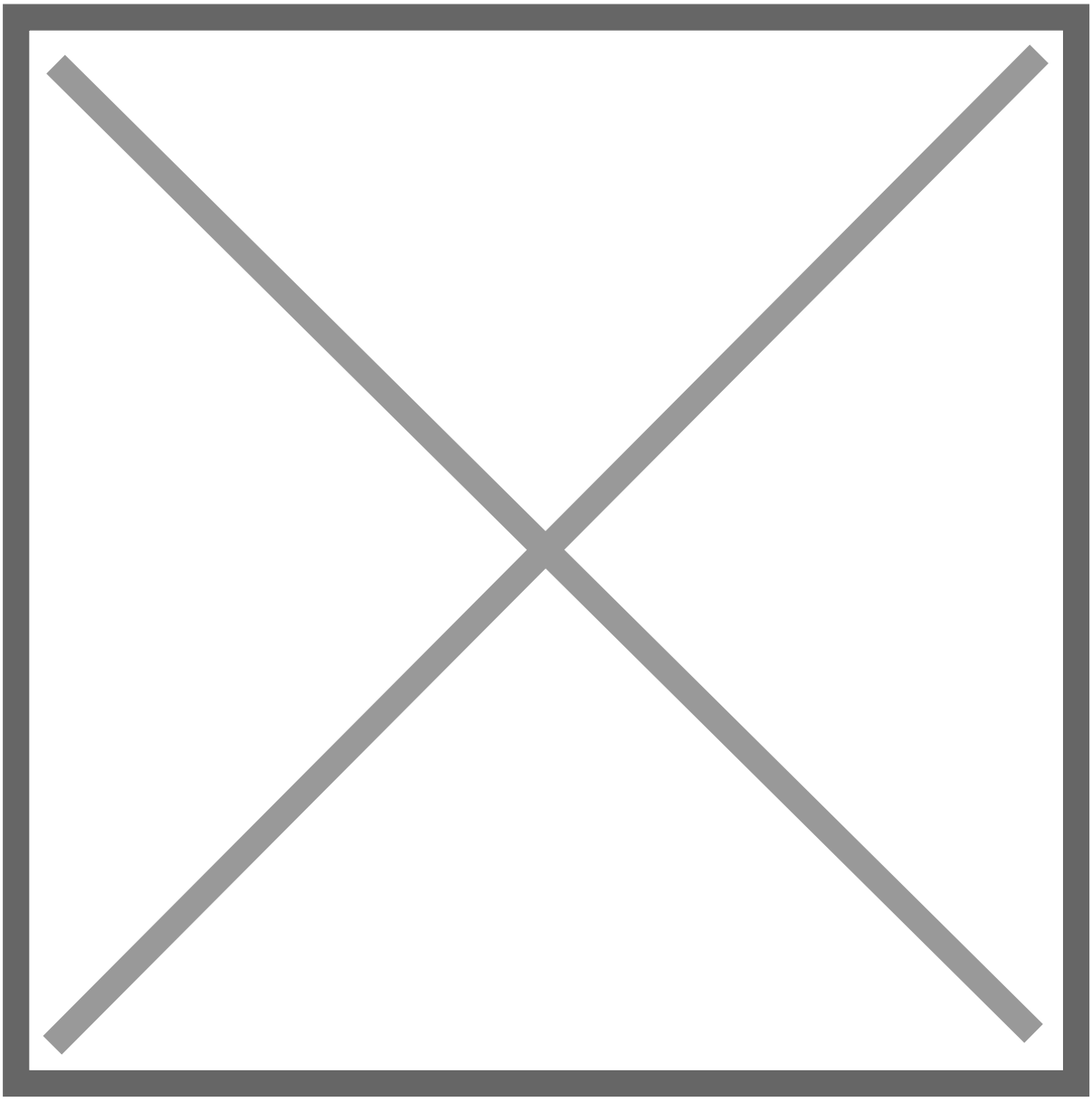
Instalação e configuração Ollama

- How to locally deploy ollama and Open-WebUI with Docker Compose
- Monitoração DeepSeek Ollama

How to locally deploy ollama and Open-WebUI with Docker Compose

Link: https://medium.com/@edu.ukulelekim/how-to-locally-deploy-ollama-and-open-webui-with-docker-compose-318f0582e01f

May 21, 2024



ollama and Open-WebUI performs like ChatGPT in local.

There are so many web services using LLM like ChatGPT, while some tools are developed to run the LLM locally.

