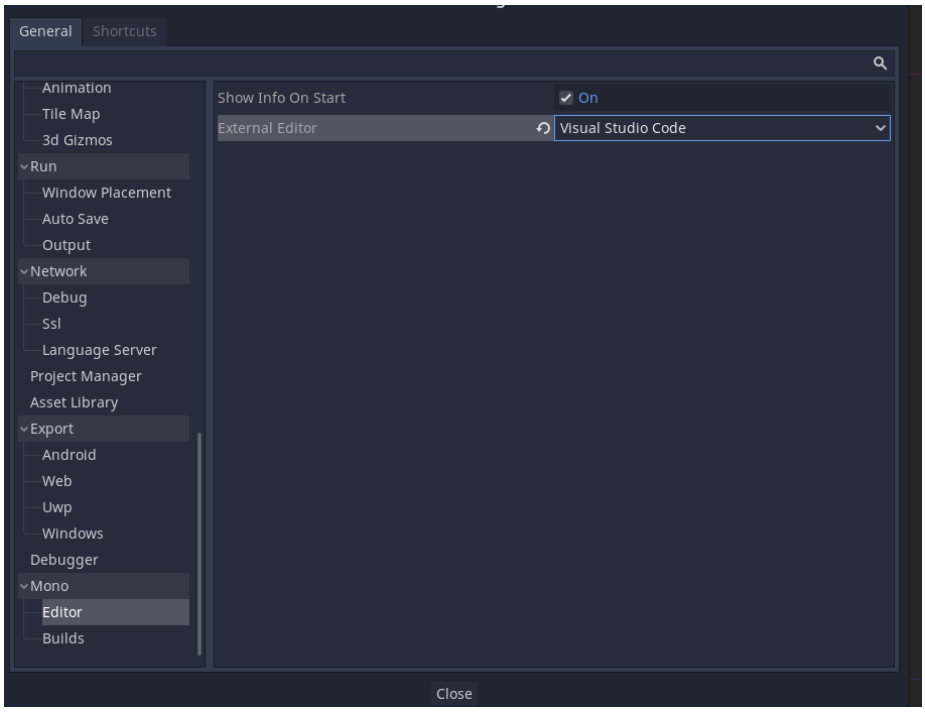
You may be required to open with → more apps → look for another app on this PC and then browse to find the godot engine executable

`ie. Godot_v3.2.1-stable_mono_win64.exe`

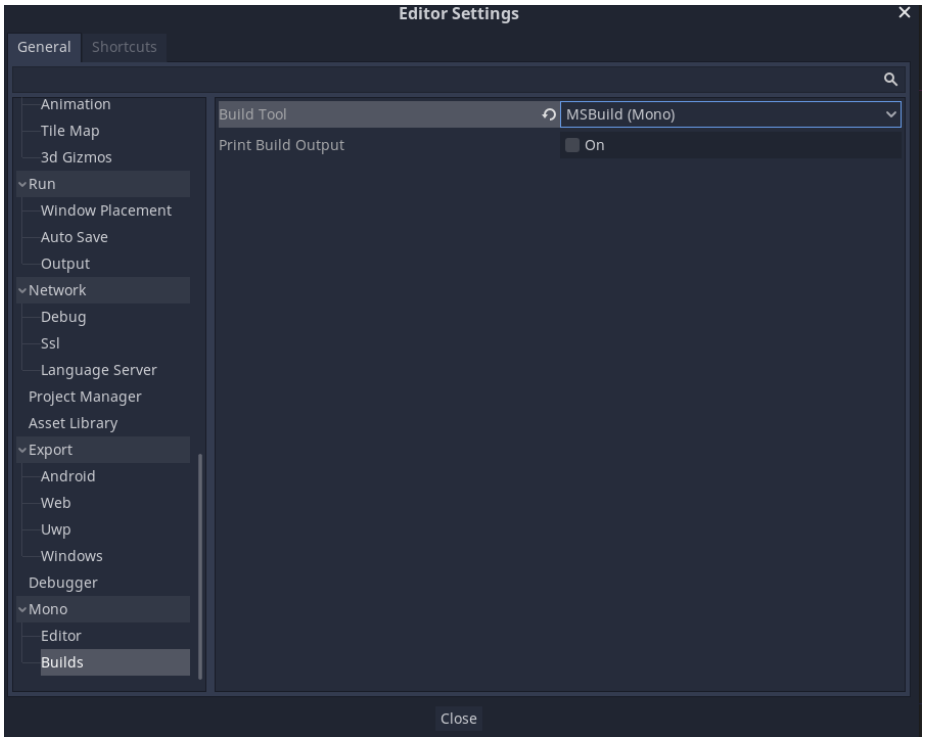
Within the Godot engine navigate to Editor → Editor Settings



