Local LaTeX and You!

A guide to setting up LaTeX on your device

Kaushik Rajendran

December 27, 2024

Contents

1 Introduction

LaTeX is a high quality typesetting environment designed for creating scientific and technical documents. It is the standard for any of our publications or official scientific communication. Perhaps the easiest and most common way to use LaTeX is through Overleaf, an online LaTeX environment that includes a bunch of useful features such as the visual editor for first-time LaTeX users, and real-time collaboration.

1.1 So why use a local version if Overleaf is this good?

Obviously, Overleaf is amazing and I do not intend to disparage its usage here. But one of the biggest flaws with using Overleaf exclusively is the fact that it is an *online* editor. This means that you need to be constantly connected to the internet to use Overleaf to edit your documents. This means that in cases like power/server outages, or times when you're travelling without an internet connection (say a long flight on the way to a conference), you can't use Overleaf. In such cases, it is good to have a local option to work in LaTeX offline. Additionally, depending on the code editor you use to write your local LaTeX documents, you may have some nice integrations/shortcuts to exploit.

1.2 The purpose and scope of this writeup

This writeup is mainly intended for LaTeX users who are comfortable with using Overleaf for preparing documents. As such, this document will assume basic understanding of the LaTeX environment. The remainder of this writeup is split into two sections, namely

- **Installation and Setup**: Talking about how to install and use LaTeX locally, and providing some personalization/workflow tips
- **Integrations**: Exploring integrating local LaTeX with Overleaf

1.3 More resources

- The source code for this writeup and the examples shown can be found in the public Github repo accessible at `https://github.com/kaushikr64/Local-LaTeX-guide`
- Quite a bit of information on the VSCode frontend for LaTeX can be found at the site for it `https://github.com/James-Yu/LaTeX-Workshop/wiki`

2 Installation and Setup

2.1 LaTeX installation

The standard distribution of LaTeXis TeX Live, which works on most platforms. Detailed instructions can be found here. Windows users may download TeX Live from `https://www.tug.org/texlive/windows.html#install`, Mac users may do so using the MacTeX installer from `https://www.tug.org/mactex/mactex-download.html`, and Linux users can download TeX Live from `https://www.tug.org/texlive/quickinstall.html`. This is technically the only thing you need to install to start using LaTeX locally, since all of these will also install a front end to edit your LaTeX documents, as well as install most commonly used packages (I have never needed to manually install packages so far).

You can also use your preferred code editing front end to edit your LaTeX code. I prefer using VSCode for this purpose, and recommend it due to its easy to use LaTeX extension and abundance of other useful tools. I will cover this in the following subsection

2.2 Using LaTeX on VSCode

The first step to this is obviously, downloading VSCode itself. You can do this at `https://code.visualstudio.com/download`. Once VSCode is downloaded, the LaTeX Workshop extension must be installed. This can be done by navigating to the Extensions tab on the sidebar and typing it into the search bar. This is shown below

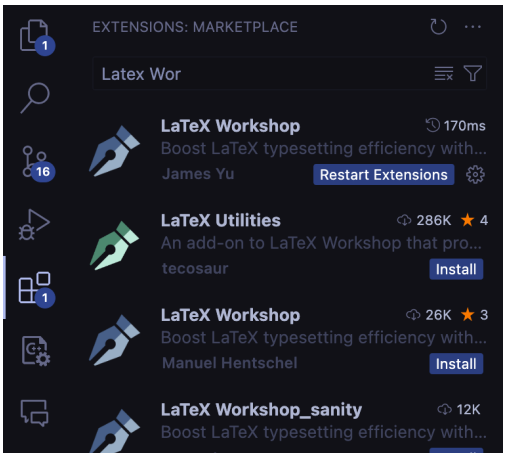

Figure 2.1: The extension you want to install is the topmost one on the search results (the one by James Yu), which I have already installed in this image.

From here, you can simply create TeX files in VSCode by ending a file with the `.tex` extension, and edit and compile them through the build button in the LaTeX Workshop sidebar tab, or the build button on top of the currently open TeX file. You can then use the LaTeX workshop sidebar once more to view the output PDF. Alternatively, you may find the output PDF in your file path and open it to the side in VSCode like you would a secondary tab of code. Nominally, this output PDF is in the same directory as your main TeX file. This is shown in Fig. 2.2

