AI & Automation

Chapter Introduction

Overview

This chapter documents the AI inference, voice processing, and workflow automation services running on Centerpoint. All GPU-accelerated workloads leverage an NVIDIA GeForce RTX 5080 connected to the Mini PC via OcuLink, providing 16 GB of GDDR7 VRAM on the Blackwell architecture (Compute Capability 12.0).

GPU Hardware

Property	Value
GPU	NVIDIA GeForce RTX 5080
VRAM	16 GB GDDR7 (16,303 MiB)
Architecture	Blackwell (Compute Capability 12.0)
Connection	OcuLink (external GPU enclosure)
NVIDIA Driver	580.159.03
CUDA Version	12.9
Container Runtime	`nvidia` (all GPU containers)

All containers that use the GPU are launched with `runtime: nvidia` and `NVIDIA_VISIBLE_DEVICES=all`. No device passthrough via `--device /dev/dri` is used — the Intel Arc / IPEX-LLM path has been retired.

Services in This Chapter

Service	Container(s)	GPU	Purpose
Ollama	`ollama`	Yes	Local LLM inference backend
Open Web UI	`open-webui`	No	Chat interface for Ollama and OpenAI APIs

Service	Container(s)	GPU	Purpose
Faster-Whisper	`faster-whisper`	Yes	Speech-to-text (Whisper large-v3-turbo)
Kokoro	`kokoro`	Yes	Text-to-speech (TTS) API
Riffado	`riffado`, `riffado-db`	No	AI audio podcast app (formerly OpenPlaud)
PaperlessAI	`paperless-ai`	No	Autonomous document classification via Ollama
N8N	`n8n`	No	Workflow automation platform
Open Notebook	`open-notebook`, `open-notebook-db`	No	AI research notebook (SurrealDB backend)
MCP GitHub	`mcp-github-proxy`, `github-mcp-server`	No	GitHub MCP server with OAuth 2.1

Compose Project

Most AI stack services (Ollama, Open Web UI, Kokoro, Faster-Whisper, Open Notebook, Riffado) are managed as a single Portainer compose project called **`ai-stack`** on a shared `ai-internal` bridge network plus `traefik-net`. N8N, PaperlessAI, and MCP GitHub are separate Portainer stacks.

Last Updated: 2026-06-16

Ollama

Overview

Ollama is the local large language model (LLM) inference backend for the homelab. It serves models via an OpenAI-compatible REST API and is consumed by Open Web UI, PaperlessAI, and any other service that needs LLM inference without sending data to external providers.

Ollama runs with full NVIDIA RTX 5080 acceleration via the `nvidia` container runtime. All model weights are stored on the local NVMe system drive.

Access

Type	URL	Notes
Internal	`https://ollama.home.local`	Traefik-proxied HTTPS
Direct	`http://192.168.1.85:11434`	Raw API (no TLS)

No external (internet-facing) route — LAN and Tailscale access only.

Configuration

Image: `ollama/ollama:latest` **Compose project:** `ai-stack` **Runtime:** `nvidia`

Ports

Port	Protocol	Purpose
`11434`	TCP	Ollama REST API (host-bound)

Traefik Labels

```
traefik.enable: "true"
traefik.docker.network: traefik-net
traefik.http.routers.ollama.rule: Host(`ollama.home.local`)
traefik.http.routers.ollama.entrypoints: websecure
traefik.http.routers.ollama.tls.certresolver: step-ca
traefik.http.services.ollama.loadbalancer.server.port: 11434
```

Internal-only route, no authentication middleware — API access is unrestricted on the LAN. Callers must be on the LAN or Tailscale.

Environment Variables

Variable	Value	Purpose
`OLLAMA_HOST`	`0.0.0.0`	Listen on all interfaces
`NVIDIA_VISIBLE_DEVICES`	`all`	Expose all NVIDIA GPUs to container
`NVIDIA_DRIVER_CAPABILITIES`	`compute,utility`	Required NVIDIA driver caps
`OLLAMA_NUM_GPU`	`999`	Use all available GPU layers
`no_proxy`	`localhost,127.0.0.1`	Bypass proxy for local calls

GPU Acceleration

Ollama uses the `nvidia` container runtime. The RTX 5080 provides 16 GB of VRAM, allowing large models (7B–27B parameter range) to run fully in VRAM without CPU offloading.

```
runtime: nvidia
environment:
  - NVIDIA_VISIBLE_DEVICES=all
  - NVIDIA_DRIVER_CAPABILITIES=compute,utility
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/ollama/`	`/root/.ollama`	Model weights and config

Model files are stored under `/home/jeeves/docker/ollama/models/` on the local NVMe system drive (1.8 TB). Large models can consume significant space.

Networks

Network	Purpose
`ai-stack_ai-internal`	Internal communication with Open Web UI, PaperlessAI
`traefik-net`	Exposes Ollama API via Traefik

Dependencies

- NVIDIA container runtime (`nvidia`) on the Docker daemon
- RTX 5080 connected via OcuLink (must be recognised as a CUDA device)

Notes / Gotchas

- `OLLAMA_NUM_GPU=999` is the conventional way to tell Ollama to use as many GPU layers as possible. It does not literally use 999 GPUs.
- Models are downloaded via `ollama pull <model>` or via the Open Web UI admin panel. Downloaded models persist in the bind-mounted `/root/.ollama` directory.
- If the OcuLink connection drops or the GPU is not recognised, Ollama falls back to CPU inference — responses will be significantly slower. Check with:

