ChatGPT

Setting Up UV and Cloning the Repo in WSL (Part 2 Guide)

Prerequisite: By now (after Part 1) you should have WSL installed with the Ubuntu distribution, and you've created a Linux username/password. The next steps (Part 2) will install the **uv** tool and set up the project repository within your WSL Ubuntu environment. We'll go through each step in detail.

Launching Ubuntu via PowerShell (Instead of `wsl`)

To enter your WSL Ubuntu environment, you should start it from PowerShell using the `ubuntu` command (not the generic `wsl` command). This ensures you start in your Linux home directory, not a Windows-mounted folder. For example, launching with `ubuntu` puts you in your Linux home (e.g. `/home/<yourname>`), whereas using `wsl` might drop you into a Windows path like `/mnt/c/Users/<YourName>` [1]. Follow these steps:

1. **Open a new PowerShell window** on your Windows PC (you can press **Start**, type "PowerShell", and open it).
2. In the PowerShell prompt, type `ubuntu` and press Enter. This will launch your default WSL distribution (Ubuntu) and log you into the Linux shell as your WSL user. You should see a prompt like `yourname@machine:~$` indicating you're in Linux.
3. (If for some reason the `ubuntu` command is not recognized, ensure that Ubuntu is installed. You can also try `wsl -d Ubuntu` as an alternative. But normally, `ubuntu` should work after the WSL installation.)

Once this is done, you are now in your WSL **Ubuntu** terminal, located in your Linux home directory (`~`). You can verify this by running `pwd` (print working directory) – it should output something like `/home/your-linux-username`. You can also run `ls` to list files (it might be empty or just contain standard folders like `Desktop`, etc., since this is your Linux home).

> **Note:** It's important to operate in your WSL's Linux filesystem (your home directory) rather than the Windows filesystem. This gives better performance and avoids permission issues when using Linux tools. Microsoft's WSL documentation recommends storing project files on the Linux side if you plan to use Linux tools, rather than on a mounted Windows folder [2] [3].

Installing uv in the WSL Ubuntu Environment

Now that you have an Ubuntu shell open in WSL, you can install the **uv** tool (per the instructions). The uv tool is installed by running a script provided by its maintainers. Here's how to do it:

1. **Ensure you are in your home directory.** If you just launched Ubuntu as above, you should already be in `~` (your Linux home). You can use `pwd` to check. If not, use `cd ~` to go to your home directory.
2. **Run the install script for uv:** Copy and paste the following command into the WSL terminal and press Enter:

   ```
   curl -LsSf https://astral.sh/uv/install.sh | sh
   ```

   This command uses `curl` to download the **uv installer** script from `astral.sh` and pipes it to `sh` (shell) to execute. It will download and set up **uv** automatically. You should see it fetching some files or packages.
3. **Allow the script to complete.** It may ask for your password if it needs to install something with sudo (since you're installing a tool). Provide your Linux password if prompted. Otherwise, wait until it finishes.

When the script finishes, **uv** will be installed (it's usually placed in your Linux home's `.cargo/bin` or similar, and added to your PATH). The installation output might mention that you should start a new shell for changes to take effect. In fact, the official guide notes that after running the install script, you may need to **open a new shell** so that the updated PATH (including uv) is recognized [4]. We'll do that next.

Refreshing the Environment (Exit and Re-open WSL)

After installing uv, you should restart your WSL session to load the new environment variables (so that the `uv` command is available in your PATH). Here's what to do:

1. **Exit the Ubuntu shell:** In the WSL terminal, type the command `exit` and hit Enter. This will terminate your WSL session and return you to the PowerShell prompt on Windows.
2. **Launch Ubuntu again:** Back at the PowerShell prompt, run `ubuntu` once more and press Enter. This opens a fresh WSL Ubuntu session. This step reloads your profile with the updates made by the uv installer (so now the `uv` command should be recognized in this new session) [4].
3. **Verify uv is installed (optional):** You can check that `uv` is on the PATH by running `which uv` (it should show a path, meaning the shell can find it) or `uv --version` to see if it responds with a version number.

You are now ready to use `uv` in your WSL environment. Make sure you're in your Linux home directory (`pwd` should show `/home/yourname`). If not, navigate to it with `cd ~`.

Creating a Projects Directory and Cloning the Repository

Next, you'll set up a directory for your projects and clone the repository as instructed:

1. **Create a "projects" folder:** Still in the WSL Ubuntu terminal, run `mkdir projects`. This makes a new directory called **projects** in your current location (which, if you stayed in `~`, will create `/home/yourname/projects`). This directory will be used to keep your code projects organized.
2. **Enter the projects directory:** Run `cd projects` to change into the newly created projects folder. You can run `pwd` to confirm you are now in `~/projects`.
3. **Clone the Git repository:** Now run the git clone command given in the instructions:

   ```
   git clone https://github.com/ed-donner/agents.git
   ```

   This will download the **agents** repository from GitHub into a folder named "agents" inside your projects directory. You should see git output indicating it's cloning the repo (progress of downloading objects, etc.).
4. *Note:* If you get an error like `git: command not found`, it means Git is not installed in your WSL environment. Ubuntu usually has Git by default, but if not, install it by running: `sudo apt update && sudo apt install git`. Then retry the `git clone` command.
5. **Enter the repository directory:** Once cloning is complete, change into the project's directory with `cd agents`. This "agents" directory is your **Project Root Directory** mentioned in the guide. Running `ls` here should show the files and subfolders of the project.

