YADEG

Yet Anoyher Development Environment Guide

Version 1.1

Manuel Alves

February 2024

Contents

1 Introduction

Welcome! February 19, 2024 10:57am Z

Free stuff!

- Setting Up GitHub Student and GitHub Copilot as an Authenticated Student Developer
- Como obter GitHub Copilot gratuito para estudantes e professores

---

- Microsoft Azure for Students
- Microsoft Azure para estudantes

2 Development Environment Guidelines

It is time to set up your work computer. The following guidelines are not mandatory, but rather a recommendation that will prepare your computer for development. Follow the sequence as some operations depend on previous requirements.

2.1 Windows Account

If you want to associate a Windows account with your profile, you should already have prepared a personal Microsoft account [1]. Windows does not allow to log in with an organisation or school account if your computer does not belong to that organisation's Active Directory, which is our case.

2.2 Terminal

The Windows Terminal Preview is a preview build of the Windows Terminal application that contains the latest features as they are developed. It is a modern, fast, efficient, powerful, and productive terminal application for users of command-line tools and shells like Command Prompt, PowerShell, and WSL. You can install it from the Microsoft Store or from the GitHub release page.

Your computer is a Windows machine that you will use for office applications, as well as MS Teams, Zoom, Wrike among others. For development, we find that it is useful to start by installing "*Terminal Preview*". Windows already comes with the "Terminal" app, but we recommend that you remove it and install "*Terminal Preview*" as it brings useful features for our tasks.

1. Uninstall the windows app "Windows Terminal" so that we do not use it instead of "Windows Terminal Preview".

   1.1 Open "Windows Settings" (Windows key + I).

   1.2 Apps (on the left).

   1.3 Installed apps.

   1.4 Search for *terminal*.

   1.5 Open the "..." menu on the right and select "Uninstall".

2. From the Microsoft Store, search and install "Windows Terminal Preview".

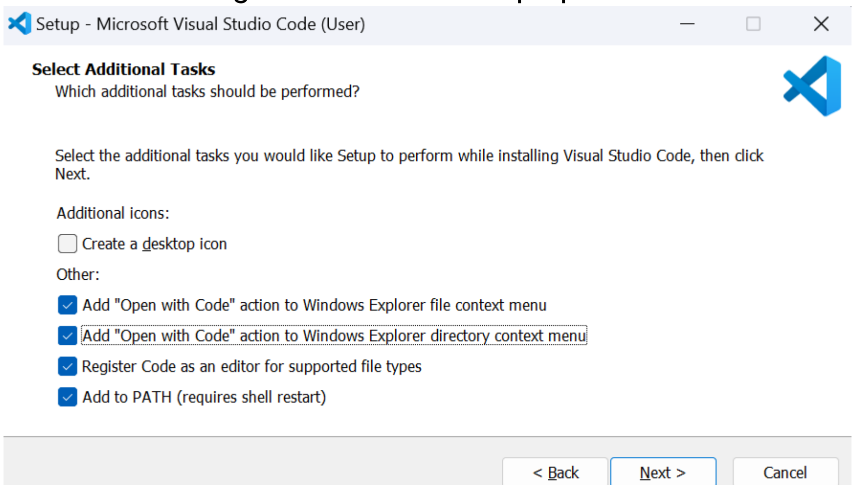
1 `https://account.microsoft.com/account/Account`

2.3 Visual Studio Code

Visual Studio Code is a code editor to build and debug modern Web and cloud applications. It is free and available on your favorite platform: Linux, macOS, and Windows. It has features such as IntelliSense, which provides smart completions based on variable types, function definitions, and imported modules. It also has built-in Git commands and is extensible with a wide range of extensions available.

1. From https://code.visualstudio.com/download download a Windows X64 version (user or system) and install it. See Figure 1.

Figure 1: VSCode setup options



Setup - Microsoft Visual Studio Code (User)

Select Additional Tasks
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Visual Studio Code, then click Next.

Additional icons:

- [ ] Create a desktop icon

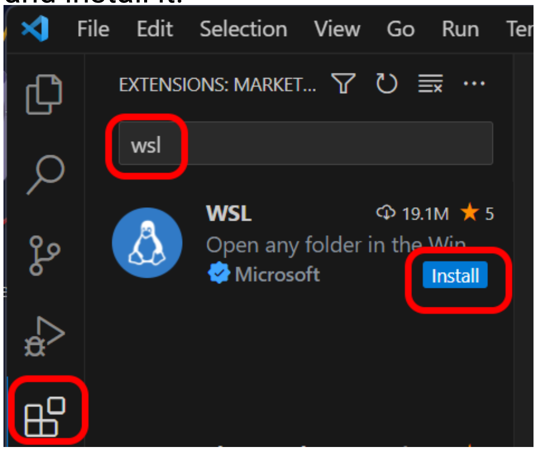
Other:

- [x] Add "Open with Code" action to Windows Explorer file context menu
- [x] Add "Open with Code" action to Windows Explorer directory context menu
- [x] Register Code as an editor for supported file types
- [x] Add to PATH (requires shell restart)

< Back | Next > | Cancel

2. Launch Visual Studio Code (VSCode) and check the welcome screen.

3. On the left side, click the extentions button , search for *WSL addon* and install it.


File Edit Selection View Go Run Ter
EXTENSIONS: MARKET...
wsl
WSL
19.1M ★ 5
Open any folder in the Win
Microsoft
Install

4. Close VSCode for now.

More information about Vscode can be found here: https://code.visualstudio.com/learn. Do not forget to check https://code.visualstudio.com/learn/collaboration/live-share for collaboration on coding.

2.4 Windows Subsystem for Linux (WSL)

The Windows Subsystem for Linux (WSL) lets developers run a GNU/Linux environment – including most command-line tools, utilities, and applications – directly on Windows, unmodified, without the overhead of a traditional virtual machine or dual-boot setup. You can choose your favorite GNU/Linux distribution from the Microsoft Store and run common command-line tools such as grep, sed, awk, or other ELF-64 binaries. You can also run Bash shell scripts and GNU/Linux command line applications including tools like vim, emacs, tmux, and languages like NodeJS, Javascript, Python, Ruby, C/C++, C#, Rust, Go, etc.

