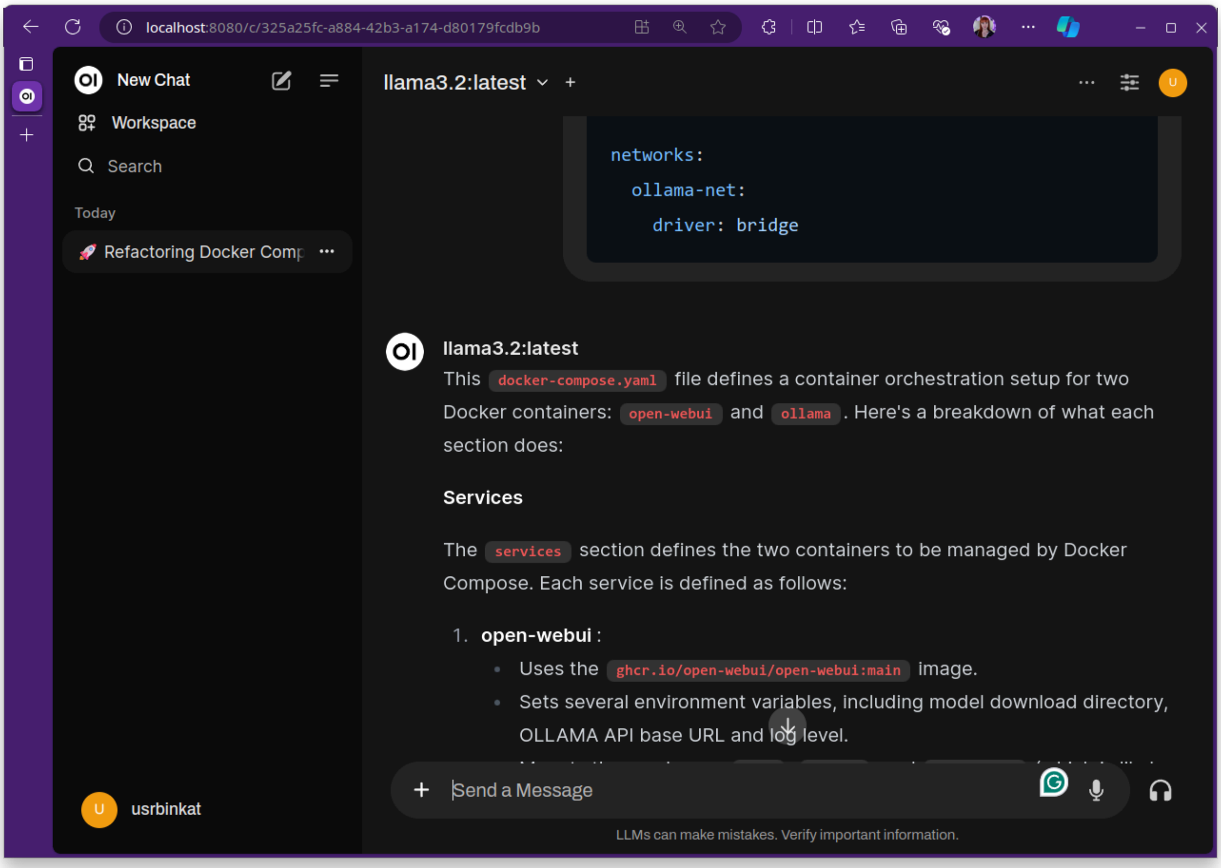
usrbinkat / README.md

Last active 3 days ago • Report abuse

Code Revisions 3 Stars 48 Forks 12

Ollama + Open-Webui + Nvidia/CUDA + Docker + docker-compose

README.md


localhost:8080/c/325a25fc-a884-42b3-a174-d80179fcdb9b
New Chat
Workspace
Search
Today
Refactoring Docker Comp
usrbinkat
llama3.2:latest
Send a Message
LLMs can make mistakes. Verify important information.

```
networks:
  ollama-net:
    driver: bridge
```

llama3.2:latest

This `docker-compose.yaml` file defines a container orchestration setup for two Docker containers: `open-webui` and `ollama`. Here's a breakdown of what each section does:

Services

The `services` section defines the two containers to be managed by Docker Compose. Each service is defined as follows:

1. **open-webui**:
   - Uses the `ghcr.io/open-webui/open-webui:main` image.
   - Sets several environment variables, including model download directory, OLLAMA API base URL and log level.

