Open WebUI integration

- Prerequisites
- Quick start with Docker Compose
- Configuration options
  - Environment variables
  - Using OpenAI-compatible API
- Network configuration
  - Docker Desktop
  - Docker Engine (Linux)
- Using Open WebUI
  - Select a model
  - Pull models through the UI
  - Chat features
- Complete example with multiple models
- Enabling authentication
- Troubleshooting
  - Models don't appear in the drop-down
  - "Connection refused" errors
  - Slow response times
  - CORS errors
- Customization
  - Custom system prompts
  - Model parameters
- Running on a different port
- What's next

Open WebUI is an open-source, self-hosted web interface that provides a ChatGPT-like experience for local AI models. You can connect it to Docker Model Runner to get a polished chat interface for your models.

Prerequisites

- Docker Model Runner enabled with TCP access
- A model pulled (e.g., `docker model pull ai/llama3.2`)

Quick start with Docker Compose

The easiest way to run Open WebUI with Docker Model Runner is using Docker Compose.

Create a `compose.yaml` file:

```
services:
  open-webui:
    image: ghcr.io/open-webui/open-webui:main
    ports:
      - "3000:8080"
    environment:
      - OLLAMA_BASE_URL=http://host.docker.internal:12434
      - WEBUI_AUTH=false
    extra_hosts:
      - "host.docker.internal:host-gateway"
    volumes:
      - open-webui:/app/backend/data

volumes:
  open-webui:
```

Start the services:

```
$ docker compose up -d
```

Open your browser to http://localhost:3000.

Configuration options

Environment variables

Variable	Description	Default
`OLLAMA_BASE_URL`	URL of Docker Model Runner	Required
`WEBUI_AUTH`	Enable authentication	`true`
`OPENAI_API_BASE_URL`	Use OpenAI-compatible API instead	-
`OPENAI_API_KEY`	API key (use any value for DMR)	-

Using OpenAI-compatible API

If you prefer to use the OpenAI-compatible API instead of the Ollama API:

```
services:
  open-webui:
    image: ghcr.io/open-webui/open-webui:main
    ports:
      - "3000:8080"
    environment:
      - OPENAI_API_BASE_URL=http://host.docker.internal:12434/engines/v1
      - OPENAI_API_KEY=not-needed
      - WEBUI_AUTH=false
    extra_hosts:
      - "host.docker.internal:host-gateway"
    volumes:
      - open-webui:/app/backend/data

volumes:
  open-webui:
```

Network configuration

Docker Desktop

On Docker Desktop, `host.docker.internal` automatically resolves to the host machine. The previous example works without modification.

Docker Engine (Linux)

On Docker Engine, you may need to configure the network differently:

```
services:
  open-webui:
    image: ghcr.io/open-webui/open-webui:main
    network_mode: host
    environment:
      - OLLAMA_BASE_URL=http://localhost:12434
      - WEBUI_AUTH=false
    volumes:
      - open-webui:/app/backend/data

volumes:
  open-webui:
```

Or use the host gateway:

```
services:
  open-webui:
    image: ghcr.io/open-webui/open-webui:main
    ports:
      - "3000:8080"
    environment:
      - OLLAMA_BASE_URL=http://172.17.0.1:12434
      - WEBUI_AUTH=false
    volumes:
      - open-webui:/app/backend/data

volumes:
  open-webui:
```

Using Open WebUI

Select a model

1. Open http://localhost:3000
2. Select the model drop-down in the top-left
3. Select from your pulled models (they appear with `ai/` prefix)

Pull models through the UI

Open WebUI can pull models directly:

1. Select the model drop-down
2. Enter a model name: `ai/llama3.2`
3. Select the download icon

Chat features

Open WebUI provides:

- Multi-turn conversations with context
- Message editing and regeneration
- Code syntax highlighting
- Markdown rendering
- Conversation history and search
- Export conversations

Complete example with multiple models

This example sets up Open WebUI with Docker Model Runner and pre-pulls several models:

```
services:
  open-webui:
    image: ghcr.io/open-webui/open-webui:main
    ports:
      - "3000:8080"
    environment:
      - OLLAMA_BASE_URL=http://host.docker.internal:12434
      - WEBUI_AUTH=false
      - DEFAULT_MODELS=ai/llama3.2
```

```
    extra_hosts:
      - "host.docker.internal:host-gateway"
    volumes:
      - open-webui:/app/backend/data
    depends_on:
      model-setup:
        condition: service_completed_successfully

  model-setup:
    image: docker:cli
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    command: >
      sh -c "
        docker model pull ai/llama3.2 &&
        docker model pull ai/qwen2.5-coder &&
        docker model pull ai/smollm2
      "

volumes:
  open-webui:
```

Enabling authentication

For multi-user setups or security, enable authentication:

```
services:
  open-webui:
    image: ghcr.io/open-webui/open-webui:main
    ports:
      - "3000:8080"
    environment:
      - OLLAMA_BASE_URL=http://host.docker.internal:12434
      - WEBUI_AUTH=true
    extra_hosts:
      - "host.docker.internal:host-gateway"
    volumes:
      - open-webui:/app/backend/data
```

```
volumes:
  open-webui:
```

On first visit, you'll create an admin account.

Troubleshooting

Models don't appear in the drop-down

1. Verify Docker Model Runner is accessible:

   ```
   $ curl http://localhost:12434/api/tags
   ```

2. Check that models are pulled:

   ```
   $ docker model list
   ```

3. Verify the `OLLAMA_BASE_URL` is correct and accessible from the container.

"Connection refused" errors

1. Ensure TCP access is enabled for Docker Model Runner.
2. On Docker Desktop, verify `host.docker.internal` resolves:

   ```
   $ docker run --rm alpine ping -c 1 host.docker.internal
   ```

3. On Docker Engine, try using `network_mode: host` or the explicit host IP.

Slow response times

1. First requests load the model into memory, which takes time.
2. Subsequent requests are much faster.
3. If consistently slow, consider:
   - Using a smaller model
   - Reducing context size
   - Checking GPU acceleration is working

CORS errors

If running Open WebUI on a different host:

1. In Docker Desktop, go to Settings > AI
2. Add the Open WebUI URL to **CORS Allowed Origins**

Customization

Custom system prompts

Open WebUI supports setting system prompts per model. Configure these in the UI under Settings > Models.

Model parameters

Adjust model parameters in the chat interface:

1. Select the settings icon next to the model name
2. Adjust temperature, top-p, max tokens, etc.

These settings are passed through to Docker Model Runner.

Running on a different port

To run Open WebUI on a different port:

```
services:
  open-webui:
    image: ghcr.io/open-webui/open-webui:main
    ports:
      - "8080:8080"  # Change first port number
    # ... rest of config
```

What's next

- API reference - Learn about the APIs Open WebUI uses
- Configuration options - Tune model behavior

- IDE integrations - Connect other tools to DMR

□□□ #2
□ □□□ □□□ 2026-10-31 18:54:37 UTC
□ □□□ □□□ 2026-10-31 18:54:38 UTC