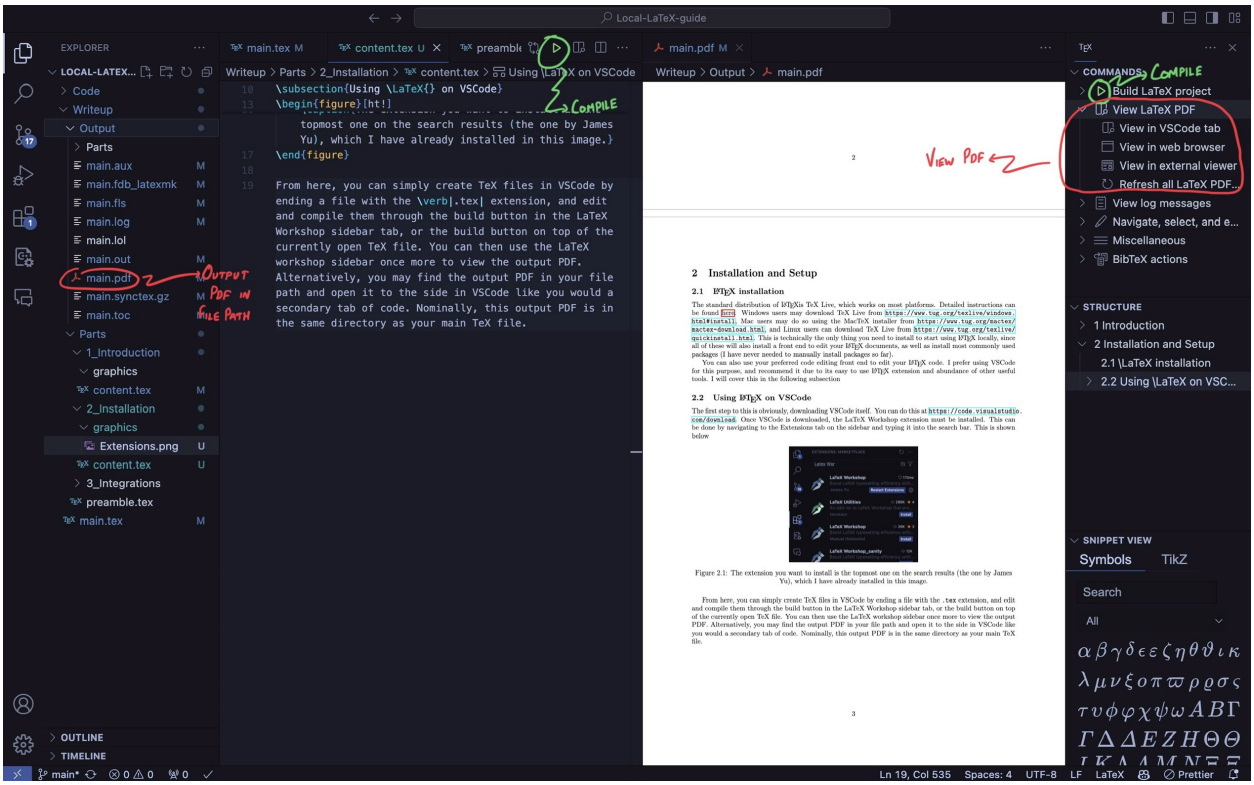

Figure 2.2: Where to find the output pdf and compile button in VSCode. Note that I moved the LaTeX workshop menu to the right sidebar in this picture, and have slightly modified the place where the output files go (more on this in a bit).

2.3 Managing LaTeX files locally

As can be noticed in Fig. 2.2, there are a few files you probably haven't seen in your days using overleaf. These are the auxiliary files needed to compile the output PDF, and are produced on every compilation. While LaTeX workshop has a pretty tempting cleanup feature for these files, keeping them does help with compilation speed. As such, it is better to put them into some output folder to avoid clutter. The path to which the output and auxiliary files are sent can be edited throught the outdir setting for LaTeX Workshop on VSCode, which can be navigated to through the gear icon on the VSCode sidebar. In Fig. 2.3 is how I have set it up, where all output files are sent to an Output folder in the same path as `main.tex`.

Note that apart from this, I have also divided my latex document into individual files, which I am then inputting into the main file using `\input{}`. Especially when dealing with local LaTeX, this is preferred to `\subfile{}` since the latter will make its own output folder and associated files for each subfile, bloating the project.

To make sure the compiler always compiles main, a magic comment on these inputted subfiles is prudent. Such a comment simply tells the compiler to set the root file to be the main file is. This is given by `% !TeX root = <root file directory>`. The comment can also be inserted through the search bar as in Fig. 2.4 (this method also gives a nifty drop down to select the root file). However, to use this comment, you must also disable forcing recipe usage by typing `"latex-workshop.latex.build.forceRecipeUsage": false,` into the settings.json or setting it in the settings tab.

