**Development environment setup (Windows)**

**What is WSL?**

**WSL** stands for **Windows Subsystem for Linux**. It's a feature introduced by Microsoft that allows you to run a Linux environment directly on your Windows machine. This means you can use familiar Linux command-line tools and applications alongside your existing Windows programs. To ensure that your code runs on various machines using Unix-based systems like Mac and Linux, you'll find WSL to be immensely beneficial.

**How to install WSL?**

Installing WSL is very easy, just open your Windows Terminal (comes preinstalled in Windows 11, available on Microsoft Store on Windows 10) and type

wsl –install

After your system reboots, the Terminal will open automatically and proceed with the installation.

You will be asked to provide a new UNIX username and an associated password.

**Making Visual Studio Code Compatible with WSL**

The next step is to ensure our VS Code is compatible with WSL.

Open up VS Code and navigate to the Extensions tab. Here, look for the Remote Development extensions and proceed to install each of them. This will enable VS Code to operate with WSL seamlessly. A new icon will appear on the bottom left of the screen called Open a Remote Window.

There's even an easier way to connect:

In the Windows Terminal, open up a new Ubuntu terminal.

Create a new folder by typing in:

mkdir solidity-course

Change the directory into the newly created folder

cd solidity-course/

Open VS Code inside the folder by typing in

code .

WIN! You just created a new instance of VS Code that uses WSL. Everything is correct if on the bottom left of your screen you see a small banner with the text WSL Ubuntu.

**Important: When you conduct your projects from a folder inside Windows (ex. Development) inside your documents, it's crucial to know that the WSL console will only access local files inside the WSL instance. Therefore, it's recommended to keep files inside the WSL instance for faster communication and convenience.**