

INSTALLATION AND OPERATIONS GUIDE

Local AI Platform on Your Own Server

Open-source language models with a web interface, web search and document search — running entirely on your own hardware.

Version 1.0
October 2026

Platform
Ubuntu 24.04 LTS, Docker

Reference hardware
NVIDIA RTX 4000 SFF Ada, 64 GB
RAM

Contents

0 Overview and prerequisites

1 Install the operating system

2 Secure the server

3 NVIDIA driver and Docker

4 DNS and AI stack

5 Configure Open WebUI

6 Verify, back up, keep up to date

A Troubleshooting

B Acceptance checklist

What you will have

A chat interface with user management runs at `https://ai.your-domain.com`. The language models run on your server's GPU; chats and documents never leave your premises. Only the optional web search sends queries to the internet.

From the outside, only ports 22 (SSH), 80 and 443 are reachable. All other services communicate exclusively within the internal Docker network.

Placeholders in this guide

Placeholder	Replace with
<code>ai.your-domain.com</code>	The (sub)domain under which the platform should be reachable
<code><SERVER-IP></code>	The server's public IPv4 address

All commands are run as `root`, so no `sudo` is needed.

0 Overview

Four containers, one public entry point. Caddy accepts HTTPS requests and forwards them to Open WebUI; everything else is invisible from the outside.

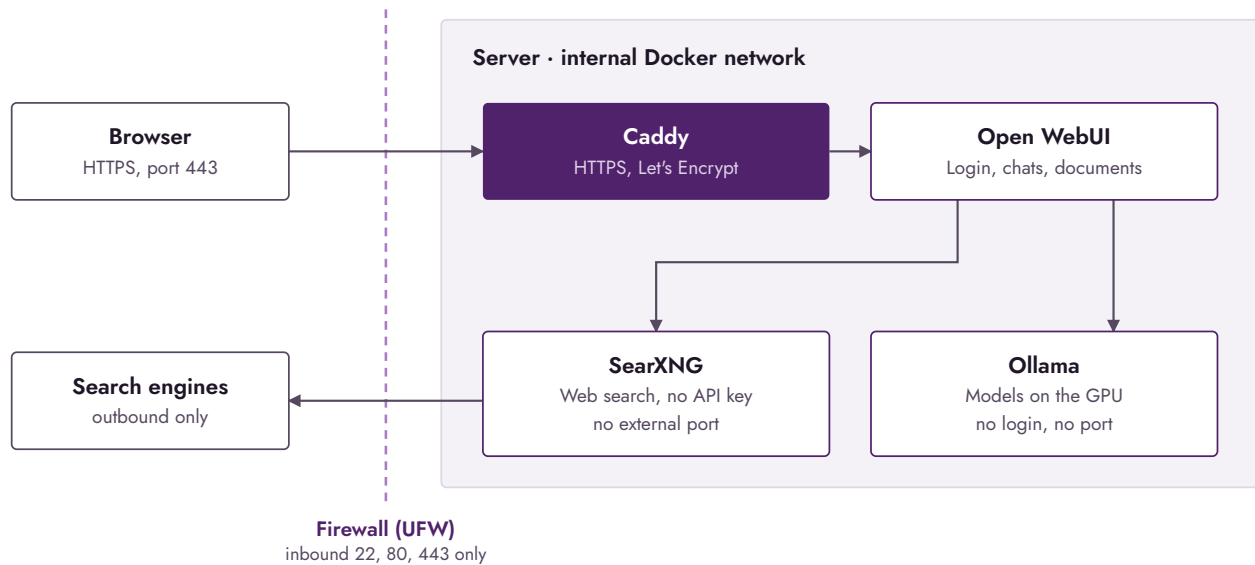


Figure 1 · Architecture: only Caddy is reachable from the internet.