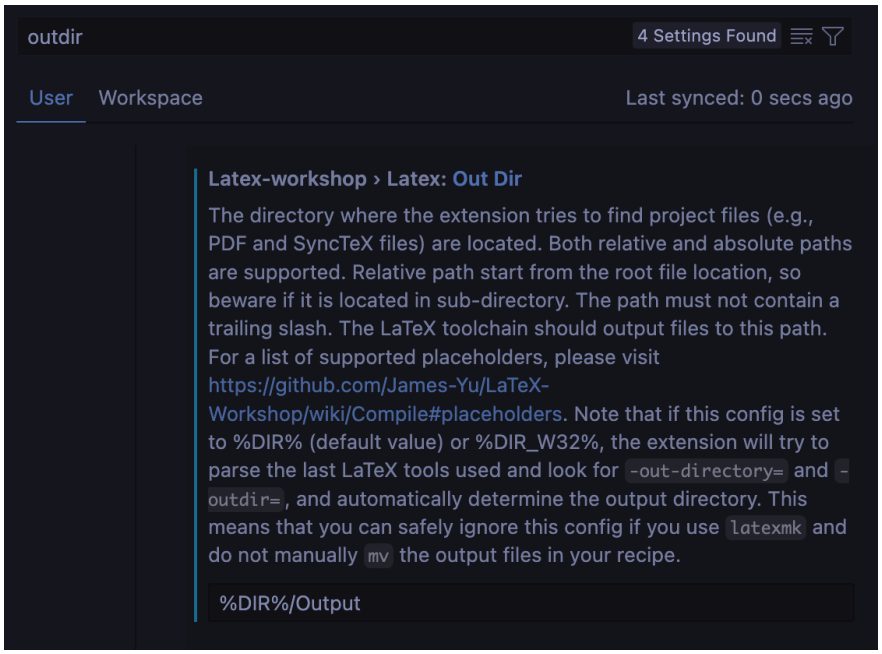

Figure 2.3: Output directory setting

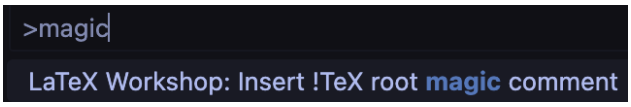

Figure 2.4: Insert a magic comment

2.4 Some useful features and tips for LaTeX in VSCode

- Turn on word wrapping in the VSCode editor. This can be done exclusively for LaTeX through the language specific settings as follows

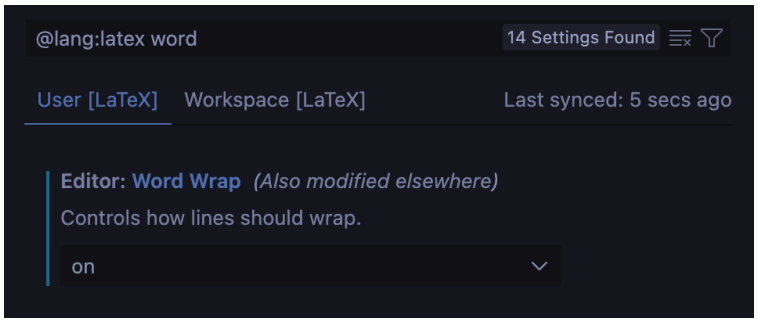

Figure 2.5: Word wrapping settings

- Github copilot is a useful extension for some menial typesetting tasks, and especially in my experience for building tikz diagrams. Get it for free with a student github account and download it from the VSCode extensions menu, just make sure to turn off the somewhat annoying inline suggestions. This, along with the former item can be done using the following lines in settings.json (can be accessed

through the VSCode searchbar)

```
 "[latex]":  {
"editor.inlineSuggest.enabled":  false,
"editor.wordWrap":  "on",
},
```

Additionally, editor autocompletion using copilot AI can be turned off with `"github.copilot.editor.enableAutoCompletions":  false,`. This way, copilot won't interfere with your typing, but can still be easily accessed through a single shortcut to help do boring/repetitive tasks.

