VPS

My $5/mo VPS Docker Stack: Full docker-compose.yml + nginx Config

You DM'd "VPS". Here's the exact docker-compose.yml and nginx config I run on a Hetzner CX22 to host Open WebUI, n8n, Dify, RAGFlow, and Flowise. Everything copy-pasteable, tested on Ubuntu 22.04 LTS as of 2026-04-27. Setup takes 45 minutes start to finish.

From the post: 5 free AI tools running on my $5/mo VPS right now

What you're getting

- `docker-compose.yml` for Open WebUI + n8n + Flowise (the light trio, ~1.3GB RAM combined)
- `dify-compose.yml` for Dify (separate stack, ~1.8GB, needs its own bring-up)
- `.env.template` with every required variable named and described
- `nginx/ai-stack.conf` with SSL-ready reverse proxy blocks for all 5 subdomains
- Certbot one-liner for free HTTPS

- Memory budget breakdown so you know what fits on 4GB
- Troubleshooting for the 4 failures I hit on first deploy

Quick start (5 minutes)

1. Provision a Hetzner CX22 (4 vCPU, 4GB RAM, 40GB NVMe). Choose Ubuntu 22.04 LTS. Frankfurt or Ashburn. Monthly cost: EUR 3.79.

2. SSH in and install Docker:

```bash
curl -fsSL https://get.docker.com | sh
sudo usermod -aG docker $USER
newgrp docker
docker --version
# Docker version 26.x.x
```

3. Install nginx + Certbot:

```bash
sudo apt install -y nginx certbot python3-certbot-nginx
```

4. Point 5 DNS A-records at your VPS IP before running Certbot. Example structure:

```
chat.yourdomain.com   → VPS_IP
n8n.yourdomain.com    → VPS_IP
flowise.yourdomain.com → VPS_IP
dify.yourdomain.com   → VPS_IP
rag.yourdomain.com    → VPS_IP
```

5. Clone the stack:

```bash
mkdir ~/ai-stack && cd ~/ai-stack
# paste docker-compose.yml and .env from below, then:
cp .env.template .env && nano .env
docker compose up -d
# Expected: 3 containers starting, open-webui healthy in ~40s
```

The actual things

docker-compose.yml (Open WebUI + n8n + Flowise)

Light trio. Runs comfortably inside 1.5GB. Leave 2.5GB headroom for Dify or RAGFlow.

```yaml
# ~/ai-stack/docker-compose.yml
# Hetzner CX22 | Ubuntu 22.04 | Tested 2026-04-27

services:

  # 01. Open WebUI — self-hosted ChatGPT clone (134K stars)
  open-webui:
    image: ghcr.io/open-webui/open-webui:main
    container_name: open-webui
    restart: unless-stopped
    ports:
      - "127.0.0.1:3001:8080"
    volumes:
      - open-webui-data:/app/backend/data
    environment:
      WEBUI_SECRET_KEY: ${WEBUI_SECRET_KEY}
      OPENAI_API_KEY: ${OPENAI_API_KEY}
      ANTHROPIC_API_KEY: ${ANTHROPIC_API_KEY}
      ENABLE_SIGNUP: "false"
    mem_limit: 512m
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080/health"]
      interval: 30s
      timeout: 10s
      retries: 3
```

```
  # 02. n8n — drag-drop workflow automation (186K stars)
  n8n:
    image: docker.n8n.io/n8nio/n8n:latest
    container_name: n8n
    restart: unless-stopped
    ports:
      - "127.0.0.1:5678:5678"
    volumes:
      - n8n-data:/home/node/.n8n
    environment:
      N8N_BASIC_AUTH_ACTIVE: "true"
      N8N_BASIC_AUTH_USER: ${N8N_USER}
      N8N_BASIC_AUTH_PASSWORD: ${N8N_PASSWORD}
      N8N_HOST: ${N8N_DOMAIN}
      N8N_PORT: "5678"
      N8N_PROTOCOL: "https"
      WEBHOOK_URL: "https://${N8N_DOMAIN}/"
      EXECUTIONS_DATA_PRUNE: "true"
      EXECUTIONS_DATA_MAX_AGE: "336"
    mem_limit: 512m

  # 03. Flowise — drag-drop LLM agent builder (32K stars)
  flowise:
    image: flowiseai/flowise:latest
    container_name: flowise
    restart: unless-stopped
    ports:
      - "127.0.0.1:3010:3000"
    volumes:
      - flowise-data:/root/.flowise
    environment:
      FLOWISE_USERNAME: ${FLOWISE_USER}
      FLOWISE_PASSWORD: ${FLOWISE_PASSWORD}
      PORT: "3000"
    mem_limit: 384m

volumes:
  open-webui-data:
  n8n-data:
  flowise-data:
```