One of them is **ollama** which makes you interact with LLM locally. And if you have local LLM rather than the ones based on other remote server, you can…

- Avoid personal information leaks
- Test the open LLM with no additional cost

So I'll deploy **ollama** with open LLM, **llama3** on my laptop.

The specification of the laptop is as below:

- CPU: AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz
- RAM: 32 GB

And I'll use **Open-WebUI** which can easily interact with ollama on the web browser.

Open-WebUI has a web UI similar to ChatGPT, and you can configure the connected LLM from ollama on the web UI as well.

I run **ollama and Open-WebUI on container** because each tool can provide its feature consistently in the independent environment from the host.

Fortunately, there are official Docker images for both of the tools, so it become easier to deploy them locally.

When deploying containerized ollama and Open-WebUI, I'll use Docker Compose which can run multiple container with consistent configuration at once.

With this article, you can understand how to deploy ollama and Open-WebUI locally with Docker Compose.

So, let's start with defining *compose.yaml* file that Docker Compose uses to deploy the containers.

```
services:
  openWebUI:
    image: ghcr.io/open-webui/open-webui:main
    restart: always
    ports:
      - "3000:8080"
    extra_hosts:
      - "host.docker.internal:host-gateway"
    volumes:
      - open-webui-local:/app/backend/data
```

```
  ollama:
    image: ollama/ollama:0.1.34
    ports:
      - "11434:11434"
    volumes:
      - ollama-local:/root/.ollama

volumes:
  ollama-local:
    external: true
  open-webui-local:
    external: true
```

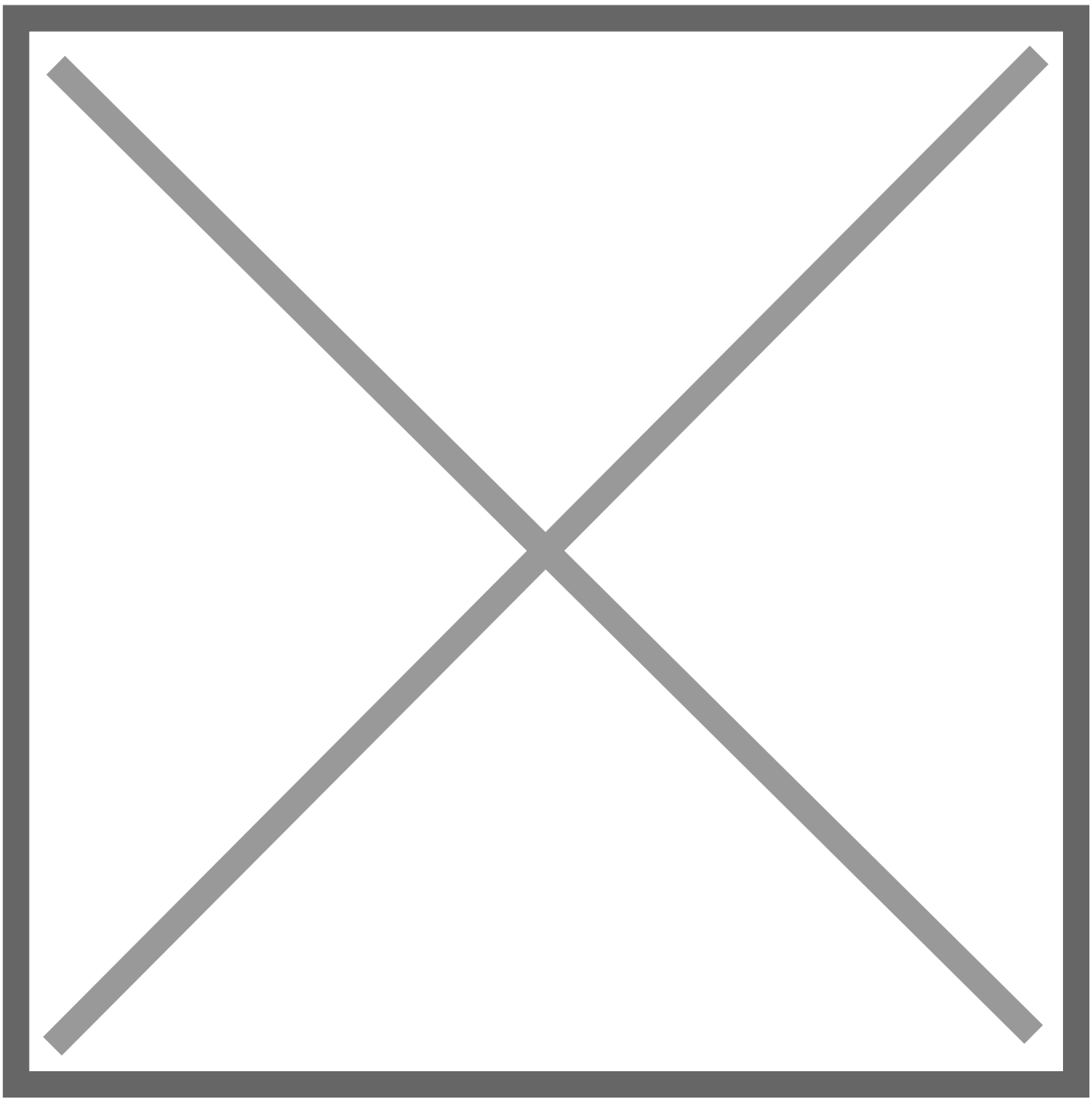
Next step is creating **Docker Volume** with Docker CLI. **Docker Volume** is a storage that Docker Container uses to save their data.

