Your local chatbot - an amateurs introduction to Open-WebUI with Ollama

Otto Lilja-Lund

I've been eyeing to run local llms and finally jumped the gun. But how do you actually do it? I share with you my fumbling steps towards local AI. As someone starting without any advanced knowledge I hope that anyone out there, inexperienced as me, but curious how to start, joins me on my journey towards local AI.

Oh, and I am also learning Linux - with Ubuntu to be more specific. Me learning Linux and local AI are personal steps to break free from big tech, and to understand more of what is going on under the hood of the technology we use everyday.

Where to start?

Hardware is separate topic so I won't go into that here. I have an NVIDIA GPU with 24 GB of VRAM, which means that I have a dedicated graphic card that will do most, if not all, of the computations needed for local AI.

> Tip
>
> {bash} is commands you run in a terminal, powershell or command lines. Copy - paste - enter from the code-blocks in the article and you are good to go! You'll basically do everything in a terminal with {bash} from here on. Oh, and you'll use sudo a lot. Using sudo prompts you to enter your password, sudo temporarily elevates the users's privileges. The `sudo`is active for a short period of time, hence you migth want to add `sudo` to our commands if the don't run.

1. **Install Ollama**

   Ollama is a tool that runs llms directly in the terminal or though other services - like Open-WebUI.

   ```
   sudo curl -fsSL https://ollama.com/install.sh | sh
   ```

2. **Pull a model**

   Once Ollama is installed you can get your model of choice from Ollama with a simple line. Check out https://ollama.com/search to find a model of interest. I recommend to start with a small one, e.g. 3b, just to see that things work before testing what your hardware manages - things might get hot! But more on that later. I use Mistral:7b in the example. You'll find the snippet to run on the page for the model of choice. Run the command run to start your model. Executing run will pull the model and run it the first time. Once it's downloaded ("pulled") the run command will just start the model. You can use `pull` if you only want to download it.

   ```
   ollama run mistral
   ```

   Cool! You can now chat with your own llm in the terminal! When you are done just close that terminal.

   > **Tip**
   >
   > Enter `ollama` to see what commands you can use. `list` lists available models, `ps` shows what processes the models runs on (e.g. 100% GPU or 45% CPU / 55% GPU)

3. **Install Open-WebUI**

   Open-WebUI gives you a user-interface similar to ChatGPT. You'll recognize how things work, like drop-and-drag images and saved chats. There is one thing that made this step a bit more tricky - Docker. It means that you'll run the application in a "safe environment" or inside a container. Run the line and let it cook for a while!

   ```
   sudo docker run -p 8080:8080 ghcr.io/open-webui/open-webui:main
   ```

   Once it's downloaded and ready (this may take some time) you should be able to see Open-WebUI in your browser at http://localhost:8080

   There you'll be prompted to register an admin account the first time. And you are in!

   > **Important**
   >
   > To restart the Open-Web-UI session you should rename the container ID (CID).
   >
   > - Get the container ID from Docker.
   >
   > ```
   > sudo docker ps
   > ```
   >
   > You should get something like this:
   > CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |

d22949556b11 | ghcr.io/open-webui/open-webui:main | "bash start.sh" | 45 minutes ago| Up 45 minutes (healthy) | 0.0.0.0:8080-8080/tcp, :8080-8080/tcp | stupefied_lamarr

- In this example the CID is *d22949556b11*. Rename the CID to *Open-WebUI*:

```
docker rename d22949556b11 Open-WebUI
```

- To restart just enter `docker start Open-WebUI` and it's active agian.

4. **Connect Ollama with Open-WebUI**

   Now that you have Ollama with a model and Open-WebUI up and running all you need to do is connect the two. Ollama runs by default on `http://localhost:11434` or `http://127.0.0.1:11434` (Note, localhost and `127.0.0.1` is not the same but that is beyond the scope here.)

   - Go to *Admin Settings* in Open-WebUI.
   - Navigate to *Connections* > *Ollama* > *Manage* (click the wrench icon).
   - Set Ollama API connection to `http://172.17.0.1:11434` (when Open-WebUI runs in docker). The default Ollama connection did not work for me but this did the trick.

5. **Start chatting**

   Congratulations! You now have your own "ChatGPT"!!

   Your model running in Ollama should pop up and you can start chatting all you want on your own hardware! pull more models and choose them in the drop-down in Open-WebUI. There you can unload models as well (to save processing, they are still downloaded but not active in the CPU/GPU).

Troubleshooting

I wrote the steps that worked for me, but I stumbled on my way to get there. Some hurdles were that Ollama and Open-WebUI did not communicate, my model ran fine on GPU or GPU/CPU mixed in the terminal but only CPU from Open-WebUI! And I did not know how to restart Open-WebUI and re-installed instead...

Here are some commands that came in handy:

Set parameters for GPU in `ollama` (GPU_LAYERS=-1 means all available layers; CUDA_VISIBLE_DEVICES=0 means only one card) and set the port for the host.

```
export OLLAMA_HOST=0.0.0.0:11434
export OLLAMA_ORIGINS="*"
export OLLAMA_GPU_LAYERS=-1
export CUDA_VISIBLE_DEVICES=0
```

If it works, you'll get no feedback. But to verify they're all set:

```
echo "OLLAMA_HOST: $OLLAMA_HOST"
echo "OLLAMA_ORIGINS: $OLLAMA_ORIGINS"
echo "OLLAMA_GPU_LAYERS: $OLLAMA_GPU_LAYERS"
echo "CUDA_VISIBLE_DEVICES: $CUDA_VISIBLE_DEVICES"
```

You should see:

```
OLLAMA_HOST: 0.0.0.0:11434
OLLAMA_ORIGINS: *
OLLAMA_GPU_LAYERS: -1
CUDA_VISIBLE_DEVICES: 0
```

Verify if Ollama is responding in localhost and what models are available: `curl http://localhost:11434/api/tags`. Check if *port :11434* is free: `sudo lsof -i :1143` Stop Ollama "across the board": `sudo systemctl stop ollama`and start: `sudo systemctl start ollama`

Check activity for NVIDIA GPU: `watch nvidia-smi`

See active processes. PID means Process ID, and you can handle them from *htop*: `sudo apt install htop`. Run `htop`to see activities.

See containers in Docker: `docker ps`. To see the configurations of a container `docker inspect <insert CID or name>`. Restarting the Docker service can resolve issues: `sudo systemctl restart docker`.

Ensure that port 8080 is not being used by another service. You can check this with: `sudo lsof -i :808`

Linux Suspend Resume (from Ollama documentation)

https://github.com/ollama/ollama/blob/main/docs/faq.md

On Linux, after a suspend/resume cycle, sometimes Ollama will fail to discover your NVIDIA GPU, and fallback to running on the CPU. You can workaround this driver bug by reloading the NVIDIA UVM driver with

```
sudo rmmod nvidia_uvm && sudo modprobe nvidia_uvm
```

That should be it! Happy chatting!