Ports are bound to `127.0.0.1` only. Nothing is exposed to the internet directly. nginx proxies from 443.

dify-compose.yml (Dify standalone stack)

Dify ships its own compose. Clone it separately so its postgres/redis don't collide with anything.

```bash
# Run this once to set up Dify
git clone https://github.com/langgenius/dify.git ~/dify
cd ~/dify/docker
cp .env.example .env
# Edit .env: set SECRET_KEY, INIT_PASSWORD, NGINX_HTTPS_ENABLED=false
# (nginx handled externally — disable their bundled nginx)
nano .env
docker compose -f docker-compose.yaml up -d
# Expected: 9 containers starting. API healthy in ~90s.
# Access internally on: http://127.0.0.1:80 (their default)
# Remap Dify's internal port to avoid clash with your nginx:
```

Add this override to `~/dify/docker/docker-compose.override.yaml`:

```yaml
services:
  nginx:
    ports:
      - "127.0.0.1:8082:80"
```

Then: `docker compose -f docker-compose.yaml -f docker-compose.override.yaml up -d`

Dify's web UI is now at `127.0.0.1:8082`. Your external nginx proxies `dify.yourdomain.com` to that.

RAGFlow note: RAGFlow pulls Elasticsearch (1.5GB alone) plus MinIO plus Redis. On 4GB total, RAGFlow + anything else = OOM. Run it on a CX32 (8GB, EUR 6.80/mo) or spin a separate CX22 just for it. Worth it if you have large document sets.

.env.template

```bash
# ~/ai-stack/.env.template — copy to .env and fill in

# Open WebUI
WEBUI_SECRET_KEY=change_me_32_chars_minimum_random
OPENAI_API_KEY=sk-...           # optional, leave blank to use local mo
ANTHROPIC_API_KEY=sk-ant-...    # optional

# n8n
N8N_USER=admin
N8N_PASSWORD=change_me_strong_password
N8N_DOMAIN=n8n.yourdomain.com

# Flowise
FLOWISE_USER=admin
FLOWISE_PASSWORD=change_me_strong_password
```

nginx/ai-stack.conf

One file handles all 5 subdomains. Paste to `/etc/nginx/sites-available/ai-stack`, symlink to `sites-enabled`, run Certbot.

```nginx
# /etc/nginx/sites-available/ai-stack
# Replace yourdomain.com throughout

# ---- Open WebUI ----
server {
    listen 80;
    server_name chat.yourdomain.com;
    return 301 https://$host$request_uri;
}
server {
    listen 443 ssl;
    server_name chat.yourdomain.com;
    ssl_certificate     /etc/letsencrypt/live/yourdomain.com/fullchain.
    ssl_certificate_key /etc/letsencrypt/live/yourdomain.com/privkey.pe
    client_max_body_size 20M;
    location / {
        proxy_pass http://127.0.0.1:3001;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_read_timeout 300;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}

# ---- n8n ----
server {
    listen 80;
    server_name n8n.yourdomain.com;
    return 301 https://$host$request_uri;
}
server {
    listen 443 ssl;
    server_name n8n.yourdomain.com;
    ssl_certificate     /etc/letsencrypt/live/yourdomain.com/fullchain.
    ssl_certificate_key /etc/letsencrypt/live/yourdomain.com/privkey.pe
    client_max_body_size 16M;
    location / {
        proxy_pass http://127.0.0.1:5678;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_read_timeout 300;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}

# ---- Flowise ----
server {
    listen 80;
    server_name flowise.yourdomain.com;
    return 301 https://$host$request_uri;
}
server {
    listen 443 ssl;
    server_name flowise.yourdomain.com;
    ssl_certificate     /etc/letsencrypt/live/yourdomain.com/fullchain.
    ssl_certificate_key /etc/letsencrypt/live/yourdomain.com/privkey.pe
    location / {
```

```
        proxy_pass http://127.0.0.1:3010;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_read_timeout 120;
    }
}

# ---- Dify ----
server {
    listen 80;
    server_name dify.yourdomain.com;
    return 301 https://$host$request_uri;
}
server {
    listen 443 ssl;
    server_name dify.yourdomain.com;
    ssl_certificate     /etc/letsencrypt/live/yourdomain.com/fullchain.
    ssl_certificate_key /etc/letsencrypt/live/yourdomain.com/privkey.pe
    client_max_body_size 15M;
    location / {
        proxy_pass http://127.0.0.1:8082;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_read_timeout 300;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}
```

Enable and get SSL:

```bash
sudo ln -s /etc/nginx/sites-available/ai-stack /etc/nginx/sites-enabled
sudo nginx -t
# expected: syntax is ok / test is successful
sudo certbot --nginx -d chat.yourdomain.com -d n8n.yourdomain.com -d fl
sudo systemctl reload nginx
```

Memory budget (4GB VPS)

```
Open WebUI    ~280MB resident
n8n           ~220MB resident
Flowise       ~180MB resident
Dify (all 9)  ~1,600MB resident
OS + nginx    ~400MB
Total         ~2,680MB of 4,096MB
Headroom      ~1,400MB (swap handles spikes)
```

RAGFlow adds ~2.2GB. Do not run RAGFlow alongside Dify on 4GB.