As defining on the above *compose.yaml* file, I need to create two volume *ollama-local* and *open-webui-local*, which are for ollama and open-webui, with the below commands on CLI.

- **docker volume create ollama-local**
- **docker volume create open-webui-local**

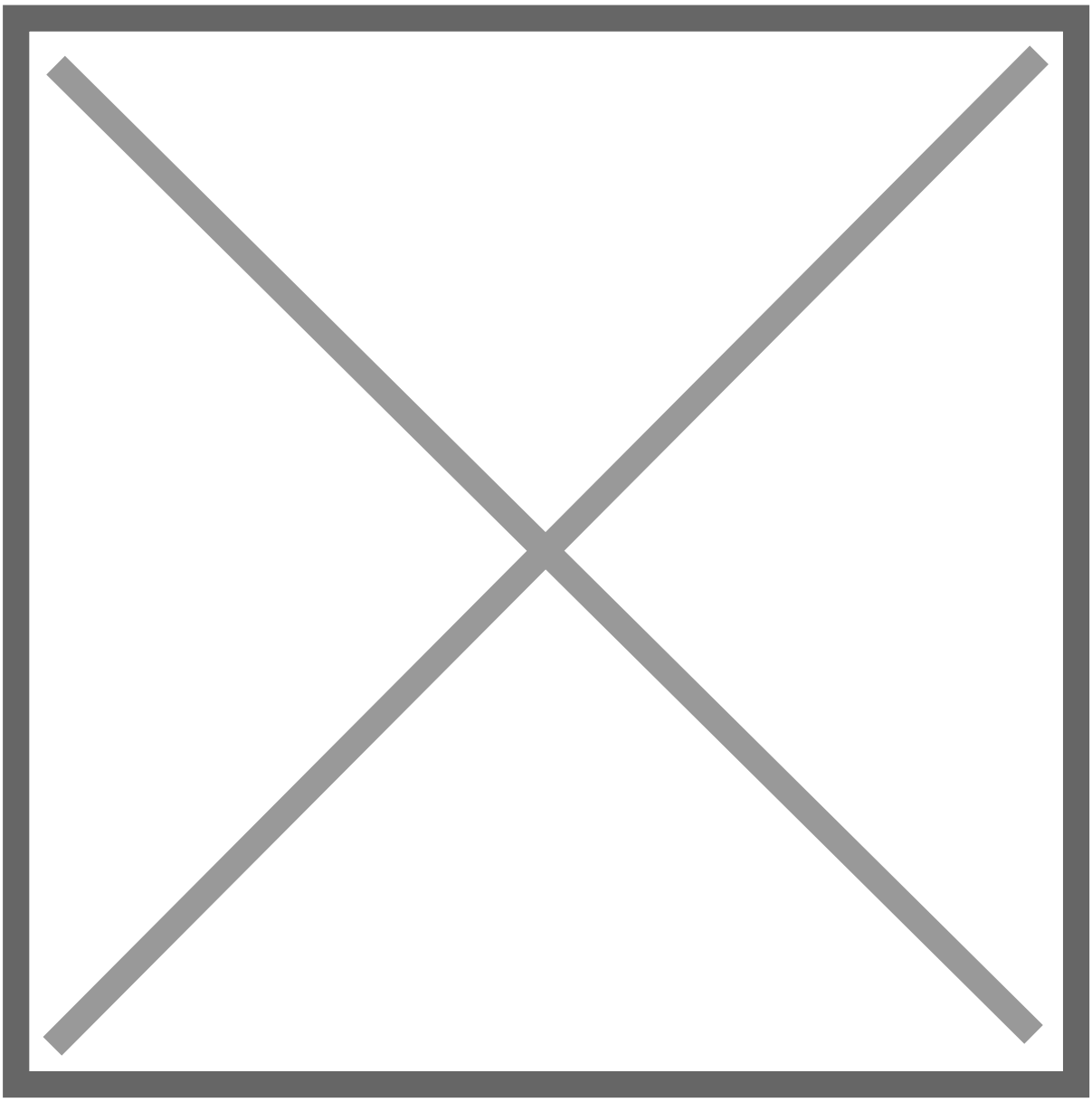
Now, I'll deploy these two containers on local with docker compose command. Before that, let's check if the compose yaml file can run appropriately. We can dry run the yaml file with the below command.