Then scroll down to Mono → Editor and select Visual Studio Code as External Editor



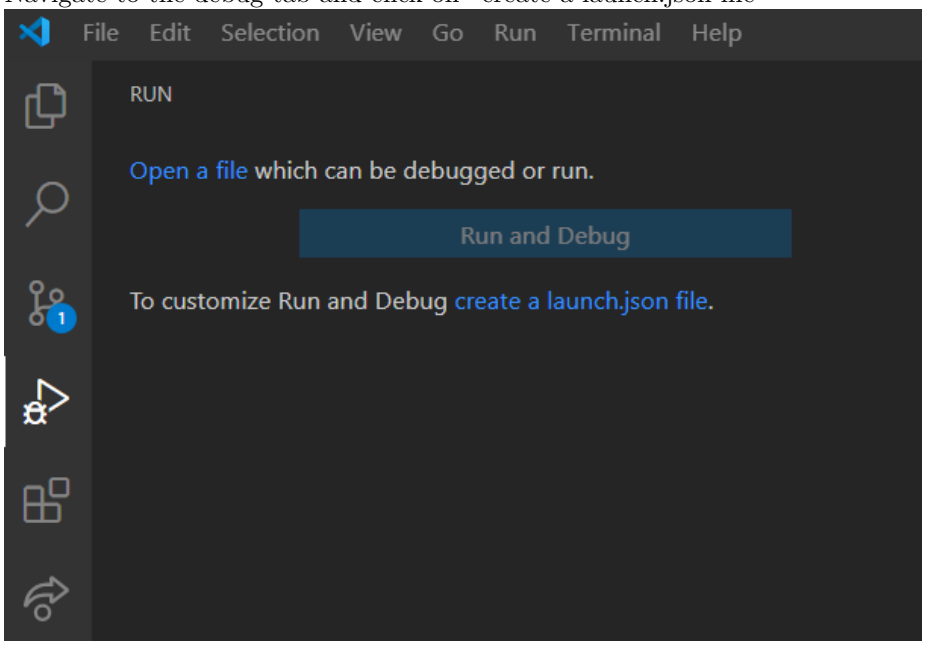
Then go to Mono → Builds and select MS Build (Mono) as the Build tool



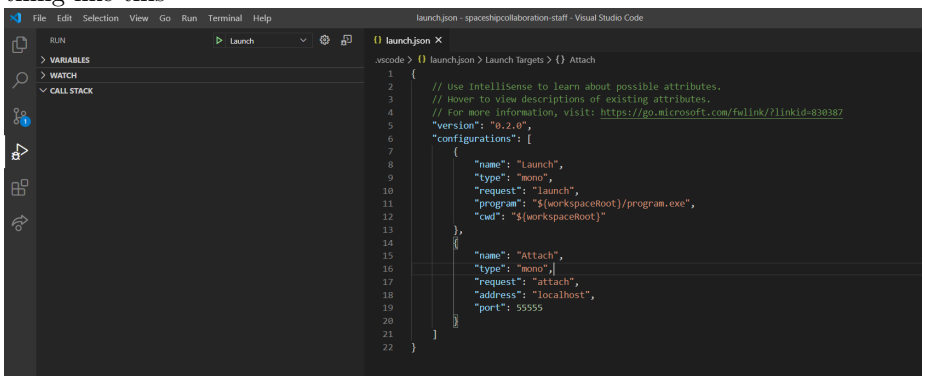
Exit the settings menu and click the play button in the top right, this should launch the game!