At this point, you have the uv tool installed and the project repository cloned in WSL. All the work is happening in your Linux filesystem (under your WSL Ubuntu home), which is correct for using Linux tools.

Running `uv sync` in the Project Directory

Finally, with the repository set up, you need to run the `uv sync` command within the project directory. This command (specific to uv) will likely set up the project's environment (install any dependencies, create virtual environments, etc., as defined by the project's configuration). Here's how to proceed:

1. **Ensure you are in the `agents` project directory** (the one you just cloned). Your prompt might already show that (e.g., `...:~/projects/agents$`). If not, `cd ~/projects/agents` to get there.
2. **Run the sync command:** Type `uv sync` and press Enter. This will initiate uv's synchronization process. You might see it downloading or installing various packages, creating environments, or doing other setup tasks as it reads the project configuration. This step could take a little while if it needs to download dependencies.
3. **Wait for it to complete:** Let `uv sync` finish. If everything is set up correctly, it should complete without errors. This means your project environment is now ready.

   Possible Issue – Memory Error: The guide's author mentioned that they encountered a memory error at this stage on their machine. This is likely an uncommon issue, so you may not run into it. If `uv sync` fails with an "out of memory" or similar error, don't panic. It could be related to WSL's resource limits. One quick fix is to ensure WSL has enough memory (you can configure WSL's `.wslconfig` to increase memory, or close other applications to free up RAM). The guide hinted that they have a specific fix if needed, so if you hit this problem, you might need to consult that source or ask for that fix. Otherwise, assume it will run smoothly.

After a successful `uv sync`, your Part 2 setup is complete . You have installed WSL (Ubuntu), added the **uv** tool, cloned the **agents** repository, and synchronized the environment. At this stage, you're ready to move on to Part 3.

(Next Step: Configuring Cursor with WSL – Part 3 Preview)

In Part 3, you will integrate this WSL environment with the Cursor editor. This involves using the Cursor **WSL extension** and opening the project in a remote WSL window:

- **Install WSL extension in Cursor:** Open Cursor (which is based on VS Code) on your Windows machine. Go to Extensions (Ctrl+Shift+X) and install "WSL" by Anysphere (the makers of Cursor). This allows Cursor to work with your WSL environment.
- **Open a WSL remote window:** Press Ctrl+Shift+P in Cursor, then select **"Remote-WSL: New Window"**. This will open a new Cursor window that is connected to your Ubuntu WSL instance.
- **Open the project folder:** In that WSL-connected window, use *File* -> *Open Folder* (or the equivalent) and navigate to `~/projects/agents` (your project root in WSL). Open that folder. (The first time you do this, it might take a little while to set up the server in WSL – grab a coffee as suggested ☕.)
- **Install Python & Jupyter extensions (in WSL):** Inside the WSL remote window of Cursor, install the **Python** (ms-python) and **Jupyter** extensions if they aren't already installed. Be sure to click "Install in WSL: Ubuntu" if prompted, so they install in the WSL context.

After that, Cursor should be configured to run and debug the project in the WSL environment. You'll have your code open on the Linux side with all the proper tools.

By following the above steps carefully, you should be able to get through Part 2 without issues. The key takeaway for the part you were stuck on: when the instructions say *"From a PowerShell, run* `ubuntu` *(rather than* `wsl`*)"*, it literally means **open PowerShell in Windows and type** `ubuntu` to start your WSL Ubuntu session in the right place [1]. Once you do that, the rest of the installation and setup occurs inside that Linux shell. Good luck with the rest of your setup! 👍

Sources:

- Microsoft Learn – *WSL Development Environment Setup* (advice on using Linux home directory for projects) [2] [3].
- SuperUser Q&A – *Difference between launching WSL with* `ubuntu` *vs* `wsl` [1].
- *Astral UV Installation Guide* – UV install script and notes on reloading shell [4].

---

[1] command line - How to set WSL default path to ~ (Ubuntu) - Super User
https://superuser.com/questions/1771813/how-to-set-wsl-default-path-to-ubuntu

[2] [3] Set up a WSL development environment | Microsoft Learn
https://learn.microsoft.com/en-us/windows/wsl/setup/environment

[4] Setting Up Django for Success
https://jilles.me/setting-up-django-for-success/