# Guide to use the BOINC Client in Microsoft's WSL

**(Version 1.2)**

**By boboviz (boboviz chiocciola boincitaly punto org)**

**Thanks to Sabayonino and Astroale !!**

**English Translation by BOINC Synergy**

**What is WSL**

**WSL installation (Windows 10)**

**WSL installation (Windows 11)**

**Linux Distro installation**

**Windows Terminal Subsystem (Optional)**

**BOINC Client installation in WSL**

**WSL Process priority**

**WSL2, GPU and …. Docker ??**

**Alternative client management methods**

**Linux BOINC-TUI Extended**

**BOINC Manager in WSL from Windows (as a service)**

## What is WSL

WSL (Windows Subsystem for Linux) is the virtualization platform for Linux that is inserted directly into Windows 10 and 11 (both Professional and Home). Its deep integration with the system allows for performance almost comparable to native Linux machines. Microsoft offers free of charge, in addition to the virtualization layer, a series of ready-to-use Linux virtual machines. Linux distributions also have the graphical interface starting with the 20H2 version of Windows 10.

Using Hyper-V as the basis of WSL, it is compatible with VirtualBox version 6.1.30 (and later) while it is still incompatible with Vmware Player / Workstation.

*The processor of your PC, of course, must support virtualization (as for VirtualBox).*

# WSL installation (Windows 10)

To use a Linux distro within Windows:

1) Determine which version of Windows 10 you have. It must be updated to at least **version 1903** (and later) or **Build 18362** (and later). To do this, just open the command prompt (type `cmd` in search) and launch the command: `winver`