6 Debugging with VS Code and Godot

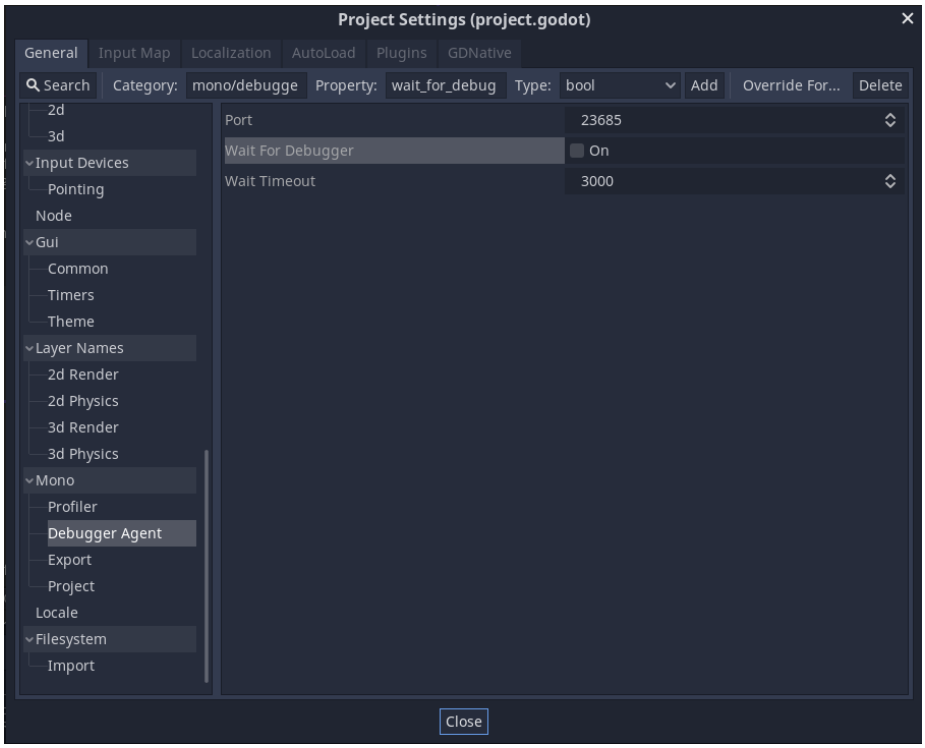
Navigate to the debug tab and click on "create a launch.json file"



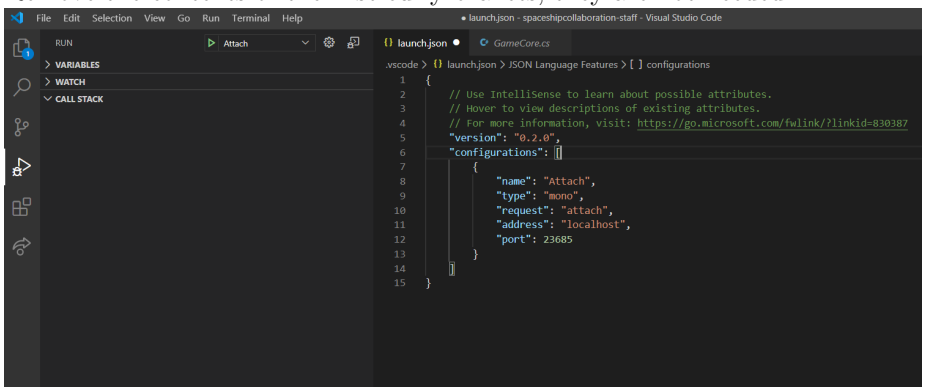
Select C sharp mono as the Environment, it will create a file that looks something like this



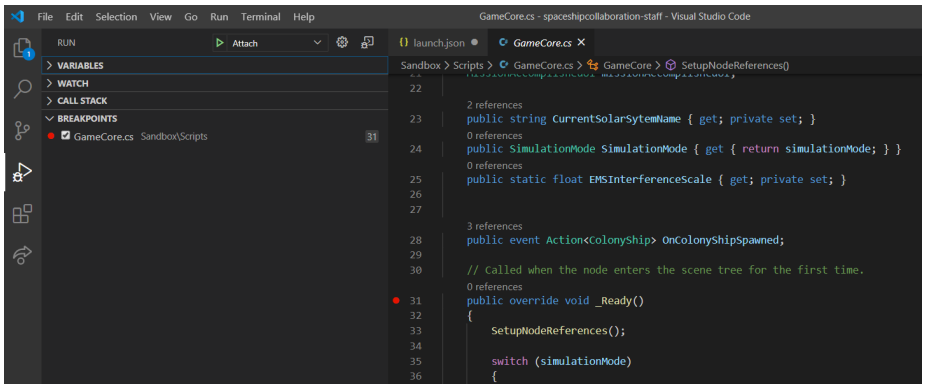
Go to Godot and navigate to project → project settings and then scroll down to mono → debug agent