  ```
  docker exec -it ollama ollama ps
  ```

- The local file at `/home/jeeves/docker/ai-stack/ollama-intel-arc/docker-compose.yml` is the legacy IPEX-LLM config and is **no longer in use**. The current stack is managed via Portainer and uses `ollama/ollama:latest` with CUDA.

Last Updated: 2026-06-16

Open-WebUI

Overview

Open Web UI is the primary chat and AI management interface for the homelab. It provides a ChatGPT-style web frontend connected to the local Ollama backend, with support for conversation history, model selection, RAG (document chat), image generation, and tool use. It can also proxy to external OpenAI-compatible APIs.

Access

Type	URL	Notes
Internal	`https://ai.home.local`	LAN access via Step-CA TLS
External	`https://ai.jeeves5454.ddns.net`	Internet-facing — Authentik SSO + GeoBlock + CrowdSec

Configuration

Image: `ghcr.io/open-webui/open-webui:main` **Compose project:** `ai-stack`

Ports

Port	Protocol	Purpose
`3015`	TCP	Web UI (mapped from internal `8080`)

Traefik Labels

```
# Internal route
traefik.http.routers.openwebui-internal.rule: Host(`ai.home.local`)
traefik.http.routers.openwebui-internal.entrypoints: websecure
```

```
traefik.http.routers.openwebui-internal.tls.certresolver: step-ca
traefik.http.routers.openwebui-internal.service: openwebui-svc

# External route
traefik.http.routers.openwebui-external.rule: Host(`ai.jeeves5454.ddns.net`)
traefik.http.routers.openwebui-external.entrypoints: websecure
traefik.http.routers.openwebui-external.tls.certresolver: letsencrypt
traefik.http.routers.openwebui-external.middlewares: authentik-auth@docker,plex-
geoblock@file,crowdsec-bouncer@file
traefik.http.routers.openwebui-external.service: openwebui-svc

traefik.http.services.openwebui-svc.loadbalancer.server.port: 8080
```

Environment Variables

Variable	Value / Notes
`WEBUI_AUTH`	`False` — authentication handled by Authentik
`ENABLE_OLLAMA_API`	`True`
`ENABLE_OPENAI_API`	`True`
`ENABLE_IMAGE_GENERATION`	`True`
`IMAGE_GENERATION_ENGINE`	`automatic1111`
`IMAGE_GENERATION_MODEL`	`dreamshaper_8`
`IMAGE_SIZE`	`400x400`
`IMAGE_STEPS`	`8`
`AUTOMATIC1111_BASE_URL`	`http://stable-diffusion:7860/`
`AUTOMATIC1111_CFG_SCALE`	`2`
`AUTOMATIC1111_SAMPLER`	`DPM++ SDE`
`AUTOMATIC1111_SCHEDULER`	`Karras`

> “ `WEBUI_AUTH=False` disables Open Web UI's own login page. Authentication is delegated entirely to Authentik ForwardAuth on the external route. On the internal LAN route, the interface is open — access is controlled by network boundary only.

Volumes / Bind Mounts

Host Path / Volume	Container Path	Purpose
`open_webui_open-webui-data`	`/app/backend/data`	Conversation history, settings, uploaded docs (named volume, external)

Networks

Network	Purpose
`ai-stack_ai-internal`	Reaches Ollama backend on `ai-internal` network
`traefik-net`	Exposes the web UI via Traefik

Dependencies

- `ollama` — must be running for model inference; Open Web UI will start without it but model requests will fail
- Authentik — required for external route SSO; LAN route is unaffected if Authentik is down

Notes / Gotchas

- The `main` image tag tracks the latest development build. For stability, consider pinning to a tagged release (e.g. `v0.6.x`).
- `WEBUI_AUTH=False` means **anyone on the LAN** can access the internal URL without credentials. If untrusted devices are on the LAN, consider enabling WEBUI_AUTH and creating user accounts, or adding Authentik middleware to the internal route as well.
- Conversation history and user settings are stored in the named Docker volume. Back this up before upgrades.
- Open Web UI admin panel is at `https://ai.home.local/admin/` — first user to register (if auth is enabled) becomes the admin.
- Image generation requires the stable-diffusion container to be running (separate service, also on `ai-internal` network).

Last Updated: 2026-06-16

Faster Whisper

Overview

Faster-Whisper is the speech-to-text transcription service for the homelab. It runs OpenAI's Whisper model via the `faster-whisper` library (CTranslate2 backend), which offers significantly faster inference than the original Whisper implementation at equivalent or lower VRAM usage.

It is used by N8N automation workflows for audio transcription tasks (e.g. podcast processing in the Minuspod pipeline).

Access

Type	URL	Notes
Internal	`https://whisper.home.local`	LAN access via Step-CA TLS

No external route — internal automation use only.

Configuration

Image: `hwdsl2/whisper-server:cuda` **Compose project:** `ai-stack` **Runtime:** `nvidia` **CUDA Version:** 12.9

Ports

Port	Protocol	Purpose
`9015`	TCP	HTTP API (mapped from internal `9000`)

Traefik Labels