First, you must check if the Hyper-V feature is enabled (virtualization). Go to **Control Panel > Programs > Programs and Features > Turn Windows features on or off** and make sure that Hyper-V is selected[2]. If not, check it. After enabling it or making sure that Hyper-v is available, you can start the installation of WSL. When configuring the Linux distribution, you are asked to create a user. DO NOT create a user named "root". Whenever you need root access, you should use `sudo` or `su`.

1. Open a command line as *Administrator* and execute[3]:
   ```
   wsl --install -d Ubuntu-22.04
   ```
2. After the installation exit all terminal windows.
3. Open an Ubuntu terminal (relaunch the terminal and look for the drop-down options).
4. Update the Linux distribution by executing:
   ```
   sudo apt update && sudo apt -y upgrade
   sudo apt dist-upgrade
   ```

Now you can use *Terminal Preview* to launch command windows and Linux terminals.

2 If the option is not available you may need to activate virtualization on the computer BIOS.

3 Follow the steps in this page if you get the error WslRegisterDistribution failed with error: 0x8...

2.5 Onedrive Setup

OneDrive is an internet-based storage platform by Microsoft. Think of it as a hard drive in the cloud, where you can save files, access them on multiple devices, and share them with others.

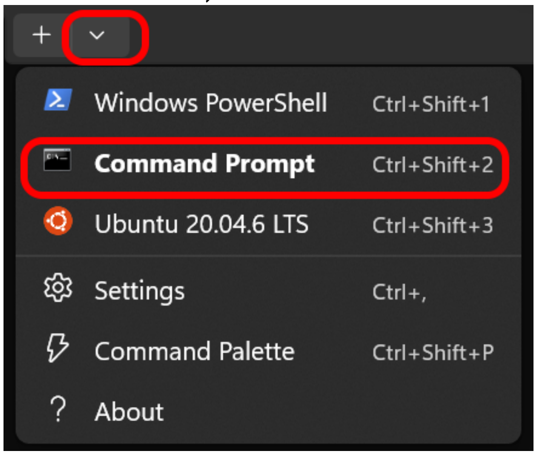
1. On Windows, launch OneDrive and login with your DTx credentials:

   ```
   Username: a123456@uminho.pt
   Password: <your password>
   ```

   Make sure onedrive is running.

   The following steps are optional and are shown here just to illustrate how to map a folder from one location to another (simpler) path. This is not needed for the rest of this guide!

2. Using "*Terminal Preview*" open a command line (NOT a powershell command line)

   
   +
Windows PowerShell Ctrl+Shift+1
Command Prompt Ctrl+Shift+2
Ubuntu 20.04.6 LTS Ctrl+Shift+3
Settings Ctrl+,
Command Palette Ctrl+Shift+P
About

   and execute (do not forget the "" in the following commands):

   ```
   cd "C:\Users\%username%\OneDrive - Universidade do Minho"

   mkdir github

   MKLINK /J c:\github ^
   "C:\Users\%username%\OneDrive - Universidade do Minho\github"

   cd \

   dir
   ```

you should see a github directory at the root of C drive but it is maped to a onedrive directory enabling the automatic backup of your files to your onedrive account.

2.6 Connect to GitHub

GitHub is a website and service that provides a cloud-based Git repository hosting service. It makes it easier for individuals and teams to use Git for version control and collaboration. Provides distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and Wikis for every project.

1. Create a personal account on GitHub with your institutional email. See Signing up for a new GitHub account.

2. Configure ssh access to GitHub by following the instructions in Connecting to GitHub with SSH. If you want to set up the access, you just need to follow the two links on the page: Generating a new SSH key and adding it to the ssh-agent. See below the necessary steps.

   IMPORTANT:

   - Please read carefully ALL the steps before copying and pasting because you must use your username instead of *myUser* and (next point);
   - When executing the command "ssh-keygen -t ed25519 -C ...", it asks the file to save the output. ENTER **/home/*myUser*/.ssh/id.rsa** REPLACING *myUser* with your Linux username;
   - When asked for a passphrase, leave it empty, just press ENTER.

   Did you read the previous instructions carefully? Are you sure?
   OK!

   ```
   ssh-keygen -t ed25519 -C "your_email@alunos.uminho.pt"
   Generating public/private ed25519 key pair.
   Enter file in which to save the key
    ↪  (/home/mylinuxuser/.ssh/id_ed25519):
   ```

   Check that the files id.rsa and id.rsa.pub were created inside ~/.ssh/

3. In windows file explorer:

   - Enter the address \\wsl$
   - Navigate to \\wsl$\Ubuntu-22.04\home\myUser\.ssh (replace myUser with your username)
   - Open the file id.rsa.pub with Notepad

4. Open https://github.com/ in your browser and sign in. In your profile choose settings.
5. On the left, choose "SSH and GPC keys".
6. Click the button "New SSH Key", give it a name, paste the contents of the file "id.rsa.pub" you have opened in Notepad, and click "Add SSH Key".
7. Make the login automatic when you open an Ubuntu terminal:

   ```
   cd ~
   code .bashrc
   # scroll to the end of the file and add the following lines:

   eval "$(ssh-agent -s)"
   ssh-add -k ~/.ssh/id.rsa
   ssh -T git@github.com

   # The following alias is not related to the login process.
   ↪  Just a shortcut to change the directory.
   alias dtxcode="cd /mnt/c/github"
   ```

8. Save the file, close the terminal, and open a new one. Now, every time you open a new Ubuntu terminal, you are authenticated to GitHub. Also, if you type ***dtxcode*** or just ***dtx+TAB*** your working directory will be changed to `/mnt/c/github` which is `C:\github` which is mapped to onedrive.

For guidelines on how to use GitHub, see Section 3.

2.7 Docker

Docker is an open platform for developing, shipping and running applications. It enables you to separate your applications from your infrastructure so that you can deliver software quickly. With Docker, you can manage your infrastructure in the same way that you manage your applications. Docker provides the ability to package and run an application in a loosely isolated environment, called a container. The isolation and security allow you to run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host[4].

