Getting Started with Linux (WSL), Claude Code, and GitHub 💻

This guide will walk you through installing the Windows Subsystem for Linux (WSL), setting up your environment with useful tools like Zsh and Node.js, installing Claude Code for AI-powered coding assistance, and managing your projects with Git and GitHub.

🛠️ Part 1: Installation and Setup

Follow these steps to get your development environment ready.

1. Installing Linux via Windows Subsystem for Linux (WSL)

- In PowerShell, run the command: `wsl --install .`
- Open the Microsoft Store (search for "store" after hitting the Windows key).
- Find and install Ubuntu from the Microsoft Store.
- Enable the Windows Subsystem for Linux feature:
  - Hit the Windows key and search for "features".
  - Select "Turn Windows features on or off".
  - Check the box for "Windows Subsystem for Linux" and click OK.
- Restart your computer.
- After restarting, open the Ubuntu application (you can find it in the Microsoft Store or search for it).
- Set up your username and password when prompted.

2. Updating Packages and Installing Zsh

- Update all packages in your WSL terminal by running: `sudo apt update && sudo apt upgrade -y`
- Install Zsh (a powerful shell alternative to bash): `sudo apt-get install zsh`
- Make Zsh your default shell: `chsh -s $(which zsh)`
- Close the terminal and restart it. You might be prompted to select an option; choose option 2.
- Enhance your Zsh experience by installing Oh My Zsh. You can find instructions here: https://www.sitepoint.com/zsh-tips-tricks/#1installohmyzsh.

3. Installing Node.js

- Install Node.js and npm (Node Package Manager) with the following command in your WSL terminal: `sudo apt install -y nodejs npm`
- Check the installed versions to ensure they were installed correctly: `node -v  npm -v`

4. Installing Claude Code

- Follow the official instructions to install Claude Code: https://docs.anthropic.com/en/docs/claude-code/getting-started.
- If you encounter issues, especially on Windows, refer to the troubleshooting guide: https://docs.anthropic.com/en/docs/claude-code/troubleshooting#linux-permission-issues.

🚀 Part 2: Using Linux and Claude Code

Now that everything is installed, let's learn how to use it.

1. Accessing Your Linux Environment

- The easiest and preferred way to access your WSL environment is by hitting the Windows key and typing `wsl`, similar to how you would open Command Prompt (cmd).
- Alternatively, you can connect to WSL through VS Code or Cursor:
  - Open VS Code or Cursor.
  - Click on the "connect" button (it looks like `><`) in the bottom left-hand corner (this opens a remote window).
  - Select "Connect to WSL using Distro" and then choose Ubuntu.
  - You can then open the WSL terminal directly in VS Code or Cursor, which should now list `zsh`.

2. Navigating to Project Files

- To navigate to your project files located on your Windows C: drive from the WSL terminal, use the `cd` (change directory) command. The path will start with `/mnt/c/`. For example: `cd /mnt/c/Users/your_username/path/to/your/project` (replace `your_username` and `path/to/your/project` accordingly).

3. Opening Projects in VS Code or Cursor

- Once you've navigated to your project directory in the WSL terminal, you can open it in your preferred editor:
  - Type `code .` to open the current directory in VS Code.
  - Type `cursor .` to open the current directory in Cursor.
- Tip: Many developers prefer to keep their Python projects directly within the Linux root directory (e.g., in your WSL Ubuntu home directory) rather than accessing them from the Windows file system.

4. Creating and Activating Python Virtual Environments

It's a best practice to use virtual environments for your Python projects to manage dependencies.

- Create a virtual environment (if you haven't already) by running the following command in your project directory within the WSL terminal: `python3 -m venv .venv` This creates a hidden directory named `.venv` containing the virtual environment.
- Activate the virtual environment: `source .venv/bin/activate` Your terminal prompt should change to indicate that the virtual environment is active.
- Alternative Activation: You might sometimes need to select the interpreter manually. Press `CTRL + Shift + P` in VS Code/Cursor and search for/select the `.venv` interpreter.

5. Using Claude Code

- Once your project is open and your virtual environment is activated (if needed), you can launch Claude Code by typing the following command in the terminal: `claude` This should open Claude Code within your project context.

- Note: You may occasionally need to close and reopen the terminal during this process.

🐣 Part 3: Git and GitHub Workflow

Version control is crucial for any developer. Here's how to use Git and GitHub.

1. Cloning an Existing Repository from GitHub

- Go to GitHub and create a new repository or navigate to an existing one.
- Copy the URL of the repository.

- In your WSL terminal, navigate to where you want to store the project, then clone the repository: `git clone YOUR_REPOSITORY_URL` (replace `YOUR_REPOSITORY_URL` with the copied URL).
- You might be prompted to enter your GitHub username (e.g., 'GormleyLab') and a password. For the password, you'll likely need to generate a Personal Access Token (PAT) from GitHub:
  - Go to GitHub Settings → Developer Settings → Personal access tokens.
- Change to the new repository's directory: `cd repository_name` (replace `repository_name` with the actual folder name).

2. Setting Up the Project Environment

- Inside the cloned repository directory, create and activate a virtual environment as described earlier: `python3 -m venv .venv  source .venv/bin/activate`

3. Syncing and Working

- Pull the latest changes from GitHub to ensure your local copy is up to date: `git pull origin main`
- Then, open the project in your editor: `cursor .` (or `code .`)

4. Making Changes and Pushing to GitHub

- When you make changes to your project files:
  - Stage all changed files: `git add .`
  - Commit your changes locally with a descriptive message: `git commit -m "Your meaningful commit message here"`
  - Push your changes to GitHub: The first time you push to a new branch or repository, you might need to specify the upstream branch: `git push -u origin main` After the initial push with `-u`, you can simply use: `git push`

5. Opening an Existing Local Repository

- When you return to work on an existing local repository that's connected to GitHub, always start by pulling the latest updates: `git pull origin main`

Happy coding! 🎉