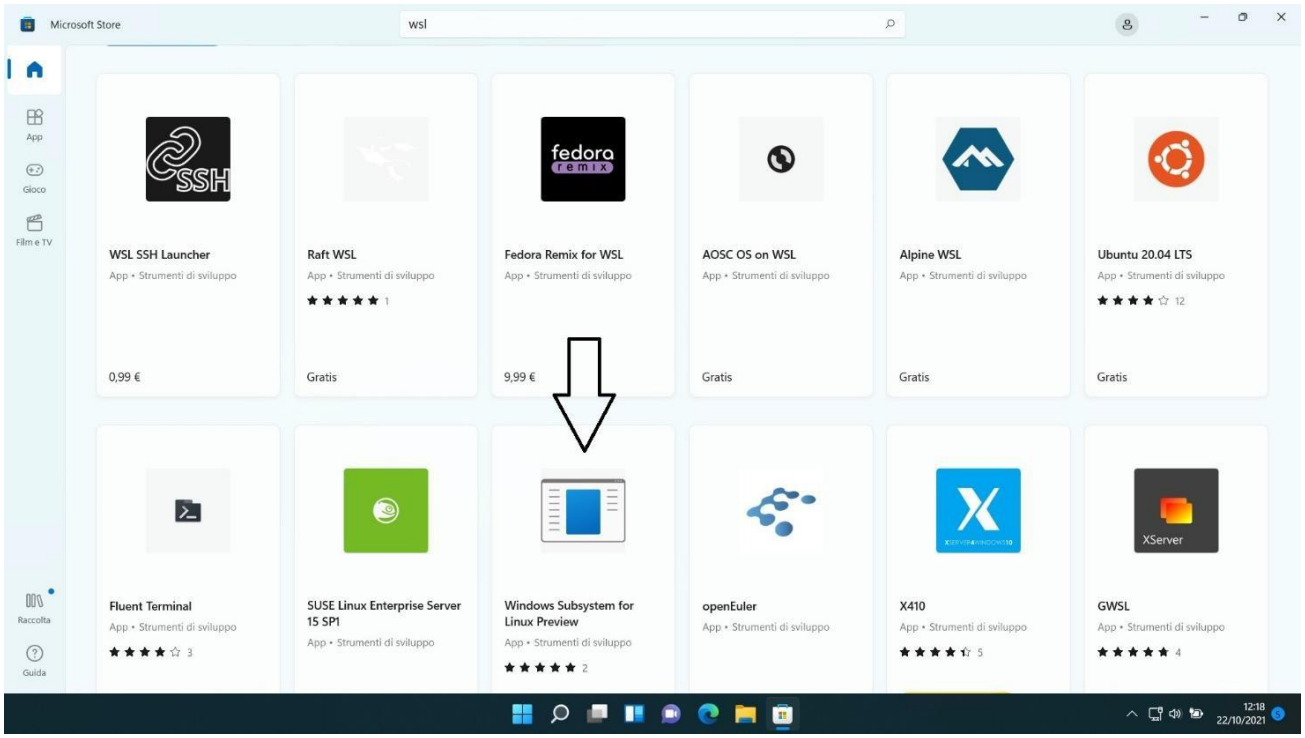

2) Enable, if not already active, the virtualization of the CPU in the bios of the PC. Each motherboard has its own bios, check your own to see how to do it (usually there is the "Virtualization Technology" item to be passed to ENABLE).
3) At this point, start PowerShell as administrator (type powershell in the search and, right clicking on the icon found, select "run as administrator") and type the command:
   `dism.exe /online /enable-feature /featurename: Microsoft- Windows-Subsystem-Linux /all /norestart`
4) Once the installation is complete, to activate the function run the command:
   `dism.exe /online /enable-feature /featurename: VirtualMachinePlatform /all /norestart`
   (for versions 2004 and higher)
   or
   `Enable-WindowsOptionalFeature -Online -FeatureName VirtualMachinePlatform –NoRestart`
   (for versions 1903 and 1909)
5) At this point it is not mandatory, but a restart of Windows is strongly recommended.
6) Download and install the latest updated version of the Linux kernel package here, approving the installation and then, if prompted, reboot.
7) At this point the WSL service is installed and ready to work, but version 2 of the service (much more complete and performing) must be set as default with the poweshell command: `wsl --set-default-version 2`
8) To confirm which WSL version you are using, use the command: `wsl -l -v`

# WSL installation (Windows 11)

In Windows 11 the installation procedure of WSL2 (WSL1 no longer exists) has been greatly simplified: now WSL2 is an app from the Microsoft Store. The functionalities remain the same but in this way, for MS developers, it is much easier to update / maintain WSL in a client environment.

To install, open the Microsoft Store and, in the search at the top right, type `wsl`

As you can see from the image below, in the Store there are no longer only the Linux Distros and the Windows Terminal, but many tools are available for developing and managing projects on WSL.




# Linux Distro installation

1) As in the image above, after installing WSL, select the preferred Distro, click on "Get" and install it.
2) Once the Linux machine is started (double click on the icon that will be created on the desktop) you will be prompted to create a user with a password. This user (and the relative password) will not affect the Windows accounts (since the machine is "isolated") and will have administrative privileges, such as the command `sudo`.
3) At this point the machine is, in all respects, a Linux system and, to begin with, it is advisable to update it with the commands:
   `sudo apt update` and `sudo apt upgrade`

## "Windows Terminal Subsystem" installation (Optional)

The Windows Terminal is an optional component of Windows WSL2 that allows you to deeply customize the WSL system: switch between multiple Linux command lines and the Windows or PowerShell command prompt, create customized key combinations (e.g. copy + paste ), use the search features, create custom themes (font styles and sizes, background image, transparencies, etc.).

Installation always takes place from the Windows 10 Store under "Windows Terminal" (in Windows 11 it is integrated by default).

Here are some interesting animations that explain the functionality of the Terminal.

## BOINC Client installation in WSL and link to a project

1) Since the BOINC client installation will be text only, the command will be:
   `sudo apt install boinc-client`
2) Start the BOINC Client with the command:
   `sudo /etc/init.d/boinc-client start` (this command is valid if used as init sysv / openrc. If, on the other hand, your system is systemd, the command is `systemctl start boinc`).
3) Link the project concerned using the classic boinccmd command and your access key (found in your project user profile), example:

```
boinccmd - host localhost --project_attach
https://boinc.bakerlab.org/rosetta df8d70z2d643b75905d052a25deg6241
```

4) At this point the Client will begin to download the job and process it.

Note: WSL2, by default, uses all available cores and memory of your PC. If you want to limit its use, from Powershell (with the virtual machine off) run the following command:
`notepad "$ env: USERPROFILE / .wslconfig"`
This command will create the virtual machine configuration wslconfig file and here it will be possible to edit the configurations, for example in this way:

```
[wsl2]
memory = 3GB # Limit machine memory to 3GB  processors = 4 #
Limit the cores available to the VM to 4
```

If the above command doesn't work, just create, with any text editor,a file with the name .wslconfig (the period must be included in the name) and save it in the user profile folder (usually it is C: \ user \ username \).
Here other parameters that can be inserted in the file.