```
traefik.enable: "true"
traefik.docker.network: traefik-net
traefik.http.routers.whisper.rule: Host(`whisper.home.local`)
traefik.http.routers.whisper.entrypoints: websecure
traefik.http.routers.whisper.tls.certresolver: step-ca
traefik.http.services.whisper.loadbalancer.server.port: 9000
```

Environment Variables

Variable	Value	Purpose
`WHISPER_MODEL`	`large-v3-turbo`	Whisper model variant to load
`WHISPER_DEVICE`	`cuda`	Run inference on GPU
`WHISPER_COMPUTE_TYPE`	`float16`	FP16 precision (optimal for CUDA)
`WHISPER_LANGUAGE`	`en`	Default transcription language
`NVIDIA_VISIBLE_DEVICES`	`all`	Expose all NVIDIA GPUs
`NVIDIA_DRIVER_CAPABILITIES`	`compute,utility`	Required NVIDIA driver capabilities

Model: `large-v3-turbo` — the distilled variant of Whisper large-v3, offering near-large accuracy at roughly 3× the speed and reduced VRAM usage.

GPU Acceleration

```
runtime: nvidia
environment:
  - NVIDIA_VISIBLE_DEVICES=all
  - WHISPER_DEVICE=cuda
  - WHISPER_COMPUTE_TYPE=float16
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/ai-stack/whisper/files`	`/var/lib/whisper`	Transcription input/output files

Model weights are downloaded to a temp directory inside the container on first start and cached within the container layer (not persisted in a named volume).

Networks

Network	Purpose
`ai-stack_ai-internal`	Internal access from N8N workflows
`traefik-net`	Exposes API via Traefik

API Usage

The whisper-server exposes a simple HTTP POST endpoint:

```
curl -X POST https://whisper.home.local/inference \
  -F file=@audio.mp3 \
  -F response_format=json
```

Response:

```
{"text": "Transcribed text here..."}
```

Dependencies

- NVIDIA container runtime with RTX 5080 access
- N8N (primary consumer via HTTP calls)

Notes / Gotchas

- The `large-v3-turbo` model is downloaded on first container start. This can take several minutes and the container will appear unresponsive until the download completes.
- `float16` compute type requires a GPU with FP16 support. The RTX 5080 (Blackwell) supports this natively. On CPU-only fallback, use `int8` instead.
- The `files` bind mount (`/var/lib/whisper`) can be used to pre-stage audio files for batch transcription if needed.

Last Updated: 2026-06-16

Kokoro

Overview

Kokoro is a high-quality, locally-hosted text-to-speech (TTS) service. It exposes an OpenAI-compatible TTS API, making it a drop-in replacement for external TTS services in any application that supports the `/v1/audio/speech` endpoint.

It runs the Kokoro TTS model via the `kokoro-fastapi` server with full RTX 5080 GPU acceleration.

Access

Type	URL	Notes
Internal	`https://kokoro.home.local`	LAN access via Step-CA TLS

No external route — internal service only.

Configuration

Image: `ghcr.io/remsky/kokoro-fastapi-gpu:latest-cu128` **Compose project:** `ai-stack` **Runtime:** `nvidia` **CUDA Version:** 12.8 (image) / 12.9 (host driver, backward compatible)

Ports

Port	Protocol	Purpose
`8880`	TCP	Kokoro FastAPI TTS API

Traefik Labels

```
traefik.enable: "true"
traefik.docker.network: traefik-net
```

```
traefik.http.routers.kokoro.rule: Host(`kokoro.home.local`)
traefik.http.routers.kokoro.entrypoints: websecure
traefik.http.routers.kokoro.tls.certresolver: step-ca
traefik.http.services.kokoro.loadbalancer.server.port: 8880
```

GPU Acceleration

```
runtime: nvidia
environment:
  - NVIDIA_VISIBLE_DEVICES=all
  - DEVICE=gpu
  - USE_GPU=true
```

The image tag `latest-cu128` targets CUDA 12.8. The host driver (580.159.03) is fully forward-compatible with this image.