- **docker compose — dry-run up -d** (On path including the *compose.yaml*)

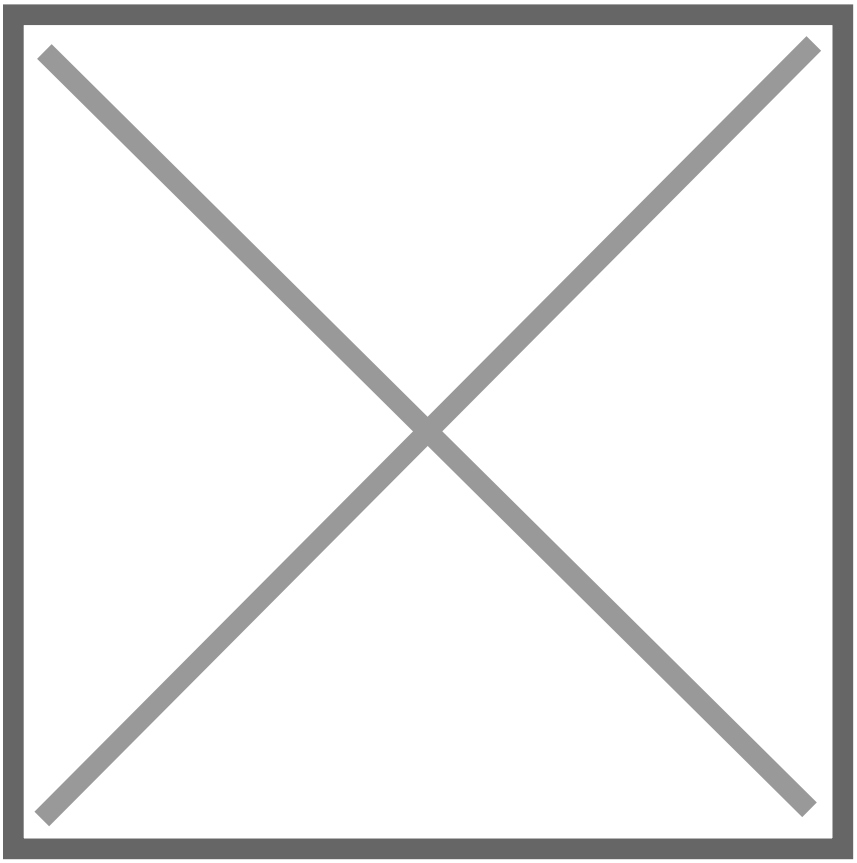


After dry running, we can see that it runs appropriately. So let’s deploy the containers with the below command

- **docker compose up -d** (On path including the *compose.yaml*)