UPDATE: This is tested and working on both Linux and Windows 11 used for LlaMa & DeepSeek

Here's a sample `README.md` file written by Llama3.2 using this docker-compose.yaml file that explains the purpose and usage of the Docker Compose configuration:

ollama-portal

A multi-container Docker application for serving OLLAMA API.

Overview

This repository provides a Docker Compose configuration for running two containers: `open-webui` and `ollama` . The `open-webui` container serves a web interface that interacts with the `ollama` container, which provides an API or service. This setup is designed to work together seamlessly, allowing users to access OLLAMA's functionality through a user-friendly web interface.

Architecture

The application consists of two main components:

- **OLLAMA**: A GPU-accelerated neural network inference service that provides a RESTful API for serving models.
- **Open-WebUI**: A web-based interface for interacting with the OLLAMA API, providing a simple and intuitive way to deploy and manage models.

Docker Compose Configuration

The Docker Compose configuration file (`docker-compose.yaml`) defines several key settings:

- **Services**: The application consists of two services: `open-webui` and `ollama` . Each service is defined with its own set of environment variables, volumes, and ports.
- **Environment Variables**:

- `MODEL_DOWNLOAD_DIR` : Specifies the directory for storing downloaded models.
- `OLLAMA_API_BASE_URL` : Sets the base URL for the OLLAMA API.
- `LOG_LEVEL` : Configures the log level for both containers.

- **Volumes**: The application mounts several volumes to share data between containers. These include:

- `data` : For storing user input and model artifacts.
- `models` : For accessing pre-trained models.
- `ollama` : For storing application-specific data.

Container Configuration

The Docker Compose configuration defines the following container configurations:

- **OLLAMA Container**:

- Uses the official OLLAMA image (`ollama/ollama:latest`).
- Specifies NVIDIA GPU acceleration using the `runtime: nvidia` option.
- Configures the container to use all available GPUs in the cluster.

- **Open-WebUI Container**:

- Uses the official Open-WebUI image (`ghcr.io/open-webui/open-webui:main`).
- Specifies environment variables for model download directories and OLLAMA API URLs.

Networking

The application uses a single network (`ollama-net`) that connects both containers. This allows them to communicate with each other seamlessly.

Running in Production

To run this application in production, you'll need to:

- Set up your OLLAMA API on the `ollama` container.
- Configure the `open-webui` container to connect to your OLLAMA API.
- Mount necessary volumes and adjust configuration variables as needed.

Troubleshooting

If you encounter issues while running this application, please refer to the Docker Compose troubleshooting guide for assistance.

Security Considerations

This application uses the following security measures:

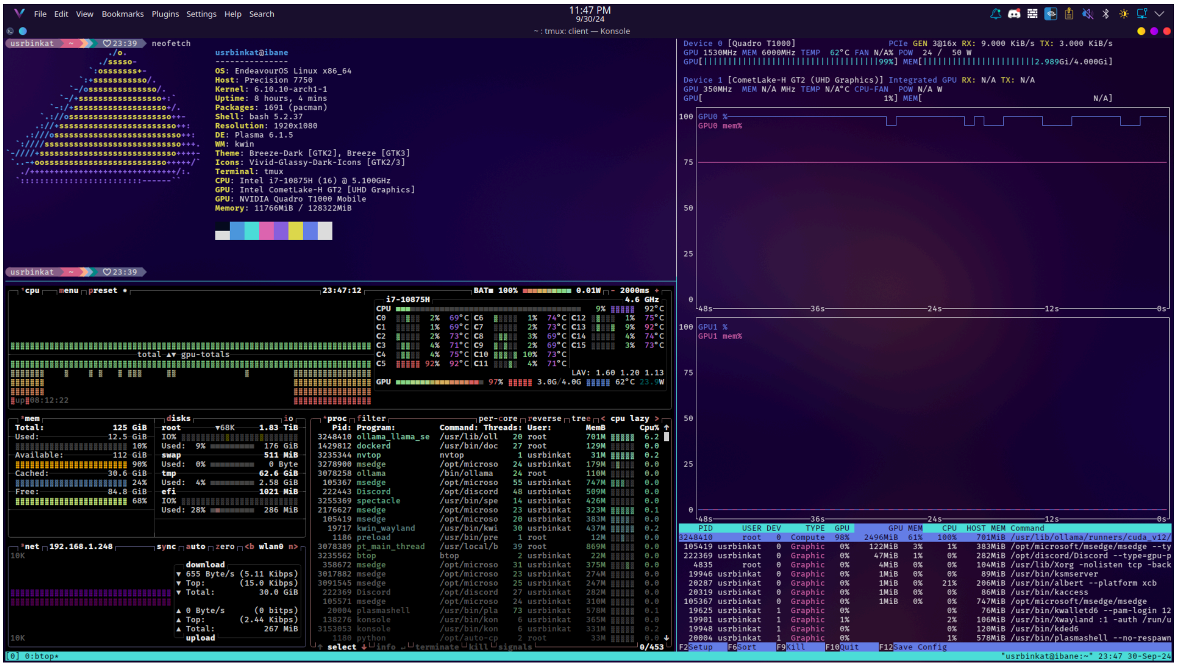
- **Model signing**: The OLLAMA API verifies model signatures using a digital certificate.
- **Input validation**: The Open-WebUI container validates user input to prevent injection attacks.
- **Encryption**: Data exchanged between containers is encrypted using SSL/TLS.

Performance Optimization

To optimize performance, consider the following:

- **Model caching**: Use a caching layer (e.g., Redis) to store frequently accessed models.
- **Container orchestration**: Use a container orchestration tool (e.g., Kubernetes) to manage and scale your containers.
- **GPU acceleration**: Configure multiple GPUs on your system for optimal performance.

This enhanced README file provides more in-depth technical explanations, including architecture, Docker Compose configuration, container configurations, networking, security considerations, and performance optimization. If you have any further questions or concerns, feel free to open a discussion on our GitHub page!

> ## Arch Linux (Elementary OS)
>
> 
>

```
usrbinkat ~ ♡23:39 neofetch
usrbinkat@ibane
---------------
OS: EndeavourOS Linux x86_64
Host: Precision 7750
Kernel: 6.10.10-arch1-1
Uptime: 8 hours, 4 mins
Packages: 1691 (pacman)
Shell: bash 5.2.37
Resolution: 1920x1080
DE: Plasma 6.1.5
WM: kwin
Theme: Breeze-Dark [GTK2], Breeze [GTK3]
Icons: Vivid-Glassy-Dark-Icons [GTK2/3]
Terminal: tmux
CPU: Intel i7-10875H (16) @ 5.100GHz
GPU: Intel CometLake-H GT2 [UHD Graphics]
GPU: NVIDIA Quadro T1000 Mobile
Memory: 11766MiB / 128322MiB
usrbinkat ~ ♡23:39
```