- LaTeX workshop supports snippets. There are several useful inbuilt ones such as `BEQ` or `BAL` to start equation or align environments, or `@a` or `@b` for greek letters like α, β. A background on snippets is found on the workshop wiki at the url `https://github.com/James-Yu/LaTeX-Workshop/wiki/Snippets`. A concise list may also be found at `https://cheatography.com/jcwinkler/cheat-sheets/latex-workshop-visual-st` You may also make custom snippets in VSCode for use by navigating to the snippets menu in the searchbar

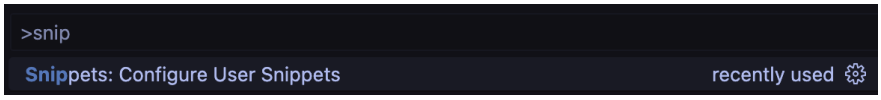

Figure 2.6: Snippets menu in the searchbar

- Intellisense is LaTeX workshop's capability to autofill citations, commands, environments, labels, and file names. Intellisense usually updates on file save, but you can make it update more aggressively by setting `latex-workshop.intellisense.update.aggressive.enabled` to `true` in the settings menu.
- Much like double clicking the PDF in Overleaf takes you to the corresponding line of TeX, Command/-Control clicking text (depending on whether the OS is Mac or Windows) in the VSCode internal PDF viewer takes you to the corresponding line of TeX. While in the TeX file, pressing Cmd+Opt+J on Mac or Ctrl+Alt+J on Windows takes you to the point in the PDF corresponding to the cursor location on the TeX file. Note that this requires the SyncTeX file to be present in the same directory as the output PDF. This file is created on each compile, and is automatically stored in the proper directory to be used.
- The smart clicks VSCode extension is pretty handy to select stuff in brackets. I highly recommend downloading it.

2.5 Keeping your project synced

A cool thing about using Overleaf is that since it is an online tool, it saves your progress on a cloud and can be used on basically any device. You can get cloud saving functionality using local LaTeX by using any cloud based file storage such as Dropbox, OneDrive, or iCloud. However, my personal favorite for this purpose is Github due to its version control capabilities, ability to make submodules, and integration with Overleaf (to be discussed in the next section). Note that Dropbox also has this behavior with Overleaf, and this can be explored further at `https://www.overleaf.com/learn/how-to/Dropbox_Synchronization`. I will be showing how the Github-Overleaf integration works in the next section.

When it comes to saving my work with Github, I essentially create a repository for entire projects, including both the code and the writeup for them. This allows me to do some cool stuff where I can easily cross reference project code in the writeup (through the listings package) and output graphics from my code directly into my writeup. Sometimes, however, it is good to save the writeup as a seperate repository within the overall project repo. A guide to creating submodules in Github is given in `https://gist.github.com/`

`gitaarik/8735255`. Make sure this submodule can act as a standalone LaTeX project, since that is crucial if you want to integrate the submodule with Overleaf (which will be elaborated upon in the next section).

Both my user settings .json file and user generated snippets .json file are included in the Github of this project for further reference. Additionally, the entire LaTeX file tree for this document is also on the Github repo.

3 Integrations

3.1 Overleaf

The best way to use local LaTeX is *in conjunction with*, and not instead of Overleaf. Overleaf does have its own strengths after all, due to its real time collaboration ability. Overleaf and local versions of LaTeX can be integrated into one seamless experience using Github (although, the owner of the document must have Overleaf premium for this feature). This can be done as follows

1. Have your LaTeX project in a Github repository. It is fine if your repo consists of other files such as project code, but keep in mind that Overleaf project sizes are limited to 100MB. If the Github repo for your project is larger than this, then it is probably better to save your writeup as a submodule in the project repository (and use this submodule as the repository for the LaTeX project itself).

2. Connect your Overleaf account to your Github account through your account settings

3. Now, when creating a new project, select the option to import from Github This will give you a drop

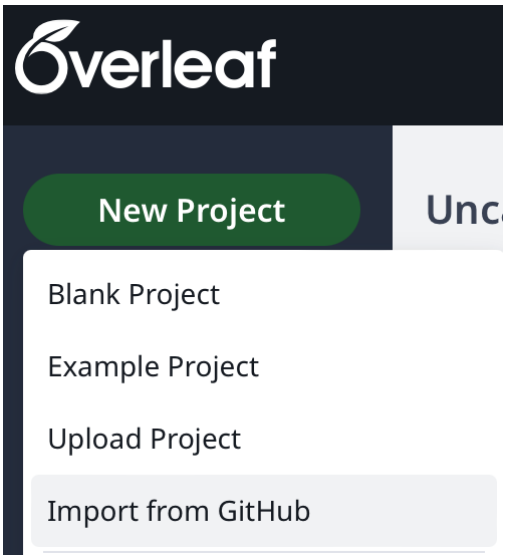

Figure 3.1: Importing a project from Github

down box of your Github repos. Select the one you wish to import into Overleaf. Note that this will import EVERY file in the repository into overleaf, so if you have your LaTeX stuff saved into a submodule, import that. Once you do this, Overleaf should automatically identify your main tex file, although if it does that improperly you can always set it in the project menu.