4 https://docs.docker.com/get-started/overview/

On your Windows machine (not WSL):

1. With your DTx email, create an account in `https://hub.docker.com/signup`.
2. Download and install Docker Desktop Windows version, accepting defaults.
3. After rebooting, start Docker Desktop from the Windows Start menu.
4. Log (top right corner) with the credentials you got from Step 1.
5. From the Docker menu, select Settings and then General. Make sure the "Use WSL 2 based engine" check box is checked.
6. From the Docker menu, select Settings, resources, and then "WSL Integration". Enable everything.
7. Launch Ubuntu from *Terminal Preview* and execute "docker –version" to check that you can access docker from linux.
8. In a Virtual Machine (VM), not WSL, you might need to execute the following commands to manage docker without the need to *sudo*:

   ```
   sudo groupadd docker
   sudo usermod -aG docker $USER
   ```

9. On a VM execute `sudo reboot`, on WSL just exit all terminals.

If you are new to Docker and want to know more, you can start with:

- Docker Crash Course for Absolute Beginners
- *https://docker-curriculum.com/*

2.8 Python Environments

A Python environment is the context in which a Python program runs and consists of an interpreter and any number of installed packages. It can be created using the "venv" module, which supports creating lightweight "virtual environments" or using conda, each with their own independent set of Python packages installed in their site directories. This allows you to isolate external dependencies and avoid dependency conflicts between different projects.

Preparation of a Python environment.

1. On windows *Terminal Preview* open a new ubuntu terminal and execute:

   ```
   sudo apt update && sudo apt -y upgrade
   sudo apt install curl -y
   cd /tmp
   ```

2. In a browser open `https://repo.anaconda.com/archive/` and look for the latest Linux distribution. Copy the link and take note of the SHA256 fingerprint[5]. Execute the following steps:

   ```
   curl --output anaconda.sh \
   https://repo.anaconda.com/archive/AnacondaXXX-Linux-x86_64.sh

   # Check the fingerprint
   sha256sum anaconda.sh

   # If the fingerprint is OK (matches the one on the website),
   run the installation script, and accept the defaults. At the
   end, reply 'YES' to let conda init on Linux shell startup:
   bash anaconda.sh

   source ~/.bashrc

   # Check the installed version
   conda --version

   # Update
   conda update conda

   # Clean up
   rm anaconda.sh
   ```

2.9 Managing Python Environments with Conda

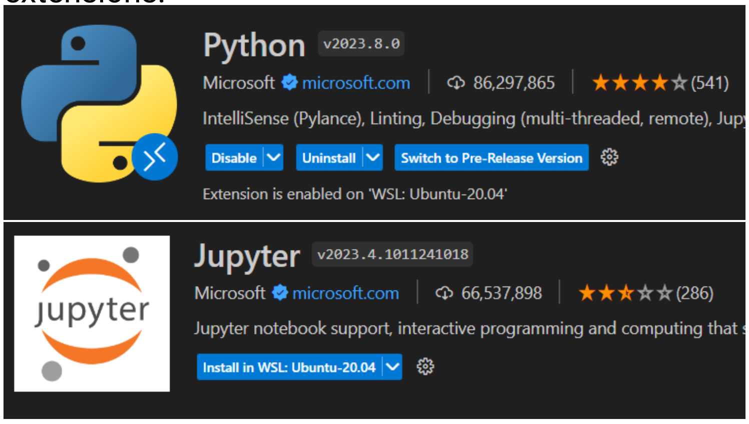
Conda is an open source package management system and environment management system running on Windows, MacOS, and Linux. It quickly installs, runs, and updates packages and their dependencies. Conda easily creates, saves, loads, and switches between environments on your local computer. It was created for Python programs but can package and distribute software for any language.

5 At the time of this writing was https://repo.anaconda.com/archive/Anaconda3-2023.03-1-Linux-x86_64.sh

Let us create an environment[6] with a specific Python version. As an example, we will create an environment to develop Python code related to MQTT. Our environment will be named "mqtt" (flag -n).

```
conda create -n mqtt python=3.11
conda activate mqtt

# We need ipykernel to use this env in jupyter notebooks
python -m pip install ipykernel
python -m ipykernel install --user --name mqtt --display-name "MQTT"

# install other required packages
python -m pip install tb-mqtt-client
...

# Now launch vscode
cd /mnt/c/github
mkdir python
cd python
code .
```

1. "Trust the authors of the files in this folders" and install recomended extensions:


Python v2023.8.0
Microsoft microsoft.com 86,297,865 ★★★★☆ (541)
IntelliSense (Pylance), Linting, Debugging (multi-threaded, remote), Jup

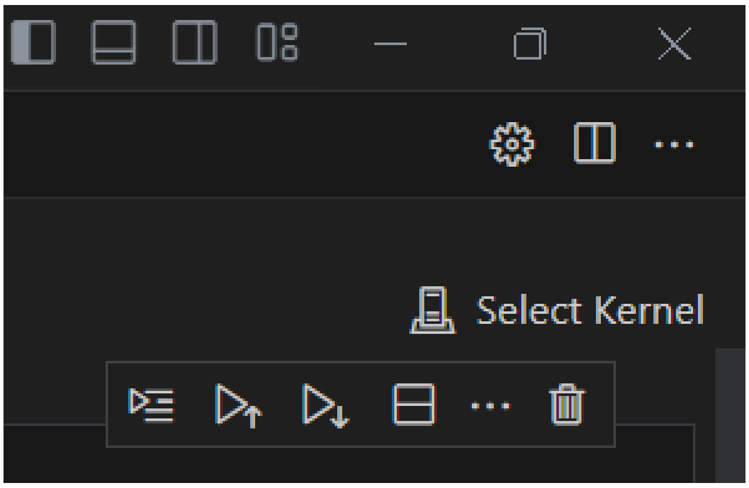
Extension is enabled on 'WSL: Ubuntu-20.04'

Jupyter v2023.4.1011241018
Microsoft microsoft.com 66,537,898 ★★★☆☆ (286)
Jupyter notebook support, interactive programming and computing that