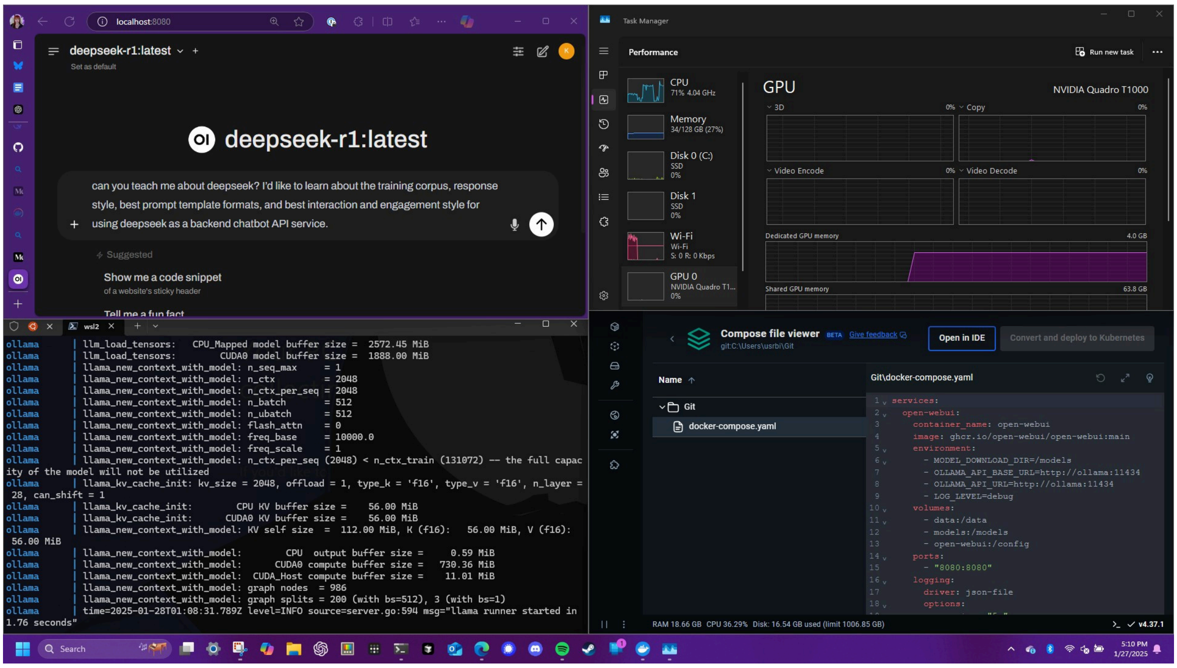
```
cpu  menu  preset *                         23:47:12        BAT 100%   0.01W  - 2000ms +
                                               i7-10875H                          4.6 GHz
                                               CPU                          9%    92°C
                                               C0   2%  69°C C6   1%  74°C C12   1%  75°C
                                               C1   1%  69°C C7   2%  73°C C13   9%  92°C
                                               C2   2%  73°C C8   3%  69°C C14   4%  74°C
                                               C3   4%  71°C C9   2%  69°C C15   3%  73°C
                 total ▲▼ gpu-totals           C4   4%  75°C C10 10%  73°C
                                               C5  92%  92°C C11  4%  71°C
                                                                 LAV: 1.60 1.20 1.13
up 08:12:22                                    GPU   97%  3.0G/4.0G  62°C 23.9W

mem                     disks              io   proc filter           per-core reverse tree < cpu lazy >
Total:       125 GiB    root   ▼68K  1.83 TiB      Pid: Program:        Command:    Threads: User:      MemB   Cpu%
Used:       12.5 GiB    IO%                    3248410 ollama_llama_se /usr/lib/oll  20 root       701M   6.2
                 10%    Used:  9%    176 GiB   1429812 dockerd         /usr/bin/doc  27 root       129M   0.0
Available:   112 GiB    swap         511 MiB   3235344 nvtop           nvtop          1 usrbinkat   31M   0.2
                 90%    Used:  0%     0 Byte   3278900 msedge          /opt/microso  24 usrbinkat  179M   0.0
Cached:     30.6 GiB    tmp         62.6 GiB   3078258 ollama          /bin/ollama   24 root       110M   0.0
                 24%    Used:  4%   2.58 GiB    105367 msedge          /opt/microso  55 usrbinkat  747M   0.0
Free:       84.8 GiB    efi         1021 MiB    222443 Discord         /opt/discord  48 usrbinkat  509M   0.0
                 68%    IO%                    3255369 spectacle       /usr/bin/spe  14 usrbinkat  426M   0.0
                        Used: 28%    286 MiB   2176627 msedge          /opt/microso  23 usrbinkat  323M   0.1
                                                105419 msedge          /opt/microso  20 usrbinkat  383M   0.0
                                                 19717 kwin_wayland    /usr/bin/kwi  30 usrbinkat  437M   0.2
                                                  1186 preload         /usr/bin/pre   1 root        12M   0.0
net 192.168.1.248     sync auto zero <b wlan0 n>  3078389 pt_main_thread /usr/local/b 39 root      869M   0.0
10K                                            3235562 btop            btop           2 usrbinkat   22M   0.0
                 download                       358672 msedge          /opt/microso  31 usrbinkat  375M   0.0