Security concept

Layer	Protects against
Firewall (UFW, 22/80/443 only)	Access to any other port
Key-only SSH, Fail2ban	Password attacks on server access
Internal services without published ports	Direct access to the model API (Ollama has no login)
Caddy with HTTPS and security headers	Interception of passwords and chat content
Registration closed, approval by admin	Strangers creating their own accounts

Prerequisites

- Dedicated server with an NVIDIA GPU (reference: Intel Core i5-13500, 64 GB RAM, 2x NVMe, RTX 4000 SFF Ada with 20 GB VRAM)
- Ubuntu Server 24.04 LTS
- SSH key on the admin workstation
- Access to the domain's DNS settings

1 Install the operating system

Install **Ubuntu Server 24.04 LTS** (64-bit) on the server and add your SSH key for the `root` user during installation.

If two drives are available, we recommend a software RAID 1 so the server keeps running if one SSD fails. Then log in:

```
ssh root@<SERVER-IP>
```

2 Secure the server

Key-only SSH, all ports except 22/80/443 blocked. Complete this before step 4 at the latest – from then on the server is publicly reachable.

2.1 Update and reboot

The reboot ensures step 3 already runs on the latest kernel.

```
apt update && apt full-upgrade -y
reboot
```

2.2 Harden SSH

Root can still log in, but only with a key.

```
cat > /etc/ssh/sshd_config.d/99-hardening.conf <<'EOF'
PermitRootLogin prohibit-password
PasswordAuthentication no
KbdInteractiveAuthentication no
PubkeyAuthentication yes
MaxAuthTries 3
EOF
sshd -t && systemctl restart ssh
```

Important: Keep the current session open and test in a second terminal. `ssh root@<SERVER-IP>` must work, `ssh -o PubkeyAuthentication=no root@<SERVER-IP>` must be refused. Only then close the first session.

2.3 Host firewall (UFW)

```
ufw default deny incoming
ufw default allow outgoing
ufw allow OpenSSH
ufw allow 80/tcp
ufw allow 443/tcp
ufw enable
```

Note: Docker bypasses UFW for published ports. That is why only Caddy publishes ports in step 4; Ollama, SearXNG and Open WebUI get none.

2.4 Fail2ban against password attacks

```
apt install -y fail2ban python3-systemd
cat > /etc/fail2ban/jail.local <<'EOF'
[sshd]
enabled = true
backend = systemd
maxretry = 5
bantime = 1h
EOF
systemctl enable --now fail2ban
fail2ban-client status sshd
```

2.5 Automatic security updates

```
apt install -y unattended-upgrades
dpkg-reconfigure -plow unattended-upgrades # choose "Yes"
```

3 NVIDIA driver and Docker

After this step Docker can use the GPU. Only the server driver is installed – deliberately without the desktop graphics stack.

Do not use: `ubuntu-drivers install`. It also installs X server, `nvidia-prime` and DKMS. In our test the server no longer booted afterwards and had to be reinstalled.

3.1 Install the driver and test without rebooting

```
# 1. Check the kernel and list available drivers
uname -r
ubuntu-drivers list --gpgpu

# 2. Simulate: the list must not contain linux-image, xserver-*, nvidia-prime or nvidia-dkms
apt install -s nvidia-headless-no-dkms-580-server-open nvidia-utils-580-server \
  linux-modules-nvidia-580-server-open-$(uname -r) | grep -E "^Inst" | cut -d' ' -f2

# 3. Install
apt install -y nvidia-headless-no-dkms-580-server-open nvidia-utils-580-server \
  linux-modules-nvidia-580-server-open-$(uname -r)

# 4. Test without rebooting: no new kernel in /boot, GPU visible with approx. 20 GB
ls /boot
modprobe nvidia && nvidia-smi

# 5. Only if step 4 passes: reboot and check again
reboot
nvidia-smi
```

The `udevadm hwdb is deprecated` messages from `ubuntu-drivers` are harmless.