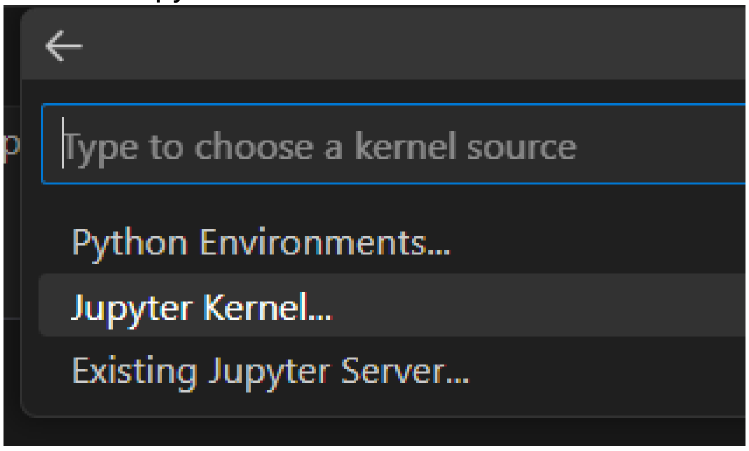
2. Create a jupyter notebook with the filename "hello.ipynb".

3. Select the kernel from the mqtt environment.

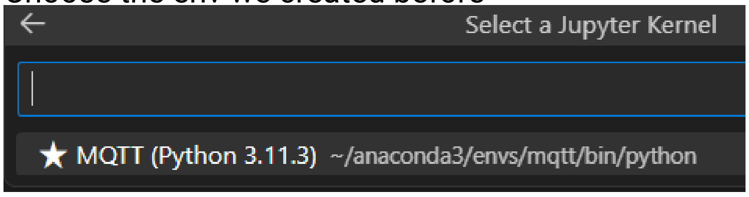
6 https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html



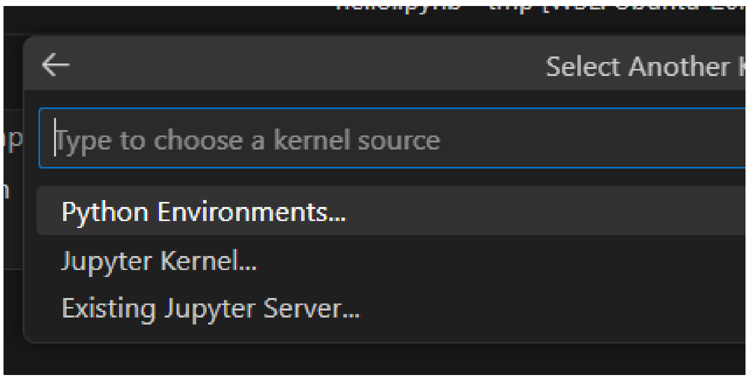
4. Choose Jupyter Kernel



5. Choose the env we created before



6. You could also select a Python environment (after closing VSCode and opening again)

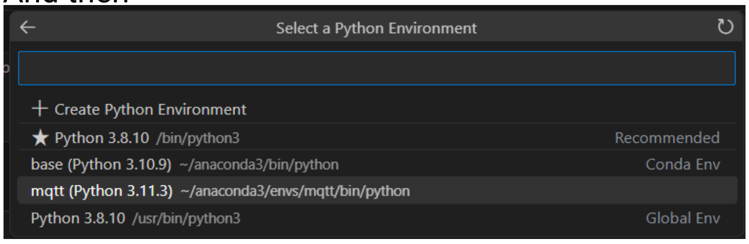


Select Another K

Type to choose a kernel source

Python Environments...
Jupyter Kernel...
Existing Jupyter Server...

7. And then



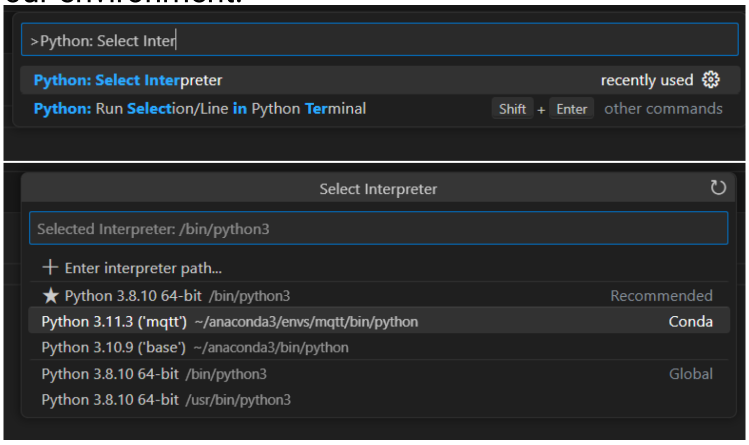
Select a Python Environment

+ Create Python Environment
★ Python 3.8.10 /bin/python3 Recommended
base (Python 3.10.9) ~/anaconda3/bin/python Conda Env
mqtt (Python 3.11.3) ~/anaconda3/envs/mqtt/bin/python
Python 3.8.10 /usr/bin/python3 Global Env

8. Now you can start using the jupyter notebook.

If you are going to work on a python script:

1. Create and open a file "hello.py".
2. You need to choose the correct Python interpreter. Press Ctrl+Shift+P and write ">Python: Select Interpreter". Select the one corresponding to our environment.



>Python: Select Inter

Python: Select Interpreter recently used
Python: Run Selection/Line in Python Terminal Shift + Enter other commands

Select Interpreter

Selected Interpreter: /bin/python3

+ Enter interpreter path...
★ Python 3.8.10 64-bit /bin/python3 Recommended
Python 3.11.3 ('mqtt') ~/anaconda3/envs/mqtt/bin/python Conda
Python 3.10.9 ('base') ~/anaconda3/bin/python
Python 3.8.10 64-bit /bin/python3 Global
Python 3.8.10 64-bit /usr/bin/python3

3. You are ready to start writing python code in a jupiter notebook!

2.10 Java

Java is a high-level, general-purpose, object-oriented programming language designed to have as few implementation dependencies as possible. Compiled Java code can run on all platforms that support Java. It is used to develop mobile applications, web applications, desktop applications, games, and much more.

First, start by checking if you do not already have Java installed: `java -version`
If Java is not present, just run the following command:
`sudo apt install default-jdk`
Check the Java compiler version: `javac -version`
Check the Java version: `java -version`
Thats all!

