Infrastructure & Networking

Chapter Introduction

Overview

This chapter documents the foundational infrastructure layer that all other services depend on. These nine services form the backbone of the Centerpoint stack — handling reverse proxying, TLS, identity, DNS, threat detection, logging, and observability.

Services in This Chapter

Service	Container(s)	Purpose
Traefik v3	`traefik`	Reverse proxy and TLS termination
Step-CA	`step-ca`	Internal ACME certificate authority
Portainer EE	`portainer`	Docker container management UI
Authentik	`authentik-server`, `authentik-worker`, `authentik-postgresql`, `authentik-geoip`	SSO / identity provider
AdGuard Home	`adguardhome`	LAN DNS resolver and ad blocking
CrowdSec	`crowdsec`, `crowdsec-bouncer-traefik`	Collaborative threat detection and blocking
Dozzle	`dozzle`	Real-time container log viewer
Uptime-Kuma	`uptime-kuma`	Service uptime and endpoint monitoring
Homepage	`homepage`	Homelab dashboard

Dependency Order

When starting from scratch, services must come up in this order:

1. **AdGuard Home** — `*.home.local` DNS must resolve before anything can find its neighbours
2. **Step-CA** — required for Traefik to issue internal certificates on first boot

3. **Traefik** — all named HTTPS routes depend on it
4. **Authentik** (PostgreSQL → Server → Worker) — required before any externally-accessible service that enforces SSO
5. **CrowdSec** → **CrowdSec Bouncer** — bouncer cannot connect to LAPI until CrowdSec is healthy
6. All other infrastructure services (Portainer, Dozzle, Uptime-Kuma, Homepage) can start in any order

---

Last Updated: 2026-06-16

Traefik v3

Overview

Traefik is the single ingress point for all named HTTP/HTTPS traffic on Centerpoint. It runs on the `traefik-net` Docker bridge network and discovers routes automatically from container labels — no manual reload is needed when stacks are added or removed. All TLS termination happens at Traefik; individual application containers serve plain HTTP internally.

Access

Type	URL / Endpoint	Notes
Dashboard	`https://traefik.home.local`	Internal only — LAN access required

The dashboard is exposed only on the internal Step-CA route. There is no external (internet-facing) route for the Traefik dashboard.

Configuration

Image: `traefik:v3.6.13` **Compose project:** `traefik` (same project as `step-ca`)

Ports

Port	Protocol	Purpose
`80`	TCP	HTTP — auto-redirects all traffic to HTTPS
`443`	TCP	HTTPS — primary entrypoint
`8080`	TCP	Traefik API / Dashboard (internal)

Static Configuration — `traefik.yml`

Location: `/home/jeeves/docker/traefik/traefik.yml`

Key settings:

```
api:
  dashboard: true
  insecure: false

entryPoints:
  web:
    address: ":80"
    http:
      redirections:
        entryPoint:
          to: websecure
          scheme: https
  websecure:
    address: ":443"

providers:
  docker:
    network: traefik-net
    exposedByDefault: false
  file:
    directory: /config
    watch: true

certificatesResolvers:
  letsencrypt:
    acme:
      email: <redacted>
      storage: /letsencrypt/acme.json
      httpChallenge:
        entryPoint: web
  step-ca:
    acme:
      email: <redacted>
      storage: /letsencrypt/step-ca.json
      caServer: https://ca.home.local:9000/acme/acme/directory
      certificatesDuration: 720
```

```
experimental:
  plugins:
    geoblock:
      moduleName: github.com/PascalMinder/geoblock
      version: v0.3.6
    crowdsec-bouncer-traefik-plugin:
      moduleName: github.com/maxlerebourg/crowdsec-bouncer-traefik-plugin
      version: v1.3.0
```

Certificate Resolvers

Resolver	Scope	Method	CA	Duration
`letsencrypt`	External (`*.jeeves5454.ddns.net`)	HTTP challenge	Let's Encrypt	90 days
`step-ca`	Internal (`*.home.local`)	ACME	Internal Step-CA at `ca.home.local:9000`	30 days

Plugins

Plugin	Version	Purpose
`PascalMinder/geoblock`	v0.3.6	Blocks requests from countries not in allowlist
`maxlerebourg/crowdsec-bouncer-traefik-plugin`	v1.3.0	Forwards requests to CrowdSec bouncer for IP checks

Geoblock is configured to allow **CA**, **US**, and **IN** only. The middleware is named `plex-geoblock@file` and is defined in `/home/jeeves/docker/traefik/config/crowdsec.yml`.