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/kokoro`	`/app/data`	Voice models and cached data

Networks

Network	Purpose
`ai-stack_ai-internal`	Internal access from Open Web UI and N8N
`traefik-net`	Exposes API via Traefik

API Usage

Kokoro exposes an OpenAI-compatible TTS endpoint:

```
curl -X POST https://kokoro.home.local/v1/audio/speech \
  -H "Content-Type: application/json" \
```

```
  -d '{
    "model": "kokoro",
    "input": "Hello from the homelab.",
    "voice": "af_sky",
    "response_format": "mp3"
  }' --output speech.mp3
```

Available voices and model details are listed at `https://kokoro.home.local/docs` (Swagger UI).

Dependencies

- NVIDIA container runtime with RTX 5080 access

Notes / Gotchas

- The `latest-cu128` tag pulls the latest build compiled against CUDA 12.8. NVIDIA driver 580.x supports CUDA 12.9 on the host, which is backward-compatible.
- Voice model files are cached in the `/app/data` bind mount on first use — initial synthesis requests may be slower while models are downloaded.
- Kokoro is OpenAI API-compatible, meaning Open Web UI can be configured to use it as its TTS provider via the admin settings.

Last Updated: 2026-06-16

Riffado (OpenPlaud)

Overview

Riffado (formerly known as OpenPlaud) is a self-hosted AI audio and podcast management application. It handles audio file storage, playback, and AI-assisted processing within the homelab ecosystem.

Audio files are stored on the Ceph OSD volume (`1TB_Vol1`) rather than the system NVMe, keeping large media off the primary drive.

Access

Type	URL	Notes
Internal	`https://riffado.home.local`	LAN access via Step-CA TLS
External	`https://riffado.jeeves5454.ddns.net`	Internet-facing — Authentik SSO + GeoBlock + CrowdSec

Containers in This Stack

Container	Image	Role
`riffado`	`ghcr.io/riffado/riffado:latest`	Application server
`riffado-db`	`postgres:16-alpine`	PostgreSQL database backend

Configuration

Compose project: `ai-stack`

Environment Variables

Variable	Value / Notes
`APP_URL`	`https://riffado.jeeves5454.ddns.net`
`HOSTNAME`	`0.0.0.0`
`DATABASE_URL`	`postgresql://postgres:**REDACTED**@riffado-db:5432/riffado`
`DEFAULT_STORAGE_TYPE`	`local`
`LOCAL_STORAGE_PATH`	`/app/storage`
`DISABLE_REGISTRATION`	`true`
`BETTER_AUTH_SECRET`	**REDACTED**
`ENCRYPTION_KEY`	**REDACTED**
`NODE_ENV`	`production`

`DISABLE_REGISTRATION=true` prevents new accounts from being created — access is limited to pre-provisioned users and gated by Authentik on the external route.

Traefik Labels

```
# External route
traefik.http.routers.riffado-ext.rule: Host(`riffado.jeeves5454.ddns.net`)
traefik.http.routers.riffado-ext.entrypoints: websecure
traefik.http.routers.riffado-ext.tls.certresolver: letsencrypt
traefik.http.routers.riffado-ext.middlewares: authentik-auth@docker,plex-geoblock@file,crowdsec-bouncer@file
traefik.http.routers.riffado-ext.service: riffado-svc

# Internal route
traefik.http.routers.riffado-int.rule: Host(`riffado.home.local`)
traefik.http.routers.riffado-int.entrypoints: websecure
traefik.http.routers.riffado-int.tls.certresolver: step-ca
traefik.http.routers.riffado-int.service: riffado-svc

traefik.http.services.riffado-svc.loadbalancer.server.port: 3000
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose

`/media/jeeves/1TB_Vol1/docker/riffado/audio`	`/app/storage`	Audio file storage (Ceph OSD)

Audio files are stored on the Ceph-managed OSD volume (`nvme2n1`, mounted at `/media/jeeves/1TB_Vol1`). This keeps large audio files off the system NVMe and on the dedicated storage volume.

Sub-section: PostgreSQL Database

`riffado-db` is a dedicated Postgres 16-alpine sidecar managing all Riffado application state: user accounts, playlists, audio metadata, and processing history. It is not shared with any other service.

Host Path / Volume	Container Path	Purpose
`/home/jeeves/docker/riffado/db`	`/var/lib/postgresql/data`	PostgreSQL data files

The database container is on the `riffado-internal` network only — it is never exposed to `traefik-net`.

Networks

Network	Purpose
`traefik-net`	Exposes the Riffado web UI
`riffado-internal`	`riffado` ↔ `riffado-db` communication

Dependencies

- `riffado-db` (must be healthy before `riffado` starts)
- Authentik for SSO on external route
- Ceph OSD volume must be mounted at `/media/jeeves/1TB_Vol1` before the container starts, otherwise the audio storage path is unavailable

Notes / Gotchas

- If the Ceph OSD volume is unmounted or degraded, audio file operations will fail even though the container itself runs normally.

- `APP_URL` must match the externally accessible URL. Changing this after initial setup requires updating any stored links.
- The `BETTER_AUTH_SECRET` and `ENCRYPTION_KEY` must remain constant — changing them invalidates all existing sessions and encrypted data.

Last Updated: 2026-06-16

Paperless-AI

Overview

PaperlessAI is an AI companion service for Paperless-NGX that automates document classification. It monitors the Paperless document inbox and uses a local Ollama model to automatically assign tags, document types, correspondents, and titles to newly ingested documents — eliminating the need for manual categorisation.

It communicates with Ollama over the internal `ai-internal` network and with Paperless-NGX via its API over the external HTTPS URL.

Access

Type	URL	Notes
Internal	`https://paperlessai.home.local`	Authentik SSO required
External	`https://paperlessai.jeeves5454.ddns.net`	Authentik SSO + GeoBlock + CrowdSec

Configuration

Image: `clusterzx/paperless-ai:latest` **Compose project:** Paperless stack (managed via Portainer, separate from `ai-stack`)

Environment Variables

Variable	Value / Notes
`LLM_PROVIDER`	`ollama`
`OLLAMA_URL`	`http://ollama.home.local:11434`
`OLLAMA_MODEL`	`gemma4:12b`
`PAPERLESS_URL`	`https://paperless.jeeves5454.ddns.net`
`PAPERLESS_API_KEY`	**REDACTED**

Variable	Value / Notes
`AUTO_MODE`	`true` — processes documents automatically
`AUTO_TAG`	`ai-processed` — applied to every AI-classified document
`NODE_ENV`	`production`

`AUTO_MODE=true` means PaperlessAI polls the Paperless inbox on a schedule and processes new documents without any manual trigger. The `ai-processed` tag is applied to documents after classification, allowing easy filtering in Paperless.

Traefik Labels