                 ▼ 655 Byte/s  (5.11 Kibps)    3017882 msedge          /opt/microso  23 usrbinkat  274M   0.0
                 ▼ Top:        (15.0 Kibps)    3091545 msedge          /opt/microso  25 usrbinkat  247M   0.0
                 ▼ Total:         30.0 GiB      222369 Discord         /opt/discord  27 usrbinkat  282M   0.0
                                                105571 msedge          /opt/microso  24 usrbinkat  310M   0.0
                 ▲ 0 Byte/s      (0 bitps)       20004 plasmashell     /usr/bin/pla  73 usrbinkat  578M   0.1
                 ▲ Top:       (2.44 Kibps)      138276 konsole         /usr/bin/kon   6 usrbinkat  365M   0.0
                 ▲ Total:          267 MiB     3153053 konsole         /usr/bin/kon   6 usrbinkat  331M   0.2
10K              upload                           1180 python          /opt/auto-cp   2 root        33M   0.0
                                               ↑ select ↓ info ↵ terminate kill signals              0/453
[0] 0:btop*
```

```
Device 0 [Quadro T1000]               PCIe GEN 3@16x RX: 9.000 KiB/s TX: 3.000 KiB/s
GPU 1530MHz MEM 6000MHz TEMP  62°C FAN N/A% POW  24 /  50 W
GPU[||||||||||||||||||||||||||||||||||99%] MEM[|||||||||||||||||||||||2.989Gi/4.000Gi]

Device 1 [CometLake-H GT2 (UHD Graphics)] Integrated GPU RX: N/A TX: N/A
GPU 350MHz  MEM N/A MHz TEMP N/A°C CPU-FAN  POW N/A W
GPU[                                  1%] MEM[                                   N/A]

100 GPU0 %
    GPU0 mem%
 75
 50
 25
  0
   48s          36s          24s          12s          0s
100 GPU1 %
    GPU1 mem%
 75
 50
 25
  0
   48s          36s          24s          12s          0s
    PID      USER DEV    TYPE  GPU     GPU MEM    CPU  HOST MEM Command
3248410      root   0 Compute  98%  2496MiB 61%  100%   701MiB /usr/lib/ollama/runners/cuda_v12/
 105419 usrbinkat   0 Graphic   0%   122MiB  3%    1%   383MiB /opt/microsoft/msedge/msedge --ty
 222369 usrbinkat   0 Graphic   0%    47MiB  1%    0%   282MiB /opt/discord/Discord --type=gpu-p
   4835      root   0 Graphic   0%     4MiB  0%    0%   104MiB /usr/lib/Xorg -nolisten tcp -back
  19946 usrbinkat   0 Graphic   0%     1MiB  0%    0%    89MiB /usr/bin/ksmserver
  20287 usrbinkat   0 Graphic   0%     1MiB  0%   21%   206MiB /usr/bin/albert --platform xcb
  20319 usrbinkat   0 Graphic   0%     1MiB  0%    0%    86MiB /usr/bin/kaccess
 105367 usrbinkat   0 Graphic   0%     1MiB  0%    0%   747MiB /opt/microsoft/msedge/msedge
  19625 usrbinkat   1 Graphic   0%                 0%    76MiB /usr/bin/kwalletd6 --pam-login 12
  19901 usrbinkat   1 Graphic   1%                 2%   106MiB /usr/bin/Xwayland :1 -auth /run/u
  19948 usrbinkat   1 Graphic   0%                 0%   120MiB /usr/bin/kded6
  20004 usrbinkat   1 Graphic   0%                 1%   578MiB /usr/bin/plasmashell --no-respawn
F2Setup  F6Sort  F9Kill  F10Quit  F12Save Config
                                          "usrbinkat@ibane:~" 23:47 30-Sep-24
```