3.2 Prepare the driver for kernel updates

This meta package automatically installs the matching NVIDIA module with every new kernel. The simulation must show only the package itself.

```
apt install -s linux-modules-nvidia-580-server-open-generic | grep -E "^Inst" | cut -d' ' -f2
apt install -y linux-modules-nvidia-580-server-open-generic
```

3.3 Docker

```
curl -fsSL https://get.docker.com | sh
docker --version
```

3.4 NVIDIA Container Toolkit

```
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey \
| gpg --dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg
curl -fsSL https://nvidia.github.io/libnvidia-container/stable/deb/nvidia-container-
toolkit.list \
| sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-
keyring.gpg] https://#g' \
> /etc/apt/sources.list.d/nvidia-container-toolkit.list
apt update && apt install -y nvidia-container-toolkit
nvidia-ctk runtime configure --runtime=docker
systemctl restart docker
```

3.5 Test

```
docker run --rm --gpus all ubuntu nvidia-smi # must show the same table as in 3.1
```

4 DNS and AI stack

Four files in `/opt/ai`, one start command. Only Caddy is reachable from the outside.

4.1 DNS record

Create an A record `ai.your-domain.com` → `<SERVER-IP>`. Only continue once the check returns the correct IP – otherwise the certificate request fails, and Let's Encrypt blocks temporarily after several failed attempts.

```
dig +short ai.your-domain.com
```

4.2 Directory and secrets

```
mkdir -p /opt/ai/searxng && cd /opt/ai
cat > .env <<EOF
DOMAIN=ai.your-domain.com
WEBUI_SECRET_KEY=$(openssl rand -hex 32)
SEARXNG_SECRET=$(openssl rand -hex 32)
EOF
chmod 600 .env
```

4.3 Compose file

```
cat > /opt/ai/docker-compose.yml <<'EOF'
services:
  caddy:
    image: caddy:2
    restart: unless-stopped
    ports: ["80:80", "443:443", "443:443/udp"]
    environment: [DOMAIN=${DOMAIN}]
    volumes:
      - ./Caddyfile:/etc/caddy/Caddyfile:ro
      - caddy_data:/data
      - caddy_config:/config
    depends_on: [open-webui]

  open-webui:
    image: ghcr.io/open-webui/open-webui:main
    restart: unless-stopped
    environment:
      - OLLAMA_BASE_URL=http://ollama:11434
      - WEBUI_URL=https://${DOMAIN}
      - WEBUI_SECRET_KEY=${WEBUI_SECRET_KEY}
      - DEFAULT_USER_ROLE=pending
    volumes: [open-webui:/app/backend/data]
    depends_on: [ollama, searxng]

  ollama:
    image: ollama/ollama:latest
    restart: unless-stopped
    environment:
      - OLLAMA_FLASH_ATTENTION=1
      - OLLAMA_KV_CACHE_TYPE=q8_0
      - OLLAMA_CONTEXT_LENGTH=16384
      - OLLAMA_KEEP_ALIVE=30m
    volumes: [ollama:/root/.ollama]
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: all
              capabilities: [gpu]

  searxng:
    image: searxng/searxng:latest
    restart: unless-stopped
    environment: [SEARXNG_SECRET=${SEARXNG_SECRET}]
    volumes: [./searxng:/etc/searxng]

volumes: {caddy_data: {}, caddy_config: {}, open-webui: {}, ollama: {}}
EOF
```

4.4 Configure Caddy and SearXNG

Caddy obtains the HTTPS certificate automatically. SearXNG returns JSON for Open WebUI and needs no rate limiter, as it is only reachable internally.

```

cat > /opt/ai/Caddyfile <<'EOF'
{$DOMAIN} {
    encode zstd gzip
    header {
        Strict-Transport-Security "max-age=31536000; includeSubDomains"
        X-Content-Type-Options "nosniff"
        X-Frame-Options "SAMEORIGIN"
        Referrer-Policy "strict-origin-when-cross-origin"
        -Server
    }
    reverse_proxy open-webui:8080
}
EOF

cat > /opt/ai/searxng/settings.yml <<'EOF'
use_default_settings: true
server:
    limiter: false
    image_proxy: false
search:
    formats: [html, json]
EOF

```

4.5 Start

```

cd /opt/ai
docker compose up -d
docker compose ps          # 4 containers "Up", only Caddy with 0.0.0.0 ports
docker compose logs -f caddy # wait for "certificate obtained successfully", then Ctrl+C

```

Continue with 5.1 immediately: As long as no admin account exists, anyone with the link could register first and would become administrator.