NOTE: For Java, we recommend a specific IDE like "IntelliJ IDEA"[7].If you want to try vscode you need to install this addon https://marketplace.visualstudio.com/items?itemName=Oracle.oracle-java.

2.11 C

C is a general-purpose, procedural, high-level programming language that was developed by Dennis Ritchie at Bell Telephone Laboratories in 1972. C is an imperative procedural language that supports structured programming, lexical variable scope, and recursion. It has a static type system and was designed to be compiled to provide low-level access to memory and language constructs that map efficiently to machine instructions, all with minimal runtime support. It is a widely used language with C compilers available for practically all modern computer architectures and operating systems.

To install the C tools on Ubuntu execute the following commands:

```
sudo apt update && sudo apt -y upgrade
sudo apt -y install git
sudo apt -y install build-essential libcurl4-openssl-dev
 ↪  libssl-dev
sudo apt -y install gdb cmake
```

Check the installed tools by executing:

```
whereis gcc make
gcc --version
```

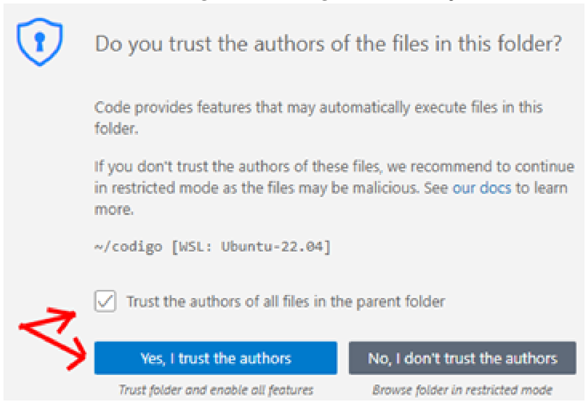
7 https://www.jetbrains.com/idea/

```
make -v
gdb --version
git --version
```

Do not forget to install the "C++ extension" from mocrosoft in VSCode.
Let us try to debug and run a minimal C program. Do not forget the " . " at the end of "`code .`" in the following commands.

```
mkdir codigo
cd codigo
code .
```

If the following warning shows, you must trust the authors (you) to proceed.



Do you trust the authors of the files in this folder?

Code provides features that may automatically execute files in this folder.

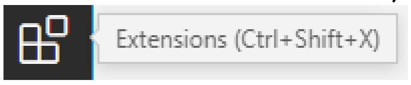
If you don't trust the authors of these files, we recommend to continue in restricted mode as the files may be malicious. See our docs to learn more.

~/codigo [WSL: Ubuntu-22.04]

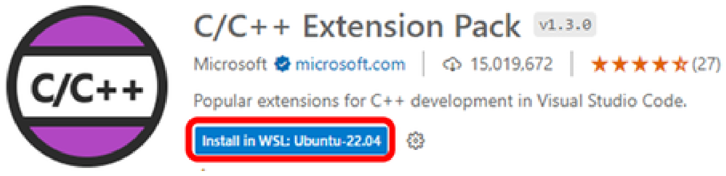
☑ Trust the authors of all files in the parent folder

Yes, I trust the authors
Trust folder and enable all features
No, I don't trust the authors
Browse folder in restricted mode

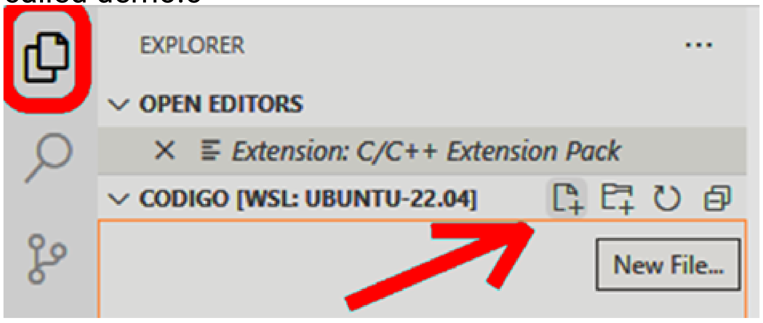
Install the extension "C/C++ Extension Pack":
On the Visual Studio code, click the shortcut for extensions on the left,



and search for **C/C++ Extension Pack**.


C/C++
C/C++ Extension Pack v1.3.0
Microsoft microsoft.com | 15,019,672 | ★★★★★(27)
Popular extensions for C++ development in Visual Studio Code.
Install in WSL: Ubuntu-22.04

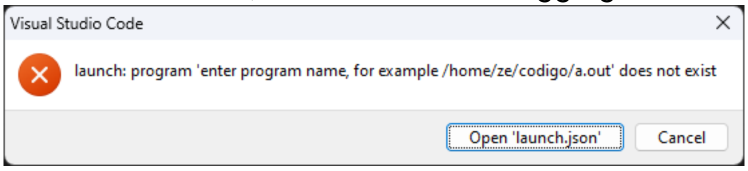
Create a C source code file. Change to the file view and create a new file called demo.c



Type the following code:

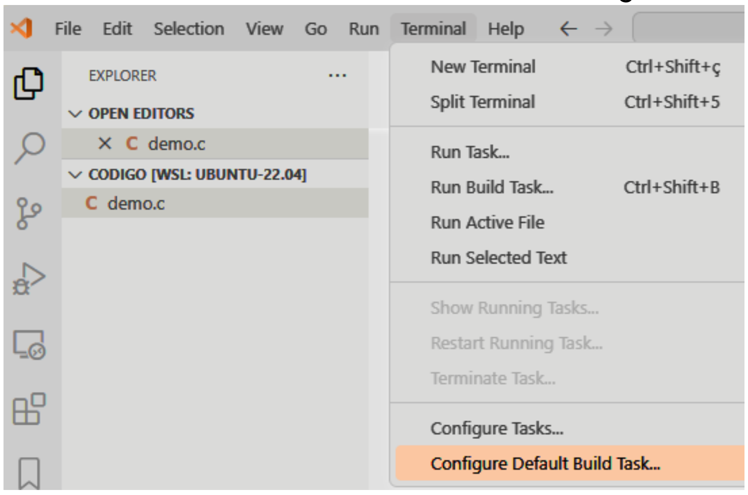
```
#include <stdlib.h>
#include <string.h>
int main()
{
  int a = 3;
  printf("\nOnce upon a time, the number %d was printed!\n", a);
  return 0;
}
```