> ## Windows 11 (circa: 2025/01/27)
>
> 
>

deepseek-r1:latest

can you teach me about deepseek? I'd like to learn about the training corpus, response style, best prompt template formats, and best interaction and engagement style for using deepseek as a backend chatbot API service.

Suggested

Show me a code snippet
of a website's sticky header

Tell me a fun fact

```
ollama    | llm_load_tensors:   CPU_Mapped model buffer size =  2572.45 MiB
ollama    | llm_load_tensors:        CUDA0 model buffer size =  1888.00 MiB
ollama    | llama_new_context_with_model: n_seq_max     = 1
ollama    | llama_new_context_with_model: n_ctx         = 2048
ollama    | llama_new_context_with_model: n_ctx_per_seq = 2048
ollama    | llama_new_context_with_model: n_batch       = 512
ollama    | llama_new_context_with_model: n_ubatch      = 512
ollama    | llama_new_context_with_model: flash_attn    = 0
ollama    | llama_new_context_with_model: freq_base     = 10000.0
ollama    | llama_new_context_with_model: freq_scale    = 1
ollama    | llama_new_context_with_model: n_ctx_per_seq (2048) < n_ctx_train (131072) -- the full capac
ity of the model will not be utilized
ollama    | llama_kv_cache_init: kv_size = 2048, offload = 1, type_k = 'f16', type_v = 'f16', n_layer =
 28, can_shift = 1
ollama    | llama_kv_cache_init:        CPU KV buffer size =    56.00 MiB
ollama    | llama_kv_cache_init:      CUDA0 KV buffer size =    56.00 MiB
ollama    | llama_new_context_with_model: KV self size  =  112.00 MiB, K (f16):   56.00 MiB, V (f16):
 56.00 MiB
ollama    | llama_new_context_with_model:        CPU  output buffer size =     0.59 MiB
ollama    | llama_new_context_with_model:      CUDA0 compute buffer size =   730.36 MiB
ollama    | llama_new_context_with_model:  CUDA_Host compute buffer size =    11.01 MiB
ollama    | llama_new_context_with_model: graph nodes  = 986
ollama    | llama_new_context_with_model: graph splits = 200 (with bs=512), 3 (with bs=1)
ollama    | time=2025-01-28T01:08:31.789Z level=INFO source=server.go:594 msg="llama runner started in
1.76 seconds"
```

```
GPU                                      NVIDIA Quadro T1000
CPU      71% 4.04 GHz                    3D            0%   Copy          0%
Memory   34/128 GB (27%)                 Video Encode  0%   Video Decode  0%
Disk 0 (C:)  SSD  0%                     Dedicated GPU memory          4.0 GB
Disk 1  SSD  0%                          Shared GPU memory            63.8 GB
Wi-Fi  Wi-Fi  S: 0 R: 0 Kbps
GPU 0  NVIDIA Quadro T1...  0%
```

Compose file viewer
git:C:\Users\usrbi\Git

Git\docker-compose.yaml

```
services:
  open-webui:
    container_name: open-webui
    image: ghcr.io/open-webui/open-webui:main
    environment:
      - MODEL_DOWNLOAD_DIR=/models
      - OLLAMA_API_BASE_URL=http://ollama:11434
      - OLLAMA_API_URL=http://ollama:11434
      - LOG_LEVEL=debug
    volumes:
      - data:/data
      - models:/models
      - open-webui:/config
    ports:
      - "8080:8080"
    logging:
      driver: json-file
      options:
```