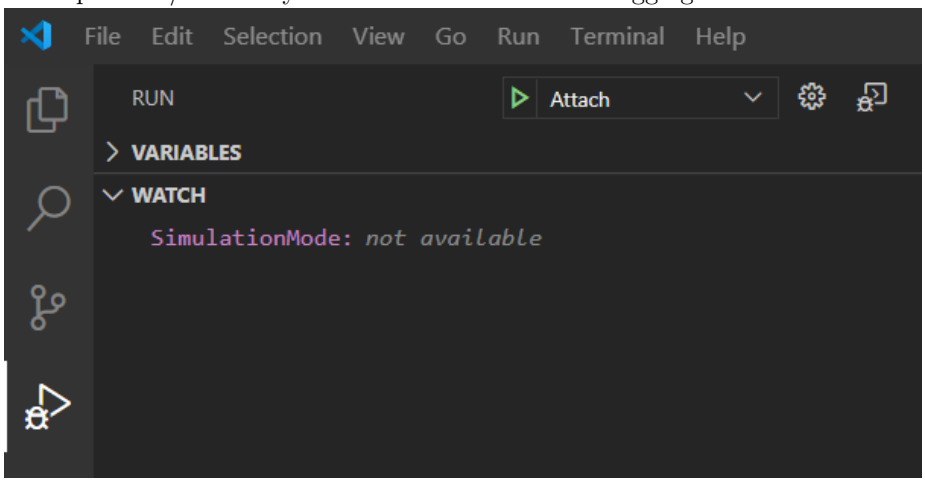
Copy the port number and then add it to the launch.json file beside port
Remove the contents of the first curly brakets, they are not needed



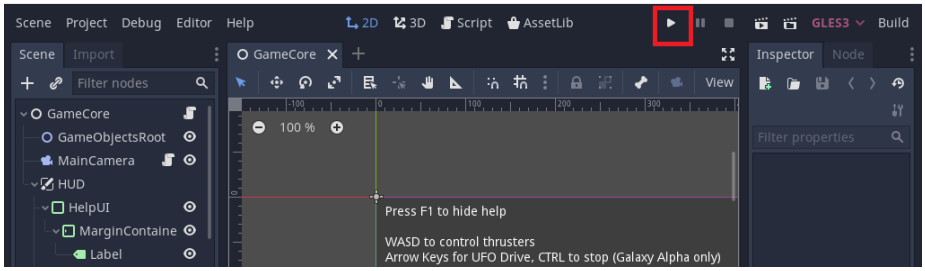
Now you are able to start debugging!
Add breakpoints by clicking the editor magin so that a red dot appears, you can keep track of them under the breakpoints tab



To add a watch expression open the watch tab and click the plus sign and add the expression/variable you want to watch while debugging

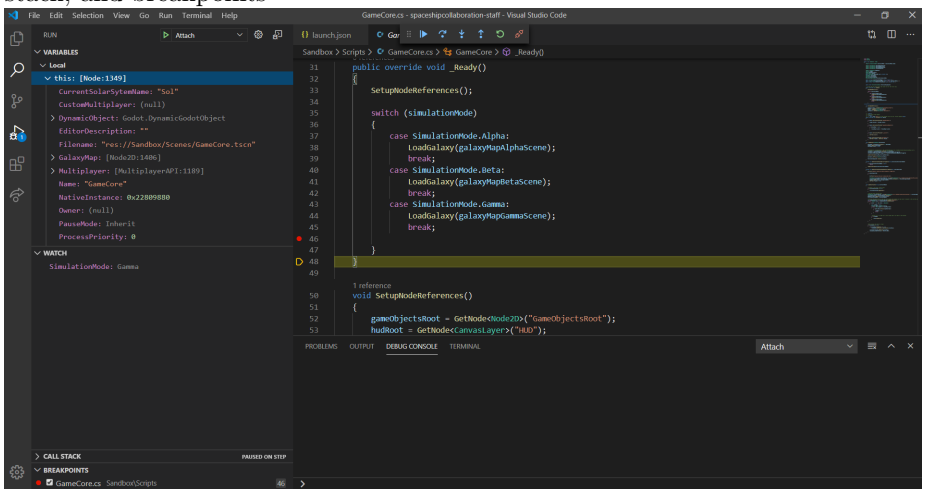


In order to start debugging, click the green arrow at the top left
Then you must run the Godot project by clicking the arrow in Godot as well



Now when you go back to VS Code, you will have a tool bar appear at the top which allows you to pause, continue, step over, step into, step out, restart, and disconnect

On the left you will be able to keep track of variables, watch expressions, call stack, and breakpoints



You are now all setup for this activity, have fun!