Dynamic Configuration

Location: `/home/jeeves/docker/traefik/config/` (file-watched)

File	Contents
`crowdsec.yml`	`plex-geoblock` and `crowdsec-bouncer` middleware defs
`dynamic.yml`	Static file-provider routes for Plex (no Docker labels)
`middlewares.yml`	Additional shared middleware definitions

Standard Dual-Route Label Pattern

Each service that requires both internal and external access uses this label pattern in its `docker-compose.yml`:

```
labels:
  - "traefik.enable=true"

  # External route — Let's Encrypt TLS + security middleware
  - "traefik.http.routers.<name>-ext.rule=Host(`<svc>.jeeves5454.ddns.net`)"
  - "traefik.http.routers.<name>-ext.entrypoints=websecure"
  - "traefik.http.routers.<name>-ext.tls.certresolver=letsencrypt"
  - "traefik.http.routers.<name>-ext.middlewares=authentik-auth@docker,plex-
geoblock@file,crowdsec-bouncer@file"

  # Internal route — Step-CA TLS, no SSO middleware
  - "traefik.http.routers.<name>-int.rule=Host(`<svc>.home.local`)"
  - "traefik.http.routers.<name>-int.entrypoints=websecure"
  - "traefik.http.routers.<name>-int.tls.certresolver=step-ca"

  # Backend
  - "traefik.http.services.<name>-svc.loadbalancer.server.port=<port>"
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/traefik/traefik.yml`	`/traefik.yml`	Static config
`/home/jeeves/docker/traefik/config`	`/config`	Dynamic config (file-watched)
`/home/jeeves/docker/traefik/letsencrypt`	`/letsencrypt`	ACME certificate storage
`/var/log/traefik`	`/var/log/traefik`	Access logs (JSON)
`/var/run/docker.sock`	`/var/run/docker.sock`	Docker label discovery

Dependencies

- `traefik-net` Docker network (must be created manually before first start)
- `step-ca` container (must be healthy for internal certs to renew)
- CrowdSec LAPI at `crowdsec:8080` (bouncer requires it; Traefik starts without it but will error on blocked requests)

Notes / Gotchas

- `traefik-net` is externally managed — always create it first:

```
docker network create traefik-net
```

- Traefik discovers routes from container labels only on containers attached to `traefik-net`. If a container is not on this network it will be invisible to Traefik even with `traefik.enable=true`.
- `exposedByDefault: false` means every service must explicitly opt in with `traefik.enable=true`.
- The `step-ca` certificate resolver stores its ACME account and certs in `/letsencrypt/step-ca.json` (separate from the Let's Encrypt storage). Do not delete this file or certificates will be re-issued from scratch.
- Access logs are written to `/var/log/traefik/access.log` in JSON format and are also bind-mounted into the CrowdSec container for log ingestion.

Last Updated: 2026-06-16

Step-CA (Internal Certificate Authority)

Overview

Step-CA is an internal ACME-compatible certificate authority that issues TLS certificates for all `*.home.local` domains. It allows Traefik to provision and auto-renew trusted certificates for internal services without relying on Let's Encrypt or exposing any ports to the internet.

Clients and browsers on the LAN trust `*.home.local` certificates because the Step-CA root certificate has been manually installed as a trusted CA on each device.

Access

Type	URL / Endpoint	Notes
ACME API	`https://ca.home.local:9000/acme/acme/directory`	Used by Traefik only
Step-CA UI	N/A	No web UI — CLI managed only

Step-CA does not have a browser-accessible dashboard. Management is via the `step` CLI tool.

Configuration

Image: `smallstep/step-ca:latest` **Compose project:** `traefik` (same project as `traefik` container)

Ports

Port	Protocol	Purpose
`9000`	TCP	ACME API (host-bound: `0.0.0.0:9000`)

Port 9000 is bound directly to the host so that Traefik (and any other ACME client on the LAN) can reach it at `ca.home.local:9000`.

CA Configuration — `ca.json`

Location: `/home/jeeves/docker/step-ca/config/config/ca.json`

Key settings (secrets redacted):