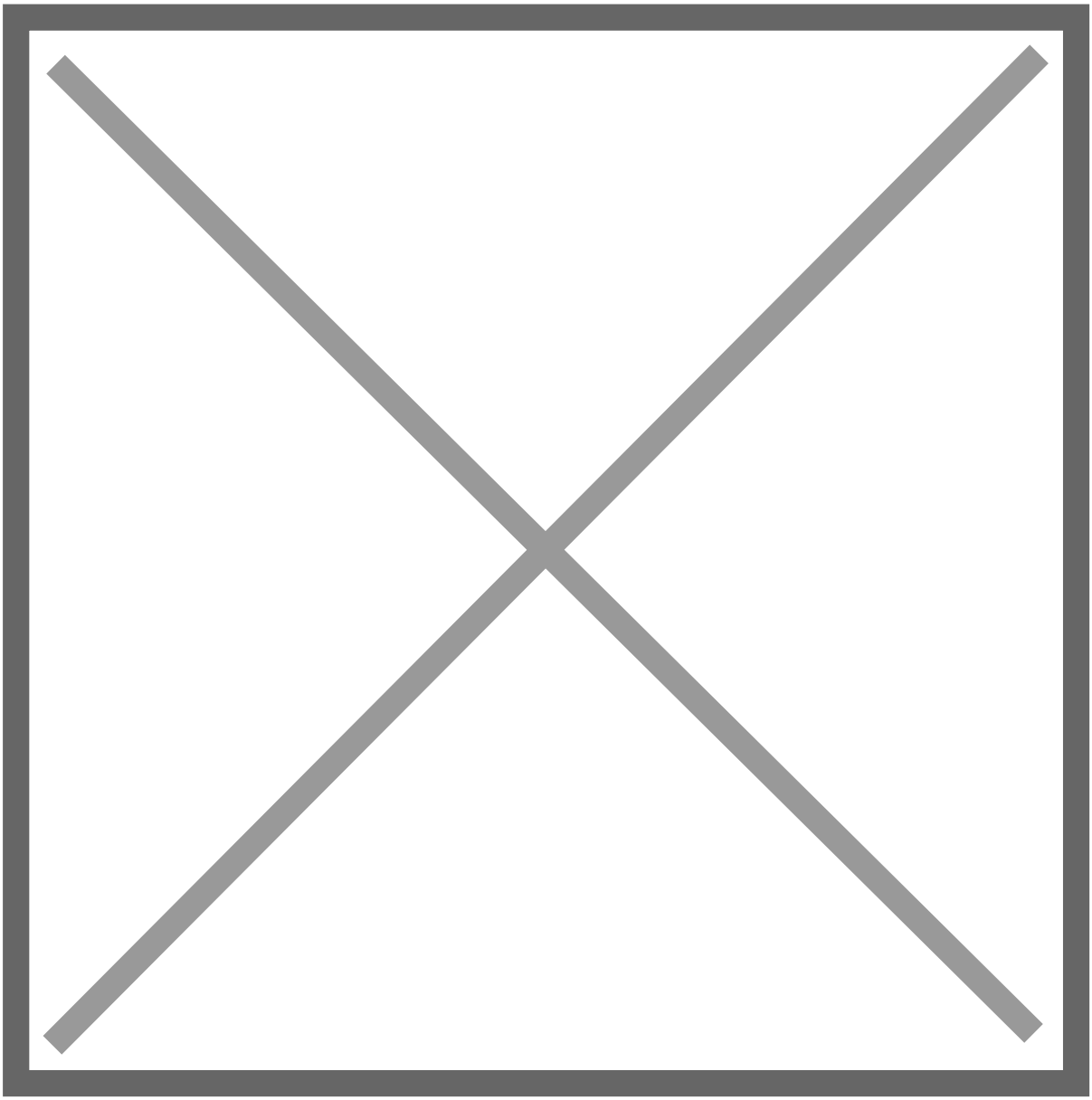


After checking a message that all of the container start, we can access Open-WebUI on the browser with its port number defined on *compose.yaml* (In this practice, access link is http://localhost:3000)