# Alternative ways of connecting to the BOINC Client

## 1) Linux BOINC-TUI Extended

**BOINC-TUI Extended** is a small BOINC project manager for Linux terminals based on ncurses libraries. Its main purpose is to monitor and connect BOINC projects, the main features listed below:

- Connection to one or more projects on local or remote host
- Pause, resume and cancel tasks
- Viewing the tasks and related logs

A list of manager features is available on the GitHub of the project.

### BOINC-TUI Extended installation

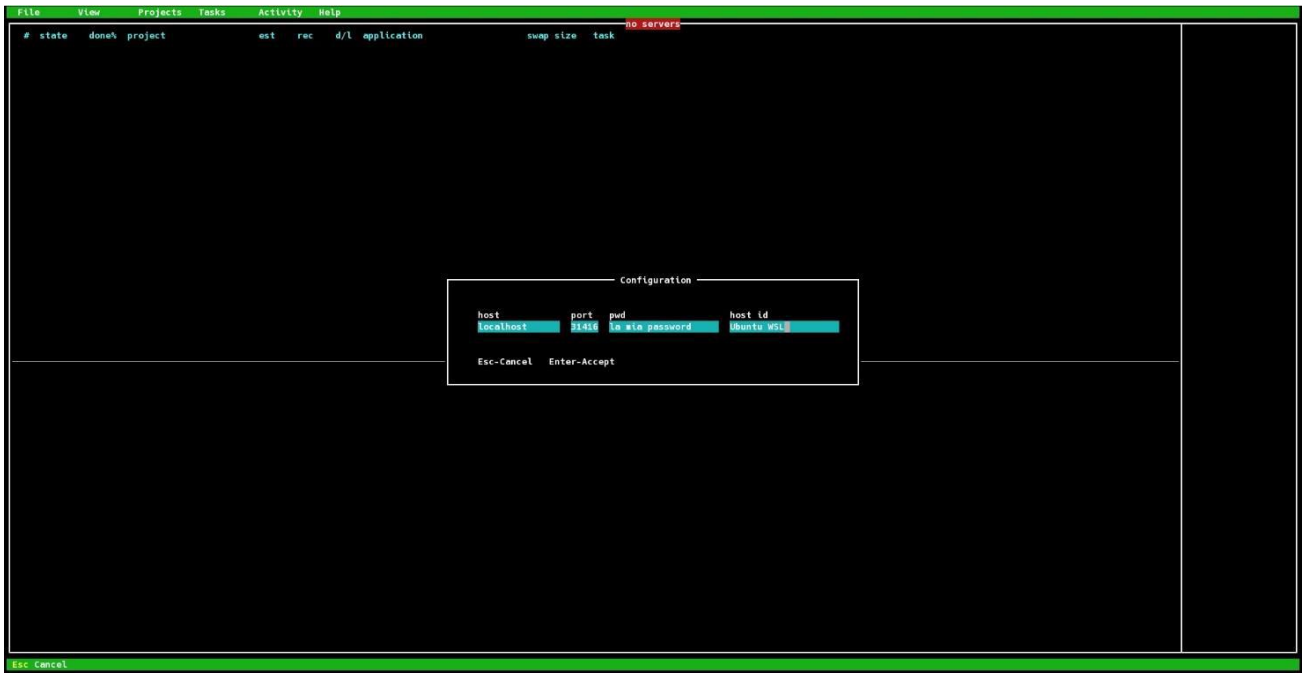
To install dependencies and commands to run, do reference the guidelines of the distribution in use. In the example we will use the commands related to Debian and its derivatives.

1) Install git if not present on your system.
   `sudo apt-get install git`
2) Clone the repository
   `git clone https://github.com/mpentler/boinctui-extended.git`
3) Install dependencies
   `sudo apt-get install libssl-dev libexpat1-dev libncursesw5-dev libgnutls-openssl27 autocon make boinc-client`
4) Enter the folder you just cloned
   `cd boinctui-extended`
5) Build as an unprivileged user
   ```
   Autoconf
   ./configure
   make
   ```
6) Install the executable on the system
   `sudo make install`