```
{
  "root": "/home/step/certs/root_ca.crt",
  "crt": "/home/step/certs/intermediate_ca.crt",
  "key": "/home/step/secrets/intermediate_ca_key",
  "address": "0.0.0.0:9000",
  "dnsNames": ["ca.home.local"],
  "db": {
    "type": "badger",
    "dataSource": "/home/step/db"
  },
  "authority": {
    "provisioners": [
      {
        "type": "ACME",
        "name": "acme"
      }
    ]
  }
}
```

Certificate duration for ACME-issued certs is set to **720 hours (30 days)** in Traefik's `step-ca` resolver config. Traefik renews automatically before expiry.

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/step-ca/config`	`/home/step`	CA configuration, certificates, database

Directory Layout Inside the Volume

```
/home/jeeves/docker/step-ca/config/
├── config/
│   ├── ca.json          ← Main CA configuration
│   └── defaults.json    ← Step CLI defaults
├── certs/
│   ├── root_ca.crt      ← Root CA certificate (install this on client devices)
│   └── intermediate_ca.crt
├── secrets/             ← Private keys — never expose these
│   ├── root_ca_key
│   ├── intermediate_ca_key
│   └── password         ← Key encryption password (REDACTED)
└── db/                  ← BadgerDB certificate issuance database
```

Dependencies

- AdGuard Home — `ca.home.local` must resolve on the LAN for Traefik to reach the ACME endpoint
- No other service dependencies

Trusting the Root CA on Client Devices

Every device that accesses `*.home.local` URLs in a browser must trust the Step-CA root certificate. The root cert is at:

```
/home/jeeves/docker/step-ca/config/certs/root_ca.crt
```

Also available at `/home/jeeves/docker/traefik/step-ca-root.crt` (Traefik keeps a copy for its own ACME client trust store).

macOS

```
sudo security add-trusted-cert -d -r trustRoot -k /Library/Keychains/System.keychain
root_ca.crt
```

Windows

```
Import-Certificate -FilePath root_ca.crt -CertStoreLocation Cert:\LocalMachine\Root
```

Ubuntu / Debian

```
sudo cp root_ca.crt /usr/local/share/ca-certificates/step-ca.crt
sudo update-ca-certificates
```

Notes / Gotchas

- The Step-CA password file (`/home/step/secrets/password`) is read on startup. If the volume is lost, the CA must be re-initialized and all client devices must re-import the new root certificate.
- Do not delete the `db/` directory — it contains the certificate issuance history. Loss means Step-CA cannot check for revoked certificates.
- Step-CA is in the same Portainer compose project as Traefik. Restarting the Traefik stack will also restart Step-CA briefly. Schedule this during off-hours.
- Traefik caches issued certs in its ACME storage file. If Step-CA is temporarily down, Traefik will continue serving existing certs until they near expiry.

Last Updated: 2026-06-16

Portainer EE

Overview

Portainer Enterprise Edition is the primary container management interface for Centerpoint. It provides a web UI for deploying stacks, viewing container logs, managing volumes and networks, and monitoring resource usage across the Docker environment.

All compose stacks are deployed and managed through Portainer rather than by running `docker compose` directly on the host. Portainer stores stack definitions internally (under `/data/compose/`) and rebuilds containers from those definitions when updated.

Access

Type	URL	Notes
Internal	`https://portainer.home.local`	LAN access via Step-CA TLS
HTTPS UI	`https://192.168.1.85:9443`	Direct IP fallback
Tunnel	Port `8001` (agent)	For Edge Agent / remote access

Portainer does not have an external (internet-facing) Traefik route. Access is LAN-only or via Tailscale.

Configuration

Image: `portainer/portainer-ee:latest` **Compose project:** `portainer`

Ports

Port	Protocol	Purpose
`8001`	TCP	Edge Agent tunnel port
`9443`	TCP	HTTPS management UI (host-bound)

Traefik Labels