In the Run menu, select "Start Debugging". The first time you may see:

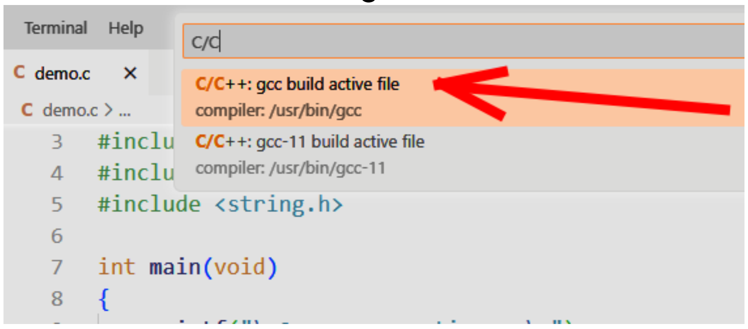

Visual Studio Code

launch: program 'enter program name, for example /home/ze/codigo/a.out' does not exist

Go to the "Terminal" menu and select "Configure default build task".



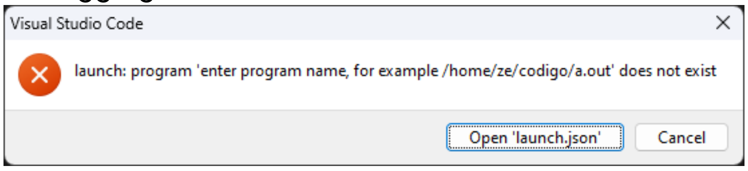
Search and select **C/C++:gcc build active file**



Terminal
Help
c/c
C/C++: gcc build active file
compiler: /usr/bin/gcc
C/C++: gcc-11 build active file
compiler: /usr/bin/gcc-11
demo.c

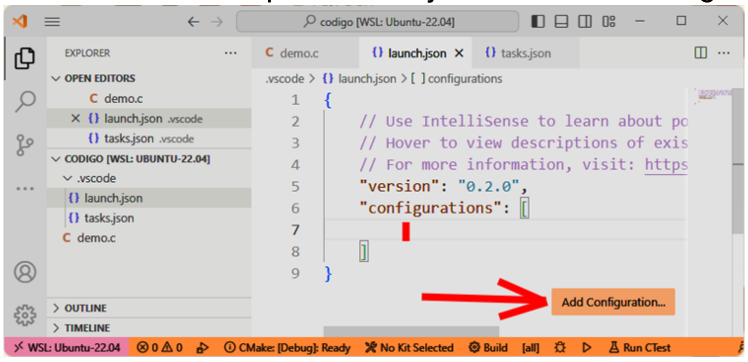
```
#inclu
#inclu
#include <string.h>

int main(void)
{
```

Change the editor to the c program file and try again Menu RUN -> Start Debugging



launch: program 'enter program name, for example /home/ze/codigo/a.out' does not exist

Click the button “Open launch.json” to add a configuration



```
{
    // Use IntelliSense to learn about po
    // Hover to view descriptions of exis
    // For more information, visit: https
    "version": "0.2.0",
    "configurations": [

    ]
}
```

After clicking “Add Configuration” Choose the option **C/C++: (gdb) Launch**

Change the line (listings at the end of this section):

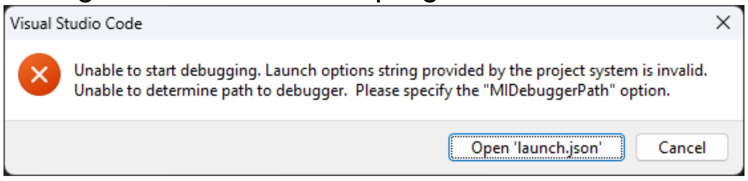
```
"program":  "enter program name, for example $workspace-
Folder/a.out",
```

to

```
"program":  "$workspaceFolder/a.out",
```

If you look at the file tasks.json you will see that this program name is defined on the “-o” argument of the task, under “args” passed to gcc

Change the editor to the c program file and Menu RUN -> Start Debugging



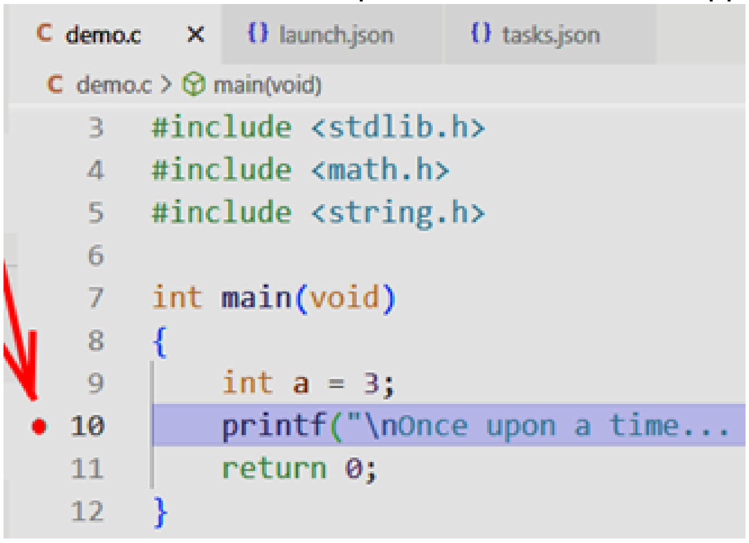
Unable to start debugging. Launch options string provided by the project system is invalid.
Unable to determine path to debugger. Please specify the "MIDebuggerPath" option.