4. Now that you've connected Overleaf and Github, you can push and pull commits between them using Overleaf as a remote repository through the project menu bar. You will see this screen

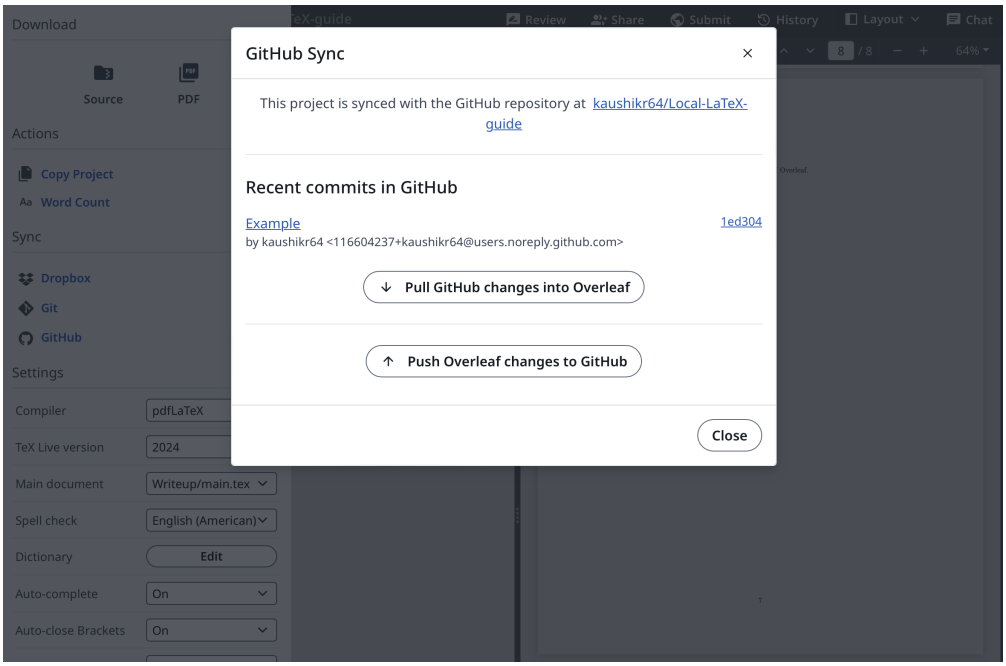

Figure 3.2: Github Sync screen, you can get here by pressing Github under the sync options to the left

You can now pull/push commits as necessary. Any user given edit access to the overleaf document can pull and push changes to the Github repo even if it is private, so this allows for easy collaboration with others

5. Suppose you already have an Overleaf document, and wish to update it locally, this can also be done by creating a Github repo through Overleaf itself for the project. Simply navigate to the project menu once more and click the Github option under the sync options. This brings up the following page

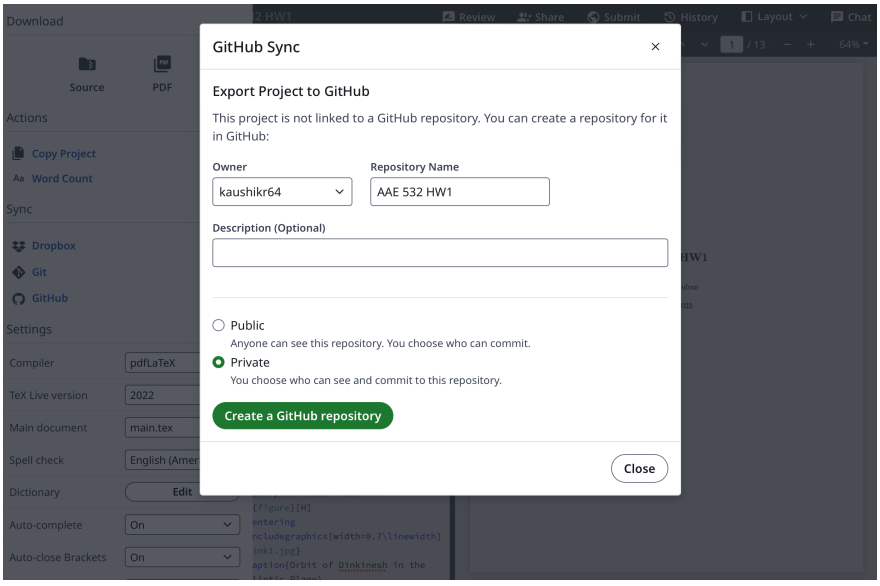

Figure 3.3: Create a Github repo from an existing Overleaf project

After using this menu to create a Github repo, you can clone it to your local machine and start using your local LaTeX setup. From here syncing Github and Overleaf works the same way as before.