```
traefik.enable: "true"
traefik.http.routers.portainer.rule: Host(`portainer.home.local`)
traefik.http.routers.portainer.entrypoints: websecure
traefik.http.routers.portainer.tls.certresolver: step-ca
traefik.http.services.portainer.loadbalancer.server.port: 9443
```

Internal-only route — no external Traefik router.

Volumes / Bind Mounts

Host Path / Volume	Container Path	Purpose
`portainer_portainer_data`	`/data`	Portainer state and stack data (named volume, external)
`/var/run/docker.sock`	`/var/run/docker.sock`	Direct Docker socket access

> The `portainer_portainer_data` volume is declared `external: true` — it must exist before the stack is started.

Note: Portainer mounts the Docker socket directly. This is intentional for Portainer EE; it is the only service with direct socket access.

Dependencies

- Docker socket (`/var/run/docker.sock`) — no other service dependencies
- Traefik on `traefik-net` for the `portainer.home.local` route
- Step-CA for the internal TLS certificate

Notes / Gotchas

- Portainer stores all deployed stack definitions under `/data/compose/<id>/v<version>/`. These are the authoritative copies of each stack's `docker-compose.yml` while managed through Portainer. Files under `/home/jeeves/docker/` may be out of date if a stack was

edited directly in the Portainer UI.
- **Portainer EE licence key** is stored in the Portainer UI and must be re-entered if the `portainer_portainer_data` volume is lost.
- The `portainer-ee:latest` tag follows the latest stable EE release. Pin to a specific version (e.g. `portainer-ee:2.22.0`) before any planned maintenance to avoid unintended upgrades.
- Portainer agent is not separately deployed on Centerpoint — Portainer connects to its local Docker daemon directly via the socket.
- Portainer does not work over Traefik - I have not been troubleshooting this, and is a future action or Task to manage

FUTURE WORK

- Identify issue with Portainer and Traefik interaction
- Migrate Portainer cert with Step-CA cert for consistent cert management

---

Last Updated: 2026-06-16

Authentik (SSO / Identity Provider)

Overview

Authentik is the identity provider and single sign-on (SSO) gateway for all externally-accessible services on Centerpoint. It implements ForwardAuth middleware for Traefik, meaning that any request arriving at an external route tagged with `authentik-auth@docker` is intercepted and validated by Authentik before being forwarded to the backend service.

Authentik also serves as an OAuth2/OIDC provider — services like BookStack use OIDC for native user login rather than ForwardAuth.

Access

Type	URL	Notes
Internal	`https://auth.home.local`	LAN access via Step-CA TLS
External	`https://auth.jeevesconsults.ca`	Internet-facing — OIDC redirect target

The external URL (`auth.jeevesconsults.ca`) is the one registered as the OIDC issuer with downstream services. It must be reachable by browsers completing OAuth2 flows.

Containers in This Stack

Container	Image	Role
`authentik-server`	`ghcr.io/goauthentik/server:2026.5.x`	HTTP server + ForwardAuth endpoint
`authentik-worker`	`ghcr.io/goauthentik/server:2026.5.x`	Background task worker (Rust entry)
`authentik-postgresql`	`postgres:16-alpine`	Primary database
`authentik-geoip`	`ghcr.io/maxmind/geoipupdate:v7.x`	GeoIP database updater (MaxMind)

Startup Order

`authentik-postgresql` must reach a healthy state before `authentik-server` and `authentik-worker` start. The `authentik-geoip` sidecar runs independently.

Configuration

Compose project: `authentik`

Key Environment Variables

Variable	Value / Notes
`AUTHENTIK_SECRET_KEY`	**REDACTED** — long random string, must stay constant
`AUTHENTIK_POSTGRESQL__HOST`	`authentik-postgresql`
`AUTHENTIK_POSTGRESQL__NAME`	`authentik`
`AUTHENTIK_POSTGRESQL__USER`	`authentik`
`AUTHENTIK_POSTGRESQL__PASSWORD`	**REDACTED**
`AUTHENTIK_REDIS__HOST`	`redis` (internal sidecar or external Redis)
`AUTHENTIK_LISTEN__HTTP`	`0.0.0.0:9000` — explicit bind required for Docker bridge
`AUTHENTIK_ERROR_REPORTING__ENABLED`	`false`

> **Important:** `AUTHENTIK_LISTEN__HTTP: "0.0.0.0:9000"` is required in Authentik 2026.5+. Newer versions default to `[::]` (IPv6 wildcard) which Docker bridge networks cannot reach via IPv4. Without this override, ForwardAuth requests from Traefik fail silently.

ForwardAuth Middleware

The Authentik ForwardAuth middleware is defined on the `authentik-server` container labels and is available to all services on `traefik-net` as:

```
authentik-auth@docker
```

Example usage on a protected external route:

```
labels:
  - "traefik.http.routers.<name>-ext.middlewares=authentik-auth@docker,plex-
geoblock@file,crowdsec-bouncer@file"
```

OIDC Configuration (for native OIDC apps)

Applications using OIDC (e.g. BookStack) configure Authentik as their provider with the following settings:

Setting	Value
Issuer URL	`https://auth.jeevesconsults.ca/application/o/<app-slug>/`
JWKS URL	`<issuer>/.well-known/jwks.json`
Client ID	Per-application (set in Authentik admin UI)
Client Secret	**REDACTED** — set per-application
`email_verified`	Requires custom Scope Mapping returning `true` (2026.5+ change)

Volumes / Bind Mounts

Host Path / Volume	Container Path	Purpose
`authentik_media`	`/media`	Uploaded assets (logos, avatars)
`authentik_custom-templates`	`/templates`	Custom email / flow templates
`authentik_postgresql_data`	`/var/lib/postgresql/data`	PostgreSQL data directory
`authentik_geoip_data`	`/usr/share/GeoIP`	MaxMind GeoIP databases

All volumes are named Docker volumes managed by Portainer.

Sub-section: PostgreSQL

The `authentik-postgresql` container is a dedicated Postgres 16 sidecar that stores all Authentik state: users, groups, policies, flows, tokens, and audit logs. It is not shared with any other service.

Database credentials are passed to the server via `AUTHENTIK_POSTGRESQL__*` environment variables. The container is on the `authentik-internal` network only — it is never exposed to `traefik-net` or the host.

Sub-section: GeoIP Updater

`authentik-geoip` runs the MaxMind GeoIPUpdate daemon, which downloads and refreshes the `GeoLite2-City` and `GeoLite2-ASN` databases on a schedule. The databases are shared into `authentik-server` via the `authentik_geoip_data` volume.

A MaxMind account and licence key (**REDACTED**) are required for the GeoIP databases.

Sub-section: Worker

`authentik-worker` runs as the same container image as `authentik-server` but with the worker entrypoint. It handles background tasks: email delivery, outpost health checks, blueprint application, and event cleanup. Since Authentik 2025.10+, the worker uses a Rust-based entrypoint for improved performance.

Dependencies

- `authentik-postgresql` (must be healthy before server/worker start)
- `traefik-net` (server must be on this network for ForwardAuth to reach it)
- MaxMind account for GeoIP updates (not strictly required, but GeoIP features will be unavailable without it)

Notes / Gotchas

- If the `AUTHENTIK_SECRET_KEY` changes, all tokens, sessions, and cookies are immediately invalidated — all users will be logged out across all services.
- Authentik's admin interface is at `/if/admin/`. Initial admin credentials are set via the `AUTHENTIK_BOOTSTRAP_PASSWORD` env var on first start.
- For OIDC apps: `email_verified` defaults to `false` since Authentik 2025.10. Create a custom Scope Mapping that hard-codes `"email_verified": True` and attach it to the OAuth2 provider.
- Existing local users who also log in via OIDC must have their **External Auth ID** set in Admin → Users → Edit → External Auth ID to match the `sub` claim from the OIDC token, otherwise two separate accounts will be created.
- `DOZZLE_ENABLE_SHELL=true` on Dozzle requires that this admin account is separate from the Authentik service account.

Last Updated: 2026-06-16

AdGuard Home

Overview

AdGuard Home serves as the **primary** LAN-wide DNS resolver for the entire homelab network. All devices on the `192.168.1.0/24` subnet use it as their primary DNS server (`192.168.1.85:53`). It provides:

- DNS-based ad and tracker blocking
- Wildcard DNS entry `*.home.local → 192.168.1.85` enabling all Traefik internal routes
- DNS rewrites for custom local hostnames
- Query logging and per-client statistics
- Upstream DNS forwarding (via DoH to external resolvers)

A **secondary AdGuard Home** instance runs on the CorSec server (Home Assistant machine at `192.168.1.64`). DNS rewrites are also maintained there so that name resolution continues during Centerpoint maintenance or restarts. Client devices should have both IPs configured as DNS resolvers.

Access

Type	URL	Notes
Internal	`https://adguard.home.local`	LAN access via Step-CA TLS
Direct	`http://192.168.1.85:80`	Plain HTTP UI (AdGuard internal port)

DNS service itself runs on port `53` (TCP + UDP), bound to the host IP.

Configuration

Image: `adguard/adguardhome:v0.107.71` **Compose project:** `adguardhome` (managed via Portainer)

Ports

Port	Protocol	Purpose
`53`	TCP+UDP	DNS resolver (host-bound)
`80`	TCP	AdGuard web UI (HTTP, internal)
`443`	TCP	DNS-over-HTTPS
`853`	TCP	DNS-over-TLS
`3000`	TCP	Initial setup port

Only port `53` is bound to the host. Other ports are exposed only within `traefik-net`.

Traefik Labels