To solve this error, on the command line execute "`whereis gdb`". in my setup it returns "`gdb: /usr/share/gdb`"

In the launch.json file **add**, under ’configurations’, the parameter:

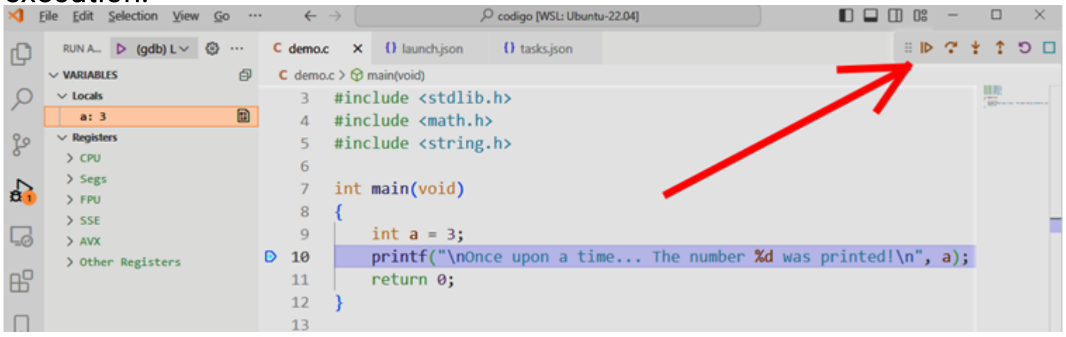
```
"miDebuggerPath": "/usr/share/gdb",
```

Now you should be able to run the program. Let us try to set a breakpoint. Go to the line with the printf statement and click on the empty space on the left of the line number of the printf. A red dot should appear.



```c
#include <stdlib.h>
#include <math.h>
#include <string.h>

int main(void)
{
    int a = 3;
    printf("\nOnce upon a time...
    return 0;
}
```

Once again: Menu RUN -> Start Debugging
The code execution stops at the breakpoint. Here we can inspect variables and other information. On the top right corner we have the buttons to continue execution.



a: 3

```c
#include <stdlib.h>
#include <math.h>
#include <string.h>

int main(void)
{
    int a = 3;
    printf("\nOnce upon a time... The number %d was printed!\n", a);
    return 0;
}
```

Files inside the folder .vscode:
tasks.json

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: gcc build active file",
            "command": "/usr/bin/gcc",
            "args": [
                "-fdiagnostics-color=always",
                "-g",
                "${file}",
                "-o",
                "${fileDirname}/${fileBasenameNoExtension}"
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "detail": "compiler: /usr/bin/gcc"
        }
    ]
}
```

launch.json

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "(gdb) Launch",
            "type": "cppdbg",
            "request": "launch",
            "program": "${workspaceFolder}/${fileBasenameNoExtension}",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${fileDirname}",
            "environment": [],
            "externalConsole": false,
            "MIMode": "gdb",
            "miDebuggerPath": "/usr/bin/gdb",
            "preLaunchTask": "C/C++: gcc build active file",
            "setupCommands": [
                {
                    "description": "Enable pretty-printing for gdb",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                },
                {
                    "description": "Set Disassembly Flavor to Intel",
                    "text": "-gdb-set disassembly-flavor intel",
                    "ignoreFailures": true
                }
            ]
        }

    ]
}
```

2.12 Install .Net SDK 8 in WSL

.NET SDK 8 is a software development kit for building and running .NET applications. It includes what you need to build and run .NET applications using command-line tools and any editor (like Visual Studio Code). It is available for Linux, macOS, and Windows.

1. `sudo apt update && sudo apt -y upgrade`
2. `sudo apt install dotnet-sdk-8.0`
3. If you get a message error stating that the package is not found (means that this package is not yet on Ubuntu official packages lists) try an alternative:
4. `wget https://dot.net/v1/dotnet-install.sh -O dotnet-install.sh`
5. `chmod +x ./dotnet-install.sh`

6. `./dotnet-install.sh --version latest`
7. `nano ~/.bashrc`
8. At the bottom of the file add the 2 following lines

   ```
   export DOTNET_ROOT=$HOME/.dotnet
   ```

   ```
   export PATH=$PATH:$DOTNET_ROOT:$DOTNET_ROOT/tools
   ```

9. Exit the terminal and launch again
10. Check the installation:

    ```
    dotnet --info
    ```

    If it reports the installed SDK the installation is ready. If only the runtimes are reported, try:

    10.1 `sudo apt remove 'dotnet*' 'aspnet*' 'netstandard*'`

    10.2 `sudo rm -f /etc/apt/sources.list.d/microsoft-prod.list`

    10.3 Install again

    10.4 Check again: `dotnet --info`

11. Let us try a simple c# app

    ```
    dotnet new console -o hello
    cd hello
    dotnet restore
    code .
    ```

    Wait for the installation of vscode and then open the *program.cs* file. **Install the suggested c# extension**!

    NOTE: If you get an error stating that you do not have the SDK installed, try setting the environment variables as described on this page on this page[8].

    ```
    Open the file Program.cs
    Press F5 or the menu -> Run -> Start Debugging
    ```

    If asked for a debugger select ".Net 5+ and .Net Core" You may need to install the C# extension. If the message "Required assets to build and debug are missing from "hello". Add them?" Choose yes and try again. Among the output noise, you should see the words:

8 `https://learn.microsoft.com/en-us/dotnet/core/install/linux-scripted-manual#set-environment-variables-system-wide`

```
Hello, World!
```

12. You can now exit the Ubuntu terminal.

Try a minimal API:
https://learn.microsoft.com/en-us/aspnet/core/tutorials/min-web-api

2.13 REST Client

`https://tech.durgadas.in/test-your-api-using-rest-client-vs-code-extensi`
`https://youtu.be/RcxvrhQKv8I`
`https://youtu.be/IZKRAK9Dxm8`
https://marketplace.visualstudio.com/items?itemName=humao.rest-client
How to use: `https://marketplace.visualstudio.com/items?itemName=humao.rest-client`

3 Using GitHub

You can use GitHub in many ways. In this section, we elaborate on the practices we follow @DAE.

Make sure that you completed Section 2.6 before continuing. Follow these tutorials and videos to get started.

1. `https://neuronize.dev/git-github-in-depth-guide`
2. `https://dev.to/documatic/the-art-of-code-review-1lo4`
3. `https://www.youtube.com/watch?v=1SnbOtw4izI`
4. `https://youtu.be/CXgNd3hketM?t=47`
5. `https://www.youtube.com/watch?v=CLfT1fH-38A`
6. `https://ohshitgit.com/`
7. How to Write a Perfect Git Commit Message

4 Introdutory Tutorials

These tutorials are intended to present some technologies and tools. If you are familiar with these topics or they are not relevant for your projects, skip them!