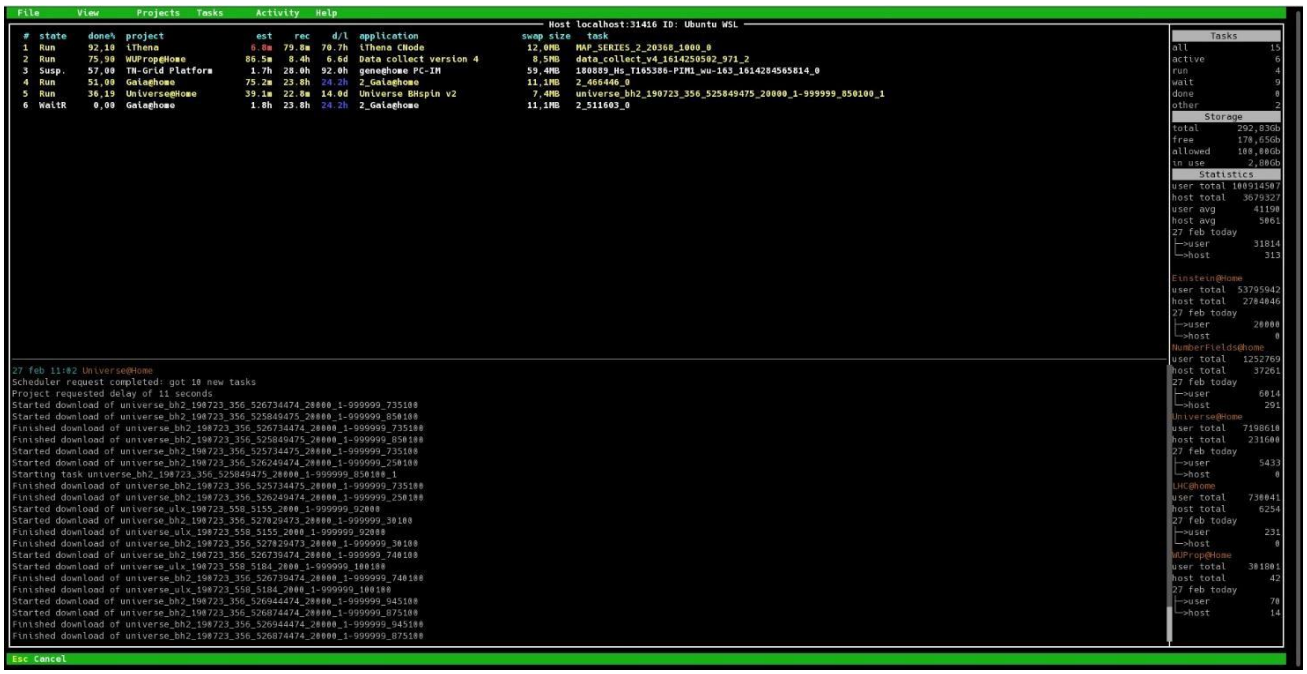
***Possible errors during compilation (in the make phase)***

If while executing the make command you get an error like:
*/lib64/libtinfow.so.6:error adding symbols: DSO missing from command line,* this may depend on the version of the compiler installed on the system and the distribution in use and the related dynamic libraries installed (DSO = Dynamic Shared Object).
If so, run:
`export LDFLAGS = "- Wl, - copy-dt-needed-entries" && make`
Install the executable on the system `sudo make install`

