

Git + VSCode setup guide (kinda)

Fabien HOAREAU

Here is a short guide, destined to the not-so-tech-savvy people (such as myself) for installing Git and Visual Studio Code on Windows 10, for syncing LaTeX modifications on a PLM Lab TeX project. As usual when dealing with PLM Lab, Git, LaTeX, or any tech related thing, we profusely thank François Le Maître for his help.

First Download VS Code from <https://code.visualstudio.com/>
Install, and add the LaTeX Workshop extension.

Next install Git from <https://gitforwindows.org/>
In the installation process, select "Use Visual Studio Code as Git's default editor" (see figure 1), and if you want you can also select "None" as a credential helper (see figure 2), as it's probably not useful. All other selections can be kept as they are by default.

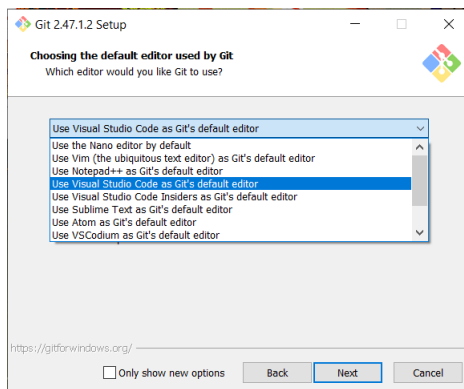


Figure 1

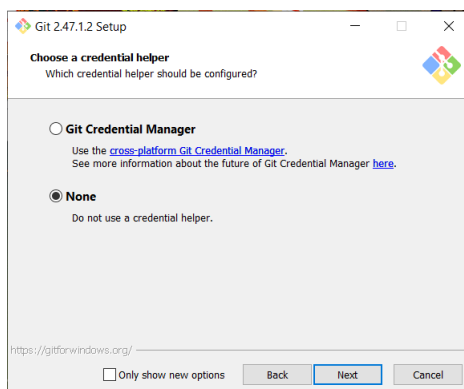


Figure 2

Next is taking care of the SSH key, which is the meaty part. We'll basically be following the instructions from <https://plmlab.math.cnrs.fr/help/user/ssh.md> but even that can seem a little daunting. We'll be using the ED25519 key type. Open the newly installed Git Bash and type the following (the command is on the previously linked page, see figure 3):

```
ssh-keygen -t ed25519 -C "<comment>"
```

replacing <comment> by the name you want your key to have on PLM lab.

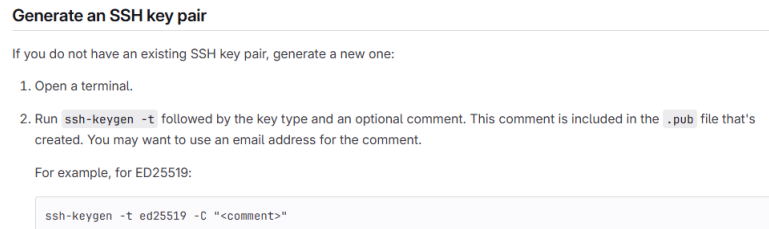


Figure 3

Make sure that you are located in your local user directory (using `"cd .."` if needed to go back one directory above, `"cd nameofdirectory"` to move forward to the designated directory, and `"ls"` to check where you are at any given moment). SSH key files, both the public and the private one should be stocked in a "hidden" (dot)ssh folder. It may happen that the previous command line does not create that folder. Creating it manually and retyping the previous command line should work.

Next is adding the newly created SSH key to the PLM account. You can do this automatically by typing the following command line (again, see figure 4):

```
cat ~/.ssh/id_ed25519.pub | clip
```

which should copy the content of the wanted key in the clipboard.