5 Configure Open WebUI

Secure the admin account first, then close registration, then set up models and web search. All settings in the admin panel apply globally to all users.

5.1 Create the admin account

Open `https://ai.your-domain.com` and create an account immediately. The first account automatically becomes administrator.

5.2 Close registration

Admin Panel → *Settings* → *General*: turn off "New sign-ups" and leave the default role for new users at *pending*. Create further users yourself under *Admin Panel* → *Users* → *+*. Change these values only in the admin panel – after the first start, changes to environment variables no longer take effect.

5.3 Load models

Ollama runs inside the container. An alias makes the command usable on the host from any directory.

```
echo "alias ollama='docker compose -f /opt/ai/docker-compose.yml exec ollama ollama'" >>
~/.bashrc
source ~/.bashrc

ollama pull gpt-oss:20b          # approx. 14 GB - main model
ollama pull qwen3:14b           # approx. 9 GB - background tasks
ollama pull nomic-embed-text    # embeddings for documents
ollama run gpt-oss:20b "Say hello briefly"
ollama ps                      # PROCESSOR must show "100% GPU"
```

`ollama ps` only lists models that are currently loaded. Without a prior request the list is empty.

5.4 Assign and share models

Setting	Value
Admin Panel → Models	<code>gpt-oss:20b</code> as default; disable <code>nomic-embed-text</code> (embedding-only model, does not belong in the chat selection)
Admin Panel → Interface → Task model	<code>qwen3:14b</code> (local and external) — generates chat titles and search queries
Edit model → Visibility	<i>Public</i> for <code>gpt-oss:20b</code> and <code>qwen3:14b</code> ; otherwise new models are visible to the admin only
Admin Panel → Users	Change new accounts from <i>pending</i> to <i>user</i>

Every user can pick a different model in the chat and make it their personal default via the "Set as default" link. This takes precedence over the admin default. The green dot next to a model only means it is currently loaded in GPU memory.

5.5 Web search

Admin Panel → *Settings* → *Web Search*: enable, search engine `searxng`, query URL:

```
http://searxng:8080/search?q=<query>
```

In the chat, the globe icon enables search per request. To enable it for every new chat: *Admin Panel* → *Models* → *Model Defaults* → *Default Features* → *Web Search*.

5.6 System prompt with the current date

The models only know the state of their training data. Without a date they sometimes contradict their own search results in follow-up questions. Under *Models* → *Edit* → *System Prompt*, enter for both chat models:

```
Today is {{CURRENT_DATE}}. Your training knowledge ends considerably earlier and is likely outdated on current topics. For questions about people in office, events, prices or anything that may have changed since 2024, always use search_web, including in follow-up questions. Search results and earlier answers in this chat that are based on a search take precedence over your training knowledge. Cite the source for current facts.
```

5.7 Documents

Admin Panel → *Settings* → *Documents*: embedding engine *Ollama*, model `nomic-embed-text`. This way embeddings also run on the GPU.

Extensible later: sign-in via Microsoft Entra ID (OIDC) instead of separate passwords, and paid cloud models under *Admin Panel* → *Settings* → *Connections* as an OpenAI-compatible API — alongside the local models.

6 Verify, back up, keep up to date

A quick external test, a daily backup and a monthly update keep the platform secure and current.