```
# External route
traefik.http.routers.paperlessai-ext.rule: Host(`paperlessai.jeeves5454.ddns.net`)
traefik.http.routers.paperlessai-ext.entrypoints: websecure
traefik.http.routers.paperlessai-ext.tls.certresolver: letsencrypt
traefik.http.routers.paperlessai-ext.middlewares: plex-geoblock@file,crowdsec-
bouncer@file,authentik-auth@docker
traefik.http.routers.paperlessai-ext.service: paperlessai-svc

# Internal route
traefik.http.routers.paperlessai-int.rule: Host(`paperlessai.home.local`)
traefik.http.routers.paperlessai-int.entrypoints: websecure
traefik.http.routers.paperlessai-int.tls.certresolver: step-ca
traefik.http.routers.paperlessai-int.middlewares: authentik-auth@docker
traefik.http.routers.paperlessai-int.service: paperlessai-svc

traefik.http.services.paperlessai-svc.loadbalancer.server.port: 3000
```

Note that Authentik middleware is applied on **both** internal and external routes.

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/paperless/paperless-ai-data`	`/app/data`	PaperlessAI configuration and state

Networks

PaperlessAI must be on the same network as Ollama to use `http://ollama.home.local:11434`. It reaches Paperless-NGX via its external HTTPS URL (outbound through Traefik).

Dependencies

- **Ollama** — must be running with the `gemma4:12b` model loaded for classification to work. Pull the model first:

  ```
  docker exec -it ollama ollama pull gemma4:12b
  ```

- **Paperless-NGX** — must be accessible at `https://paperless.jeeves5454.ddns.net` with a valid API key
- Authentik for SSO on both routes

Notes / Gotchas

- If Ollama is down or the model isn't loaded, documents will queue and not be processed until Ollama is available again.
- The `AUTO_TAG` (`ai-processed`) helps track which documents were classified by AI vs manually. Review newly tagged documents periodically to validate AI accuracy.
- `gemma4:12b` (12 billion parameters) fits comfortably within the RTX 5080's 16 GB VRAM. If Ollama is under pressure from simultaneous inference requests, PaperlessAI classification may be slower.
- PaperlessAI connects to Paperless via the external URL (`https://paperless.jeeves5454.ddns.net`). Ensure this domain remains reachable from within the container network.

Last Updated: 2026-06-16

n8n

Overview

N8N is the workflow automation platform for the homelab. It connects services together through visual, node-based workflows — handling tasks such as document pipeline automation, audio transcription orchestration (Minuspod), API integrations, scheduled jobs, and webhook-triggered processing.

N8N is the glue layer that ties together Whisper, Ollama, Paperless, and external APIs into end-to-end automated pipelines.

Access

Type	URL	Notes
Internal	`https://n8n.home.local`	LAN access via Step-CA TLS — basic auth

No external (internet-facing) Traefik route. Remote access via Tailscale.

Webhooks are received at `https://n8n.home.local/webhook/...` — internal and Tailscale-reachable only.

Configuration

Image: `n8nio/n8n:latest` **Compose project:** Standalone (managed via Portainer)

Ports

Port	Protocol	Purpose
`5678`	TCP	N8N web UI and API (host-bound)

Traefik Labels