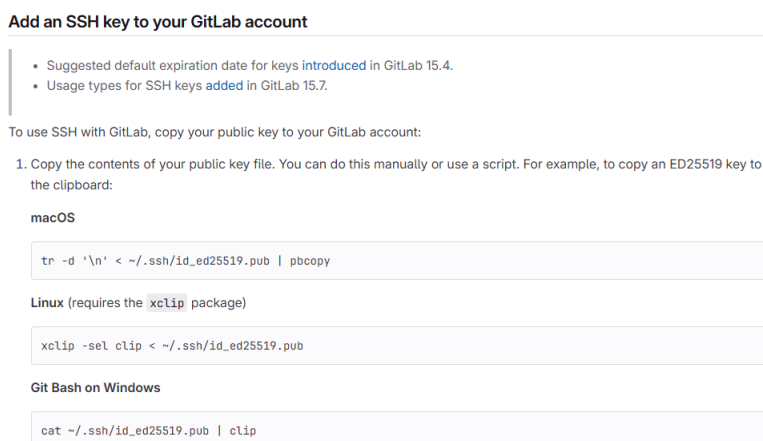


Figure 4

You can then paste it in the SSH keys manager in PLM Lab (under user -> Preferences -> SSH Keys) (see figure 5). You can at this point manually change the name of your key, which is useful if you have many of those, or work with several computers. You should also remove the expiration date of your key, which has a date by default, in order for it to become permanent. You can then click add key, and it should be operational.

SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab. SSH fingerprints verify that the client is connecting to the correct host. Check the [current instance configuration](#).

Your SSH keys 2

Add an SSH key

Add an SSH key for secure access to GitLab. [Learn more](#).

Key

Begin with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'.

Title

Example: MacBook key

Key titles are publicly visible.

Usage type

Authentication & Signing

Expiration date

YYYY-MM-DD

Optional but recommended. If set, key becomes invalid on the specified date.

Figure 5

If it does not work, or if you want to, you can also do this step manually. In order to do that, open the (dot)ssh folder, and open the

`id_ed25519.pub`

file with Notepad, or anything similar. Copy all the content of the file, and paste it in the same way.

The next step is making your LaTeX PLM project "usable" by your VScode setup. Create a folder named "Git" in your local user directory. We need to clone the project in this folder. Go to your project page, and click the "copy URL" button, in "clone with SSH", under "CODE" (see figure 6). Of course this only works if your SSH keys are functional.

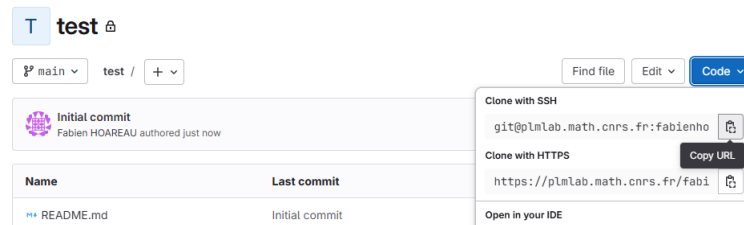


Figure 6

Go back to Git Bash, and type "cd Git" to move to your newly created Git directory. Then type "git clone " (don't forget the space after clone) and paste the copied URL, then press Enter. The folder should be named as your project is. We now need to identify ourselves in order for VScode/Git not to question us each time we want to modify anything. Just type the following in Git Bash:

```
git config --global user.name "YOUR NAME"
```

```
git config --global user.email youremailadress@whatever.com
```

Finally, we need to tell VSCode to work "in the right place", and an easy way to do this is to open it while "in the right place" by using the terminal. Open a terminal: (windows key + R) and type "cmd" then press Enter. Move to the project folder ("cd Git", then "cd projectname") then open VSCode by typing "code ."

For the subsequent utilisations you should be good to go from the start.

When working with VSCode, don't forget to make sure the version of your project you are working is up to date. If you did not set it up to check automatically (there should be a popup asking if you want the "Fetch" functionality to be active, DO press Yes) you can do it manually by clicking the "refresh button" on the bottom left of the VSCode interface (see figure 7).

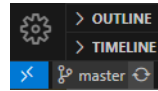


Figure 7

Finally, when making changes to your TeX file, whether you compile or not, save (CTRL + S) then commit the changes (with a nice message that is readable and instructive for all members of course) (see figure 8). Press YES when VSCode asks you to stage all the changes, and then sync the changes for them to take effect for good (see figure 9).

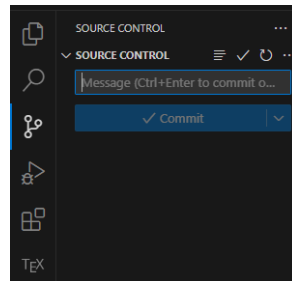


Figure 8

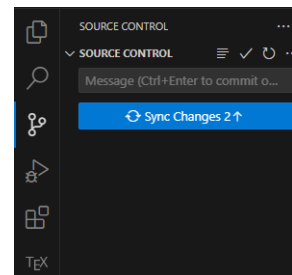


Figure 9

As a small addition, if you (like me) do not like having to scroll horizontally to see code lines in VSCode, the shortcut "Alt + z" makes all code lines fit the current window horizontally, independantly of the size of said window.