RAM 18.66 GB CPU 36.29% Disk: 16.54 GB used (limit 1006.85 GB) v4.37.1

5:10 PM
1/27/2025

docker-compose.yaml

```
# https://gist.githubusercontent.com/usrbinkat/de44facc683f954bf0cca6c87e2f9f88/raw/04
services:
  open-webui:
    container_name: open-webui
    image: ghcr.io/open-webui/open-webui:main
    environment:
      - MODEL_DOWNLOAD_DIR=/models
      - OLLAMA_API_BASE_URL=http://ollama:11434
      - OLLAMA_API_URL=http://ollama:11434
      - LOG_LEVEL=debug
      - WEBUI_SECRET_KEY=your_secret_key_here  # Add this to prevent logouts after upda
    volumes:
      - data:/data
      - models:/models
      - open-webui:/app/backend/data  # Corrected path based on documentation
    ports:
      - "8080:8080"
    logging:
      driver: json-file
      options:
        max-size: "5m"
        max-file: "2"
    depends_on:
      - ollama
    extra_hosts:
      - "host.docker.internal:host-gateway"
    networks:
      - ollama-net
    restart: unless-stopped

  ollama:
    container_name: ollama
    image: ollama/ollama:latest
    runtime: nvidia
    environment:
      - NVIDIA_VISIBLE_DEVICES=all
      - NVIDIA_DRIVER_CAPABILITIES=compute,utility
      - CUDA_VISIBLE_DEVICES=0
      - LOG_LEVEL=debug
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              capabilities: [gpu]
              count: all
    volumes:
      - ollama:/root/.ollama
      - models:/models
    ports:
      - "11434:11434"
    logging:
      driver: json-file
      options:
        max-size: "5m"
        max-file: "2"
```

```yaml
    networks:
      - ollama-net
    restart: unless-stopped

  watchtower:
    image: containrrr/watchtower
    container_name: watchtower
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    command: --interval 300 open-webui  # Check for updates every 5 minutes
    depends_on:
      - open-webui
    networks:
      - ollama-net
    restart: unless-stopped

volumes:
  data:
  models:
  ollama:
  open-webui:
networks:
  ollama-net:
    driver: bridge
```

pdoyle12 commented on Mar 13

This is fantastic; thank you. This really ought to get a reference from the OpenWebUI docs, since it's a lot less hassle than fiddling with the containers manually.

I'm not certain this was the issue, but I wasn't able to get this to work until I renamed OLLAMA_API_BASE_URL to OLLAMA_BASE_URL. According to the OpenWebUI documentation, OLLAMA_API_BASE_URL is deprecated. Before that, although the containers did start correctly, openwebui didn't seem to respect the Ollama URL.

RockportTigger commented on Jun 16

> This is fantastic; thank you. This really ought to get a reference from the OpenWebUI docs, since it's a lot less hassle than fiddling with the containers manually.

> > > other interesting options an possible addition to this! here -->

Optimize Open WebUI: Three practical extensions for a better user experience

jonndoe47pp commented on Jul 1

Right got it working (nearly) with a few caveats. After installing the nvidia controller toolkit had to:
sudo nvidia-ctk runtime configure --runtime=docker
Last issue, cant get ollama to find any llms. So think the bridge is not working. HELP! I am a newbie. So, read i need to add the
--network=host flag, but thjats in a straight docker statement not docker compose , help total newbie....

ghost commented on Jul 13

> Right got it working (nearly) with a few caveats. After installing the nvidia controller toolkit had to: sudo nvidia-ctk runtime configure --runtime=docker Last issue, cant get ollama to find any llms. So think the bridge is not working. HELP! I am a newbie. So, read i need to add the --network=host flag, but thjats in a straight docker statement not docker compose , help total newbie....

this worked for me...had same issue, then after sudo systemctl restart docker I retried deploying stack and it executed great!

The webUI is not super easy to figure out how to add the models. this link helped me:

https://docs.openwebui.com/getting-started/quick-start/starting-with-ollama

use this link to see whats available. gemma3 qwen3 llama3.1 to name a few. then enter the model name into download model and it will pull it.