```
traefik.enable: "true"
traefik.docker.network: traefik-net
traefik.http.routers.n8n.rule: Host(`n8n.home.local`)
traefik.http.routers.n8n.entrypoints: websecure
traefik.http.routers.n8n.tls.certresolver: step-ca
traefik.http.services.n8n.loadbalancer.server.port: 5678
```

Internal-only route.

Environment Variables

Variable	Value / Notes
`N8N_HOST`	`n8n.home.local`
`N8N_PROTOCOL`	`https`
`N8N_PORT`	`5678`
`N8N_EDITOR_BASE_URL`	`https://n8n.home.local`
`WEBHOOK_URL`	`https://n8n.home.local`
`N8N_BASIC_AUTH_ACTIVE`	`true`
`N8N_BASIC_AUTH_USER`	`jeeves`
`N8N_BASIC_AUTH_PASSWORD`	**REDACTED**
`N8N_ENCRYPTION_KEY`	**REDACTED** — encrypts stored credentials
`N8N_PAYLOAD_SIZE_MAX`	`16` (MB)
`EXECUTIONS_PROCESS`	`main`
`GENERIC_TIMEZONE`	`America/Toronto`

`N8N_ENCRYPTION_KEY` encrypts all stored credentials (API keys, passwords) in the N8N database. This key must remain constant — changing it invalidates all stored credentials and they must be re-entered.

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/n8n/data`	`/home/node/.n8n`	Workflows, credentials, execution history

All workflow definitions, credentials, and execution logs are stored in this bind-mounted directory. Back up before upgrades.

Networks

Network	Purpose
`traefik-net`	Exposes N8N UI and webhook endpoint

N8N calls other services by hostname (e.g. `http://ollama.home.local:11434`, `https://whisper.home.local`) — it must be on `traefik-net` or have DNS resolution for these names.

Dependencies

- AdGuard Home for `*.home.local` DNS resolution (N8N calls other services by name)
- Step-CA / Traefik for the `n8n.home.local` route
- Services called by workflows: Ollama, Faster-Whisper, Paperless-NGX, external APIs

Notes / Gotchas

- N8N stores all credentials encrypted with `N8N_ENCRYPTION_KEY`. If this key is lost or rotated, all stored API keys and passwords must be re-entered manually.
- `EXECUTIONS_PROCESS=main` runs workflow executions in the main process. For high concurrency, consider `EXECUTIONS_PROCESS=own` with worker processes, but this requires additional setup.
- Webhook URLs are internal only (`https://n8n.home.local/webhook/...`). External webhooks (e.g. from GitHub, Stripe) will not reach N8N unless a Tailscale exit node or Traefik external route is configured.
- `N8N_PAYLOAD_SIZE_MAX=16` limits incoming webhook payload size to 16 MB. Increase this if processing large audio or document payloads via webhook.
- The `latest` tag may introduce breaking changes between workflow schema versions. Pin to a specific N8N version before any significant workflow investments.

Last Updated: 2026-06-16

Open Notebook

Overview

Open Notebook is a self-hosted AI research notebook application — a local alternative to Google NotebookLM. It allows structured AI-assisted research sessions where sources (documents, URLs, text) are ingested into a notebook and queried via an LLM for synthesis, summaries, and question-answering.

It uses SurrealDB as its backend database for storing notebooks, sources, and conversation state.

Access

Type	URL	Notes
Internal	`https://notebook.home.local`	LAN access via Step-CA TLS
External	`https://notebook.jeeves5454.ddns.net`	Authentik SSO + GeoBlock + CrowdSec

Containers in This Stack

Container	Image	Role
`open-notebook`	`lfnovo/open_notebook:v1-latest`	Application server (Streamlit)
`open-notebook-db`	`surrealdb/surrealdb:v2`	SurrealDB database backend

Configuration

Compose project: `ai-stack`

Environment Variables

Variable	Value / Notes
`API_URL`	`https://notebook.home.local`
`CORS_ORIGINS`	`https://notebook.home.local,https://notebook.jeeves5454.ddns.net`
`INTERNAL_API_URL`	`http://localhost:5055`
`SURREAL_URL`	`ws://open-notebook-db:8000/rpc`
`SURREAL_NAMESPACE`	`open_notebook`
`SURREAL_DATABASE`	`open_notebook`
`SURREAL_USER`	`root`
`SURREAL_PASSWORD`	**REDACTED**
`HOSTNAME`	`0.0.0.0`

Traefik Labels