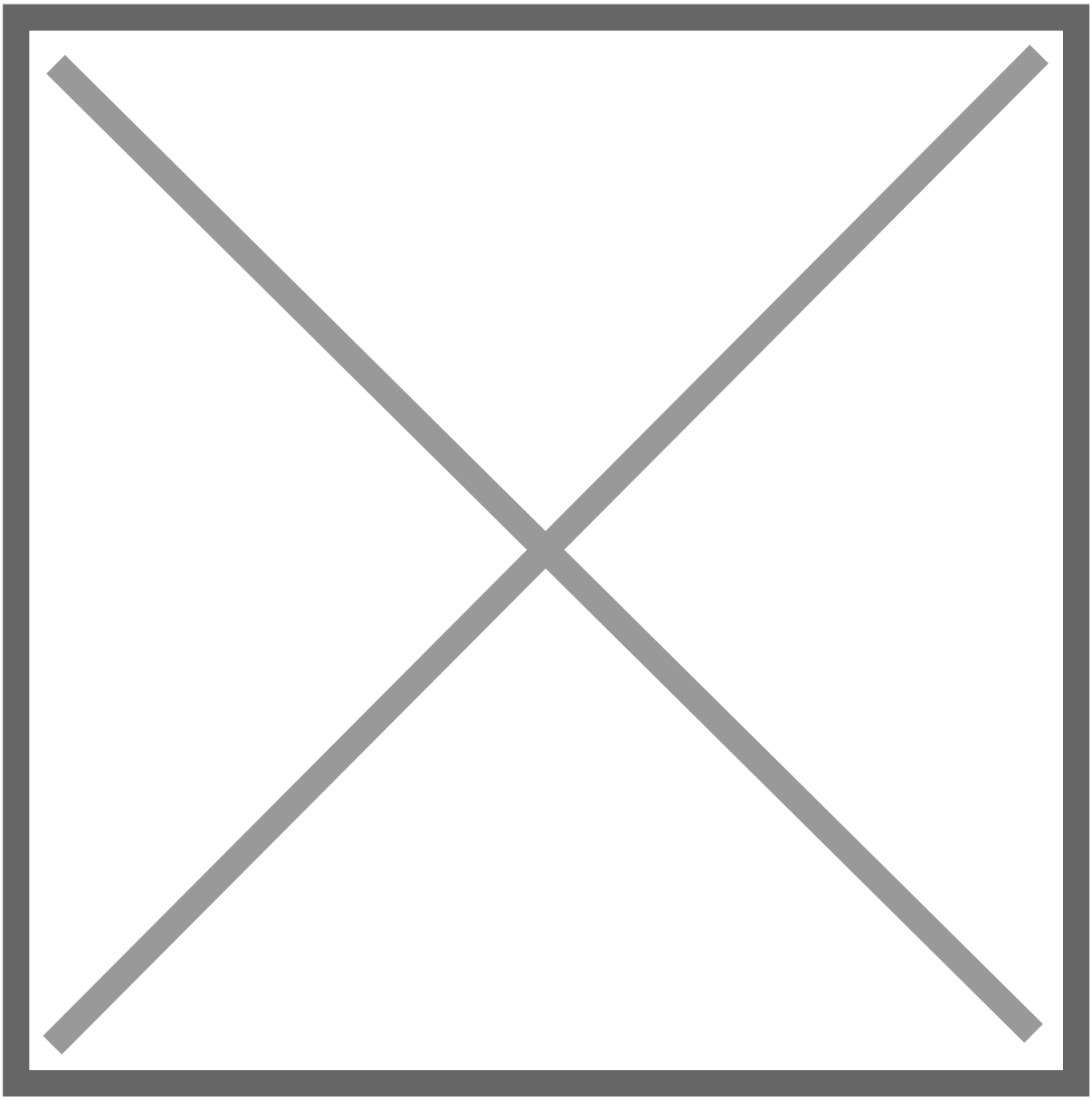


If you access the Open-WebUI first, you need to sign up.