6.1 Verify from the outside

Run on your own computer, not on the server:

```
nmap -Pn ai.your-domain.com # only 22, 80, 443 "open"
curl -sI https://ai.your-domain.com | head # headers incl. strict-transport-security
curl -m 5 http://ai.your-domain.com:11434 # Ollama: must time out
ssh -o PubkeyAuthentication=no root@ai.your-domain.com # must be refused
```

6.2 Backup

Only the `open-webui` volume with users, chats and settings matters. Models can be downloaded again at any time.

```
cat > /opt/ai/backup.sh <<'EOF'
#!/bin/sh
mkdir -p /opt/backup
docker run --rm -v ai_open-webui:/data:ro -v /opt/backup:/backup alpine \
  tar czf /backup/open-webui-$(date +%F).tar.gz -C /data .
find /opt/backup -name 'open-webui-*.tar.gz' -mtime +14 -delete
EOF
chmod +x /opt/ai/backup.sh
(crontab -l 2>/dev/null; echo "30 3 * * * /opt/ai/backup.sh") | crontab -
/opt/ai/backup.sh && ls -lh /opt/backup # test once manually
```

Recommendation: The backup is stored on the same server and only protects against operator error. For real resilience, also copy the archives to external storage or a NAS.

6.3 Updates

The operating system updates itself (2.5), and the NVIDIA module follows new kernels (3.2). Update the containers roughly once a month:

```
cd /opt/ai && docker compose pull && docker compose up -d && docker image prune -f
```

After kernel updates, occasionally `reboot` and then check `nvidia-smi`.

A Troubleshooting

These issues actually occurred during the reference installation.

Symptom	Cause and solution
Server no longer boots after driver installation	<code>ubuntu-drivers install</code> installed the desktop stack. Reinstall (step 1) and set up the driver as described in 3.1.
<code>docker compose up</code> : timeout at <code>ghcr.io</code>	Usually a brief outage of the GitHub registry: repeat <code>docker compose pull open-webui</code> . If it persists, check with <code>curl -4 -m 10 https://ghcr.io/v2/</code> (response <code>401</code> = connection OK) and check any upstream firewall for blocked reply packets.
<code>ollama: command not found</code>	Ollama only runs in the container. Create the alias from 5.3 or use <code>docker compose exec ollama ollama ...</code> .
<code>ollama ps</code> is empty	Not an error — no model is currently loaded. Run <code>ollama run</code> or send a chat request once, then check again.
"nomic-embed-text does not support chat"	The embedding model is selected as chat or task model. Apply the settings from 5.4.
Other users see no models	Models are private by default. Set visibility to <i>Public</i> (5.4).
New users stuck on the waiting screen	Role <i>pending</i> . Change to <i>user</i> under <i>Admin Panel</i> → <i>Users</i> .
Outdated answers despite web search	Check whether "Explored search_web" appears above the answer. If not, the model did not search — set the system prompt from 5.6 and use <code>gpt-oss:20b</code> , which calls search tools more reliably.
Web search returns nothing	Test SearXNG directly: <code>docker compose exec open-webui curl -s "http://searxng:8080/search?q=test&format=json" head -c 500</code> If <code>results</code> is empty, enable further engines (e.g. DuckDuckGo, Brave) in <code>settings.yml</code> .
<code>systemctl enable nvidia-persistenced</code> reports "no installation config"	Not an error. The service starts automatically; persistence mode is not needed for this setup.

B Acceptance checklist

Tick off every item before handing the platform over to users.

- | | |
|---|---|
| ☐ Ubuntu 24.04 LTS installed and up to date | ☐ Admin account created, registration closed |
| ☐ SSH login without key refused | ☐ <code>ollama ps</code> shows 100% GPU |
| ☐ UFW active, Fail2ban running | ☐ Embedding model hidden, task model set |
| ☐ <code>nvidia-smi</code> works after reboot | ☐ Chat models set to <i>Public</i> |
| ☐ NVIDIA meta package installed | ☐ System prompt with date set |
| ☐ nmap shows only 22, 80, 443 | ☐ Web search returns sources |
| ☐ HTTPS certificate valid | ☐ Backup cron job running, first archive present |

Moregate Software

Tools for teams that would rather keep their data in their own house.

moregate-software.com