```
# External route
traefik.http.routers.open-notebook-ext.rule: Host(`notebook.jeeves5454.ddns.net`)
traefik.http.routers.open-notebook-ext.entrypoints: websecure
traefik.http.routers.open-notebook-ext.tls.certresolver: letsencrypt
traefik.http.routers.open-notebook-ext.middlewares: authentik-auth@docker,plex-
geoblock@file,crowdsec-bouncer@file
traefik.http.routers.open-notebook-ext.service: open-notebook-ui-svc
traefik.http.routers.open-notebook-ext.priority: 1

# Internal route
traefik.http.routers.open-notebook-int.rule: Host(`notebook.home.local`)
traefik.http.routers.open-notebook-int.entrypoints: websecure
traefik.http.routers.open-notebook-int.tls.certresolver: step-ca
traefik.http.routers.open-notebook-int.service: open-notebook-ui-svc
traefik.http.routers.open-notebook-int.priority: 1

traefik.http.services.open-notebook-ui-svc.loadbalancer.server.port: 8502
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose

`/media/jeeves/1TB_Vol1/docker/open-notebook/notebook_data`	`/app/data`	Notebook content and uploads (Ceph OSD)
`/home/jeeves/docker/open-notebook/surreal_data`	`/mydata`	SurrealDB data files

Notebook data (uploaded sources, generated content) is stored on the Ceph OSD volume (`1TB_Vol1`) to keep large research files off the system NVMe.

Sub-section: SurrealDB

`open-notebook-db` runs SurrealDB v2, a multi-model database used by Open Notebook to store all notebook definitions, source metadata, embeddings, and conversation history.

SurrealDB is connected to `open-notebook` via WebSocket at `ws://open-notebook-db:8000/rpc` over the `ai-internal` network. It is not exposed to `traefik-net` — no external access.

Networks

Network	Purpose
`ai-stack_ai-internal`	`open-notebook` ↔ `open-notebook-db` communication
`traefik-net`	Exposes the Open Notebook UI via Traefik

`open-notebook-db` is on `ai-internal` only and never on `traefik-net`.

Dependencies

- `open-notebook-db` (SurrealDB must be running before the app starts)
- Ollama or an external LLM API configured within the app settings
- Authentik for SSO on the external route
- Ceph OSD volume mounted at `/media/jeeves/1TB_Vol1`

Notes / Gotchas

- The `v1-latest` image tag tracks the latest v1 stable release. Breaking changes between major versions may require a database migration.
- SurrealDB stores its data in the bind-mounted `surreal_data` directory. This must be backed up before upgrades to SurrealDB v3+ as schema changes are not automatically

reversible.
- LLM provider settings (which Ollama model to use, API keys for external providers) are configured inside the Open Notebook UI, not via environment variables.
- `CORS_ORIGINS` must include both the internal and external URLs — missing one will cause browser CORS errors for the corresponding route.

Last Updated: 2026-06-16

09-mcp-github.md

kstack: book: Centerpoint Home Lab chapter: AI & Automation page: MCP GitHub Server tags: [mcp, github, oauth2, ai, claude]

Overview

The MCP GitHub Server exposes GitHub's Model Context Protocol (MCP) server over HTTPS with OAuth 2.1 authentication, making it accessible to Claude.ai and other MCP-compatible AI clients from anywhere on the internet.

The stack consists of two containers:

- `mcp-github-proxy` — a custom Node.js OAuth 2.1 proxy (locally built image) that validates JWTs from Authentik before forwarding requests to the MCP server.
- `github-mcp-server` — the official GitHub MCP server running in HTTP mode, internal-only, with access scoped to repos and issues.

Access

Type	URL	Notes
Internal	`https://mcp-github.home.local`	LAN access via Step-CA TLS
External	`https://mcp-github.jeeves5454.ddns.net`	OAuth 2.1 authenticated — no Authentik ForwardAuth

The external route uses **OAuth 2.1** (not Authentik ForwardAuth) as the auth layer. GeoBlock and CrowdSec are still applied. The OAuth 2.1 issuer is Authentik.

Containers in This Stack

Container	Image	Role
`mcp-github-proxy`	`mcp-github-proxy:latest` (local build)	OAuth 2.1 proxy + MCP request forwarder
`github-mcp-server`	`ghcr.io/github/github-mcp-server:latest`	GitHub MCP server (HTTP mode, internal)

Configuration

Compose project: `mcp-github` **Compose file:** `/home/jeeves/docker/mcp-github/proxy/docker-compose.yml`

`mcp-github-proxy` Environment Variables

Variable	Value / Notes
`PORT`	`3000`
`GITHUB_MCP_URL`	`http://github-mcp-server:8080`
`GITHUB_PERSONAL_ACCESS_TOKEN`	**REDACTED** — PAT with read-only scopes (Contents, Issues, Metadata)
`AUTHENTIK_ISSUER`	`https://auth.jeevesconsults.ca/application/o/mcp-github/`
`AUTHENTIK_JWKS_URL`	`https://auth.jeevesconsults.ca/application/o/mcp-github/jwks/`
`SERVER_URL`	`https://mcp-github.jeeves5454.ddns.net`

`github-mcp-server` Environment Variables

Variable	Value / Notes
`GITHUB_PERSONAL_ACCESS_TOKEN`	**REDACTED** — same read-only PAT as proxy
`GITHUB_TOOLSETS`	`repos,issues` — restricts available tools
`GITHUB_READ_ONLY`	`1` — belt-and-suspenders read-only flag

> **Security note:** `GITHUB_READ_ONLY=1` has a known bug in HTTP mode (github/github-mcp-server #2156) — toolset flags may not fully enforce read-only behaviour. Primary read-only enforcement is the PAT scopes themselves (Contents/Issues/Metadata: read only).