7) Start the application with the command
   `boinctui-extended`



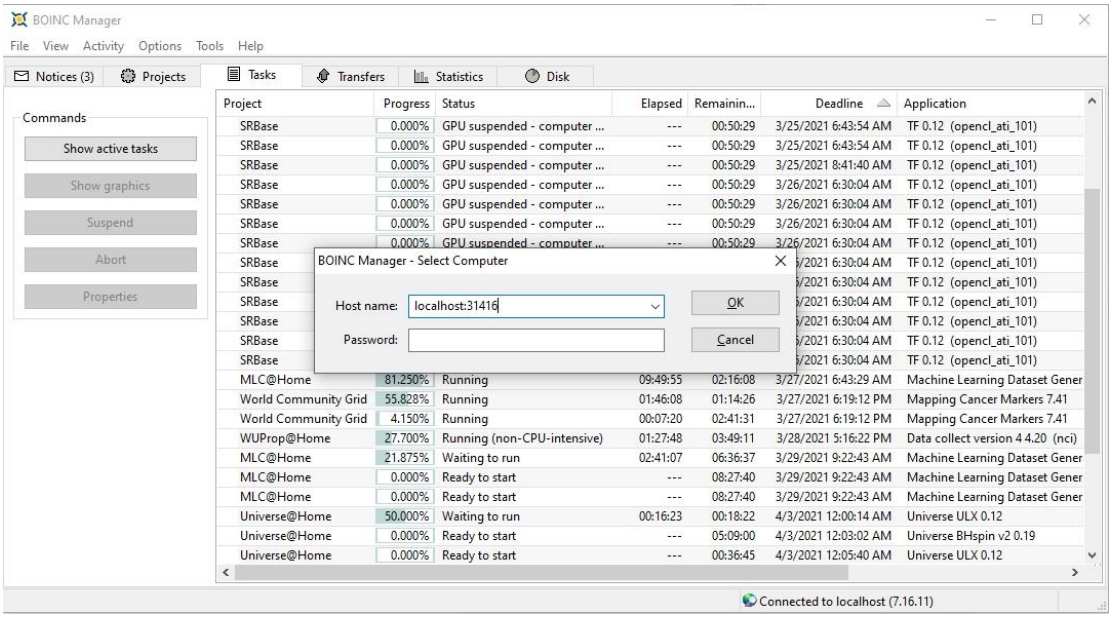

Enter the local (or remote) machine IP: localhost or 127.0.0.1 Port: 31416 (default) Possible password and an identifier for this Host




## 2) BOINC Manager in WSL from Windows (as a service)

If you prefer, instead of managing the machine from the command line, to use the BOINC Manager in Windows (if installed as a service) it is necessary to change the port on which it listens (both services - on Windows and on WSL Linux - use the same port, 31416).

1) Open the Windows command prompt, go to the folder C: \ Program files \ BOINC and here start launch the following command (for old versions of BOINC): `boinccmd --gui_rpc_port 2536` (this is one random leads of example, put whatever you want - recommended above 50,000). This will set the Windows BOINC port to 2536, while the WSL port will remain 31416.
2) For the latest versions of BOINC: `boinc.exe -daemon -gui_rpc_port 2536`
3) At this point it is possible to switch from one instance to another simply by opening the BOINC Manager and selecting File \ Select Computer.
4) A window will open asking you to select the instance and just enter the item `localhost: 31416` (the password will be entered automatically).




5) To return to the Windows instance, repeat the operation and insert

   `localhost: 2536`

**NOTE: This is valid only for the current session; the configuration will be lost when Windows restarts.** *To make it definitive, you need to modify the system registry, as follows: open the registry with the Regedit command and go to the key* `Computer \ KHEY_LOCAL_MACHINE \ SYSTEM \ CurrentControlSet \ Services \ Boinc` *here change (or create, if not there), the ImagePath key with the value C: /boincpath/boinc.exe -daemon - gui_rpc_port 2536 (example random port), where boincpath is the BOINC installation path.*

## WSL process priority

The WSL processes (and therefore the VMs) run under Windows at "normal" priority and this, at times, can create system performance problems. The solution is only possible via the command line:

```
wmic process where name = "Process Name" CALL setpriority "Number" or "String"
```

Where the "process name" is the name of the process / service you want to manage, while "Number" or "String" are the priority levels you want to set, according to this table:

- idle: 64 (or "idle")
- below normal: 16384 (or "below normal")
- normal: 32 (or "normal")
- above normal: 32768 (or "above normal")
- high priority: 128 (or "high priority")
- real time: 256 (or "real time") Ex:

```
wmic process where name = ”calc.exe” CALL setpriority 16384
```

## WSL2, GPU…. and Docker ??

In WSL2 it is possible to directly access not only the CPU, but also the GPUs present in the system. Microsoft's official support for this feature is included in Windows 11 and is available for Windows 10 from version 21H2 - November 2021 (with this version, in Windows 10, support for “Azure IoT Edge for Linux on Windows” is introduced). Nvidia, AMD and Intel support, with appropriate drivers, this paravirtualization feature and there is also the possibility of using a specific version of Docker desktop for GPUs. Furthermore, a Docker version for BOINC already exists, so the two functions can be integrated.

**Nvidia driver**: https://developer.nvidia.com/cuda/wsl/download

**AMD driver**: https://www.amd.com/en/support/kb/release-notes/rn-rad-win-wsl-support

**Intel driver**:
https://www.intel.com/content/www/us/en/artificial-intelligence / harness-the-power-of-intel-igpu-on-your-machine.html # articleparagraph_983312434

**Docker on WSL:** https://www.docker.com/blog/wsl-2-gpu-support-is-here/

**Docker BOINC:** https://hub.docker.com/r/boinc/client

**Cuda WSL (with test applications included):**
https://docs.nvidia.com/cuda/wsl-user-guide / index.html

**DirectML and Tensorflow on WSL:**
https://docs.microsoft.com/en-us / windows / ai / directml / gpu-tensorflow-wsl