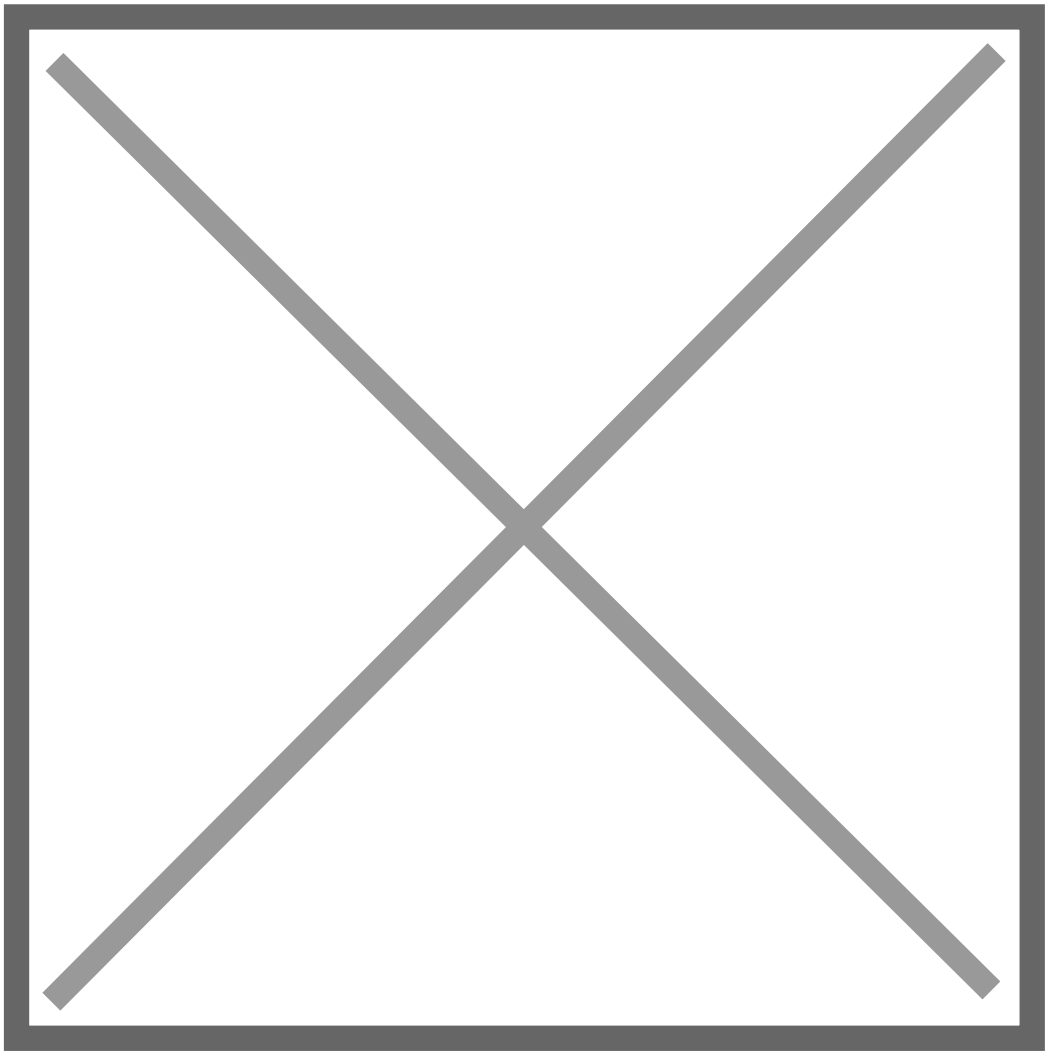
After accessing to the Open-WebU, I need to sign up for this system. My account for the system will be stored on its Docker volume, so the sign-up will be required only for the first time.



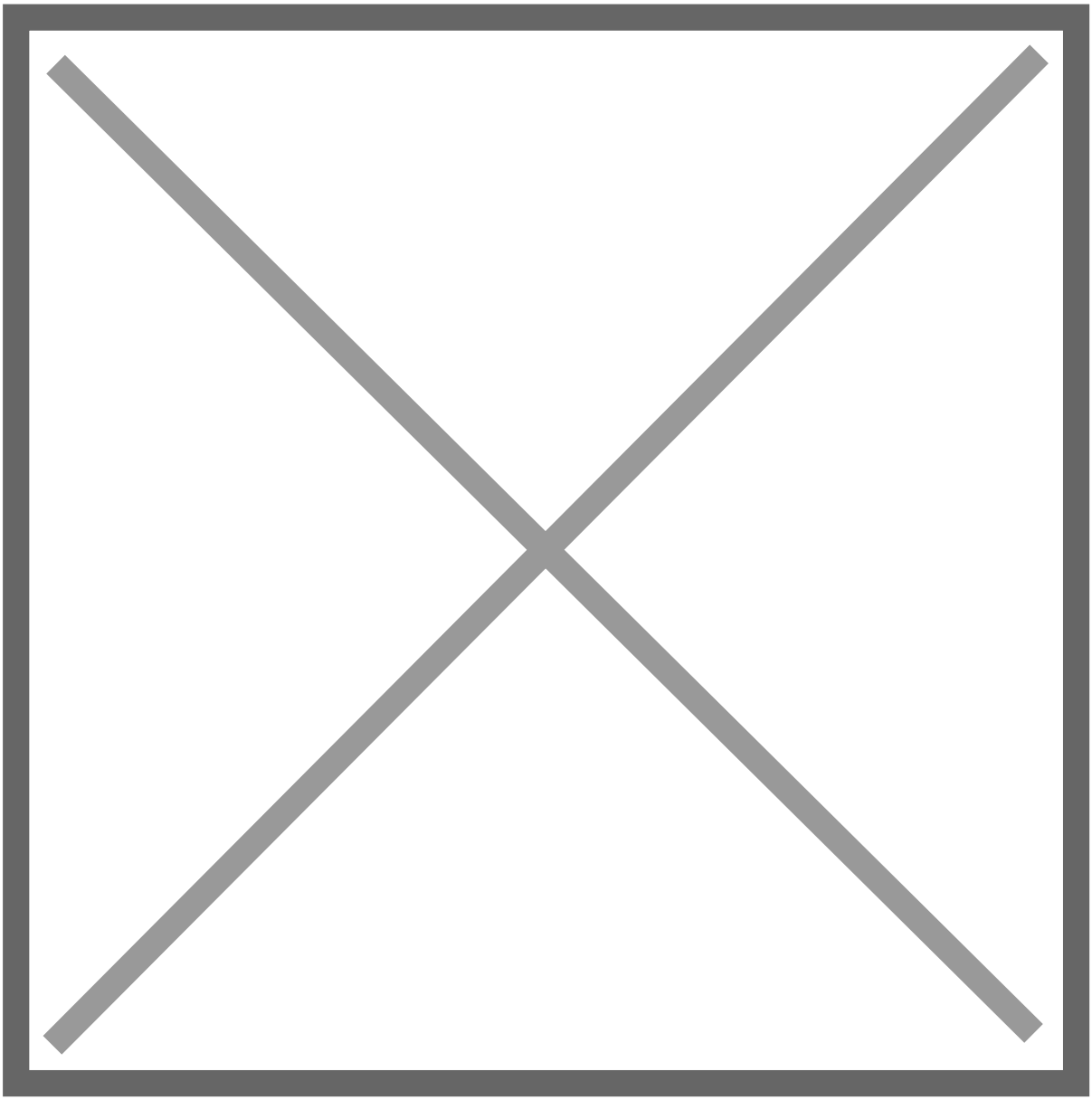
Since there is no LLM model on ollama yet, we need to pull open LLM by inserting its tag on '**Pull a model from Ollama.com**' in **models** menu which will be displayed after pushing a gear button on top right of the home window. The setting window is like below:



If a desired LLM is pulled, you can select downloaded model on top left of the home window like the below capture:



Finally, we can chat with the open LLM, llama3 in this practice, like ChatGPT.



There are some additional Docker commands that you can use to manage containers.

If you want to stop and remove containers with Docker Compose, use the below command.

- **docker compose down** (On path including the *compose.yaml*)

If you want to remove the Docker volumes which ollama and Open-WebUI are using, for the further storage management, use the below command. (**Warning: You can’t restore the removed volumes which not backed up.**)

- **docker volume rm ollama-local**
- **docker volume rm open-webui-local**

That’s all. I think the local LLM service is pretty interesting project.

I’ll share if there are any awesome use cases with the local LLM.

Please let me know what you think :)

References

- https://github.com/ollama/ollama
- https://github.com/open-webui/open-webui

Monitoração DeepSeek Ollama

Link: https://blog.networkchuck.com/posts/is-deepseek-safe-to-run-locally/

Is DeepSeek Safe To Run (locally)

2025-01-31

#NetworkChuck

Monitor Network Connections

Windows

```
while($true) {
    Get-Process ollama | ForEach-Object {
        $id = $_.Id
        Write-Host "`nConnections for Ollama process $id" -ForegroundColor Green
        Get-NetTCPConnection | Where-Object OwningProcess -eq $id | Select-Object
LocalAddress, LocalPort, RemoteAddress, RemotePort, State
    }
    Start-Sleep -Seconds 2
    Clear-Host
}
```

Mac

```
# One-time check
pid=$(pgrep ollama)
lsof -i -P -n | grep ollama

# For continuous monitoring
while true; do
    echo "$(date): Ollama Connections"
    lsof -i -P -n | grep ollama
```

```
    sleep 2
    clear
done

# Alternative using netstat
netstat -p tcp -v | grep ollama
```

Linux

```
# One-time check
pid=$(pgrep ollama)
lsof -i -P -n | grep ollama

# Or for continuous monitoring
watch -n 2 "lsof -i -P -n | grep ollama"

# Alternative using netstat
netstat -np | grep ollama

# Or using ss (more modern)
ss -np | grep ollama
```

Running Ollama Inside Docker

```
docker run -d \
--gpus all \
-v ollama:/root/.ollama \
-p 11434:11434 \
--security-opt=no-new-privileges \
--cap-drop=ALL \
--cap-add=SYS_NICE \
--memory=8g \
--memory-swap=8g \
--cpus=4 \
--read-only \
--name ollama \
ollama/ollama
```