Traefik Labels

```
# External route — GeoBlock + CrowdSec only (OAuth 2.1 handles auth)
traefik.http.routers.mcp-github-ext.rule: Host(`mcp-github.jeeves5454.ddns.net`)
traefik.http.routers.mcp-github-ext.entrypoints: websecure
traefik.http.routers.mcp-github-ext.tls.certresolver: letsencrypt
traefik.http.routers.mcp-github-ext.middlewares: plex-geoblock@file,crowdsec-bouncer@file
traefik.http.routers.mcp-github-ext.service: mcp-github-proxy-svc

# Internal route
traefik.http.routers.mcp-github-int.rule: Host(`mcp-github.home.local`)
traefik.http.routers.mcp-github-int.entrypoints: websecure
traefik.http.routers.mcp-github-int.tls.certresolver: step-ca
traefik.http.routers.mcp-github-int.service: mcp-github-proxy-svc

traefik.http.services.mcp-github-proxy-svc.loadbalancer.server.port: 3000
```

Note: Authentik ForwardAuth (`authentik-auth@docker`) is **not** used here. OAuth 2.1 is the authentication mechanism — the proxy validates the `Authorization` header JWT against Authentik's JWKS endpoint directly.

Sub-section: GitHub MCP Server

`github-mcp-server` runs the official GitHub MCP server in HTTP mode on port `8080`, internal-only. It is never directly exposed to Traefik or the host — all requests arrive through the `mcp-github-proxy` over the `mcp-github-internal` network.

The server is restricted to `repos` and `issues` toolsets, providing read-only access to repository content, metadata, and issues. Write operations are not available via the PAT scopes.

Proxy Image Build

The `mcp-github-proxy` image is built locally on Centerpoint before deployment:

```
cd /home/jeeves/docker/mcp-github/proxy
docker build -t mcp-github-proxy:latest .
```

Source files: `/home/jeeves/docker/mcp-github/proxy/{Dockerfile,package.json,src/}`

Volumes / Bind Mounts

No persistent volumes required. Both containers are stateless — the proxy holds no state, and the MCP server reads from GitHub's API on every request.

Networks

Network	Purpose
`traefik-net`	Exposes `mcp-github-proxy` via Traefik
`mcp-github_mcp-github-internal`	`mcp-github-proxy` ↔ `github-mcp-server` communication

`github-mcp-server` is on the internal network only — never on `traefik-net`.

Healthchecks

Both containers have healthchecks defined:

```
healthcheck:
  test: ["CMD", "wget", "-qO-", "http://localhost:<port>/health"]
  interval: 30s
  timeout: 5s
  retries: 3
```

- Proxy healthcheck: `http://localhost:3000/health`
- MCP server healthcheck: `http://localhost:8080/health`

Authentik OAuth 2.1 Provider Setup

In Authentik, an OAuth2/OIDC provider is configured for MCP GitHub with:

Setting	Value
Redirect URI	`https://claude.ai/api/mcp/auth_callback`
Issuer	`https://auth.jeevesconsults.ca/application/o/mcp-github/`
JWKS URL	`<issuer>/jwks/`

Dependencies

- Authentik (`auth.jeevesconsults.ca`) — proxy validates JWTs against Authentik JWKS
- GitHub API (internet access required for the MCP server to function)
- Traefik on `traefik-net` for both routes

Notes / Gotchas

- The GitHub PAT must be rotated before expiry. If it expires, all GitHub MCP tool calls will fail with 401 errors. Update in both container environment variables.
- The proxy image must be rebuilt after any source code changes:

```
cd /home/jeeves/docker/mcp-github/proxy
docker build -t mcp-github-proxy:latest .
docker compose up -d --force-recreate mcp-github-proxy
```

- Claude.ai connects to this server via the external URL. The redirect URI (`https://claude.ai/api/mcp/auth_callback`) must be registered in the Authentik OAuth2 provider — adding any other redirect URI will cause the OAuth flow to fail.
- GeoBlock allows CA, US, and IN. Claude.ai's servers may originate from other regions — if MCP calls fail, check Traefik logs for GeoBlock rejections and adjust the country allowlist accordingly.

Last Updated: 2026-06-16