Real example

First boot sequence I ran on 2026-04-27:

BASH

```
cd ~/ai-stack
docker compose up -d
# Creating network "ai-stack_default"
# Creating open-webui ... done
# Creating n8n          ... done
# Creating flowise      ... done

docker ps
# NAMES        STATUS        PORTS
# open-webui   Up 43 seconds  127.0.0.1:3001→8080/tcp
# n8n          Up 42 seconds  127.0.0.1:5678→5678/tcp
# flowise      Up 42 seconds  127.0.0.1:3010→3000/tcp

curl -s http://127.0.0.1:3001/health
# {"status":"ok"}
```

Then opened `chat.yourdomain.com`, added Anthropic API key in Settings, sent a test message. Response from Claude 3.5 Sonnet in 2.1s. Memory at that point: `free -m` showed 1,840MB used out of 4,096MB.

n8n: loaded `n8n.yourdomain.com`, created a test webhook workflow, triggered it via curl. Got 200 in 340ms. Credentials stored encrypted in the n8n-data volume.

Flowise: loaded `flowise.yourdomain.com`, dropped in a basic ChatAnthropic node, connected to a Conversation Chain, tested via the built-in chat window. Working in under 4 minutes.

Troubleshooting

open-webui keeps restarting. `docker logs open-webui` shows "database is locked".
Volume permission issue. Fix:

```bash
docker compose down
sudo chown -R 1000:1000 $(docker volume inspect ai-stack_open-webui-dat
docker compose up -d
```

n8n shows "cannot GET /" after SSL setup.
The `WEBHOOK_URL` env var must include trailing slash and use https. Check your `.env`:

```bash
N8N_DOMAIN=n8n.yourdomain.com
WEBHOOK_URL=https://n8n.yourdomain.com/
```

Then `docker compose up -d --force-recreate n8n`.

Certbot says "could not find parameter" or fails on multiple -d flags.
DNS must propagate before Certbot runs. Check all A-records resolve:

```bash
for sub in chat n8n flowise dify; do echo "$sub: $(dig +short $sub.your
```

All must return your VPS IP. Wait and retry if any are blank.

Dify API container restarts with "relation does not exist".
Dify runs migrations on first start. Postgres needs ~30s before API comes up. Fix:

```bash
cd ~/dify/docker
docker compose restart api worker
```

OOM kill — one container disappears, `dmesg | tail -20` shows "oom-killer".
You're over 4GB. Check:

```bash
docker stats --no-stream
```

Kill the heaviest non-essential container or add 2GB swap:

```bash
sudo fallocate -l 2G /swapfile && sudo chmod 600 /swapfile
sudo mkswap /swapfile && sudo swapon /swapfile
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

Step it up

Automatic container restarts on VPS reboot. All services have `restart: unless-stopped` already. Confirm Docker itself starts on boot:

```bash
sudo systemctl enable docker
```

Watchtower for automatic image updates. Pulls new images and restarts containers weekly. Add to your compose:

```yaml
watchtower:
  image: containrrr/watchtower
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
  command: --schedule "0 0 3 * * 0" --cleanup
  restart: unless-stopped
  mem_limit: 64m
```

Runs Sunday 3am. Cleans old images. Zero manual maintenance.

Uptime monitoring with Uptime Kuma. Self-hosted status page. 1 more container, 100MB:

```yaml
uptime-kuma:
  image: louislam/uptime-kuma:1
  container_name: uptime-kuma
  ports:
    - "127.0.0.1:3002:3001"
  volumes:
    - uptime-kuma-data:/app/data
  restart: unless-stopped
  mem_limit: 128m
```

Add a subdomain block to nginx. Monitor all 5 services. Get Telegram alerts when anything goes down.

What's next

Next magnet: "PROMPTS" (my 7 production n8n workflows that connect Claude to this stack). DM "PROMPTS" on the post to get them.

Newsletter at elvison.com/newsletter covers new tools, stack updates, and what I'm actually building each week.

--- Roelof @ Elvison