4.1 Web Assembly

Take a look at:

1. WebAssembly (abbreviated Wasm).
2. Overview of Progressive Web Apps (PWAs). More about PWAs.
3. Follow the tutorial Build a Web App with Blazor.
4. Real-Time Notifications Using Blazor and SignalR from scratch
5. Visual Studio Code, .NET 8, Blazor and C# Dev Kit
6. Blazor in .Net 8

4.2 Docker

1. If you are new to Docker and want to know more, you can start with *https://docker-curriculum.com/*. Then, you can explore other Docker venues (following steps)!
2. How to Build a Docker Image.
3. Docker images for ASP.NET Core.
4. Containerise a .NET app.
5. Containerise a .NET app with *docker init*
6. Containerise Web API with Docker & use PostgreSQL
7. Automate Docker Image Building and Publishing with Pack CLI and GitHub Actions

.

4.3 Run and Debug a Web API in a Docker Container

Follow the tutorial from ASP.NET Core in a container with a small change.

```
# Instruction in the manual:
dotnet new webapi --no-https
# Execute instead:
dotnet new webapi --no-https -o myAPI
# Our project will be located in the folder "myAPI" (-o flag)
cd myAPI
code .
```

If the message: *Required assets to build and debug are missing from 'myAPI'. Add them?* pops up, just click YES. When asked to choose the operating system, choose Linux. The keyboard shortcut to open a terminal within VSCode is Ctrl + ' on Linux and Mac. On Windows (our case), it is Ctrl+ç.

Notes:

- To build the Docker image, use the command:
  `docker build -t myapi:1.0 .`
- In the debug part, when instructed to "find Docker .NET Core Launch" look instead for "Docker .NET Launch". The first run may take some time, as the debugger must be downloaded and installed.

Related Tutorials:

1. Learn about debugging .NET in a container
2. Learn about using Docker Compose
3. Customize your Docker build and run tasks
4. Push your image to a container registry

4.4 Publish a Web API in a container

TODO: Explain the usage of:

- dotnet new webapi –no-https -o myAPI
- docker init
- `https://doc.traefik.io/traefik/`

4.5 Kubernetes

1. Kubernetes Crash Course for Absolute Beginners

4.6 Other .Net Tutorials

In this section we present a long list of useful tutorials. Consider this as reference material to which you can come later if needed. It is okay if you do not complete all of them!

1. Getting Started with Entity Framework Core.
2. Mohamad Lawand Tutorials:

    2.1 Containerise Web API with Docker using PostgreSQL.
    2.2 Automatic EF Migration.
    2.3 API Health Checks.
    2.4 API Resiliency.
    2.5 API Caching.
    2.6 Web API Rate Limiting.

3. Profiling
    3.1 MiniProfiler for .NET
    3.2 MiniProfiler in .NET Core API

4. Clean Code Principles [9]
    4.1 The Clean Code Blog
    4.2 Clean Code concepts adapted for .NET/.NET Core
    4.3 Clean Code in ASP.NET Core with Dependency Injection
    4.4 Clean Architecture with ASP.NET Core 7
    4.5 N-layer vs Clean Architecture. Who wins?
    4.6 Rich domain model. What does that even mean? Why is it useful?
    4.7 REST API following CLEAN ARCHITECTURE & DDD Tutorial
    4.8 Milan Jovanović
    4.9 Nick Chapsas
    4.10 Amichai Mantinband

4.7 Resources

"Awesome .NET!", a collection of libraries, tools, frameworks, and software for dotNet.

4.8 Pyspark/Python Development Containers

If you are not familiar with PySpark, skip this tutorial.
Pyspark/Python Development With Docker Containers.

9 This is not a mandatory guideline. There is no software architecture that is the best fit for every case. Nevertheless, if you are not familiar with *Clean Code Architecture*, these tutorials may give you some useful insights!

4.9 Testing

Testing helps to ensure code quality. It is useful for tracking and sorting bugs in the early stages of development.

1. Unit Testing with C# and .NET[10].
2. Test a .NET class library using Visual Studio Code[11].
3. Unit Testing & Code Coverage in ASP.NET Core[12].
4. Python testing in Visual Studio Code[13].
5. Introduction to Test Driven Development.

5 WSL Extras

WSL enables access to USB devices and is not limited to removable drives.

5.1 USB Drives

Assuming that our removable USB drive appears in Windows as drive D, we can mount that drive on Ubuntu using the following commands:

```
sudo mkdir /mnt/d
sudo mount -t drvfs D: /mnt/d
```

List the contents of our usb drive:

```
ls -la /mnt/d
```

Dismount the drive:

```
sudo umount /mnt/d
```

5.2 USB Devices

Full details in https://learn.microsoft.com/en-us/windows/wsl/connect-usb.

```
sudo apt install linux-tools-virtual hwdata

sudo update-alternatives --install /usr/local/bin/usbip usbip
↪  `ls /usr/lib/linux-tools/*/usbip | tail -n1` 20
```

10 https://www.bytehide.com/blog/unit-testing-csharp
11 https://learn.microsoft.com/en-us/dotnet/core/tutorials/testing-library-with-visual-studio-code?pivots=dotnet-7-0
12 https://tinyurl.com/ynjuymf8
13 https://code.visualstudio.com/docs/python/testing

```
sudo apt upgrade
```

From an administrator command prompt on Windows, run this command. It will list all USB devices connected to Windows:

```
winget install --interactive --exact dorssel.usbipd-win

usbipd wsl list
# Take note of the bus id of your usb device. In this example let
↪   us assume it is 1-5

# Change the bus id in the following command
usbipd wsl attach --busid 1-5 -d Ubuntu-22.04

# When you don't need the drive anymore:
usbipd wsl detach --busid <busid>
```

Back on Ubuntu:

```
sudo apt-get install bc

# to list the available usb devices:
lsusb
```

Your USB device should be visible inside Linux.