```
traefik.enable: "true"
traefik.http.routers.adguard.rule: Host(`adguard.home.local`)
traefik.http.routers.adguard.entrypoints: websecure
traefik.http.routers.adguard.tls.certresolver: step-ca
traefik.http.services.adguard.loadbalancer.server.port: 80
```

Internal-only route — no external Traefik router.

Critical Wildcard DNS Entry

AdGuard Home (both primary and secondary) must have a DNS Rewrite configured as:

```
*.home.local → 192.168.1.85
```

This single wildcard record means Traefik receives all `*.home.local` HTTP requests and routes them by hostname. Without it, no internal service domain resolves.

All custom DNS rewrites (for individual hostnames outside the wildcard) must be kept in sync between the Centerpoint and CorSec instances.

Upstream DNS Resolvers

Upstream DNS (for forwarding public queries) is configured in AdGuard's settings UI. Common configuration:

- Primary: `https://dns.cloudflare.com/dns-query` (DoH)
- Secondary: `https://dns.google/dns-query` (DoH)

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/adguard/conf`	`/opt/adguardhome/conf`	Configuration (`AdGuardHome.yaml`)
`/home/jeeves/docker/adguard/work`	`/opt/adguardhome/work`	Query logs and statistics database

Networks

AdGuard Home is on `traefik-net` for its web UI route and has port `53` bound directly to the host for DNS service.

Dependencies

- None (AdGuard must start first; it has no upstream service dependencies)
- AdGuard's own DNS resolution bootstraps via the upstream DoH resolvers using hardcoded IPs if DNS is not yet available

High Availability

Instance	Host	IP	Role
Primary	Centerpoint	`192.168.1.85`	Primary — full config, rewrites, logging
Secondary	CorSec (Home Assistant)	`192.168.1.64`	Backup — same rewrites, ad-block lists

Client devices should configure both DNS servers in priority order. The secondary takes over automatically if the primary becomes unreachable.

DNS rewrites must be manually kept in sync between the two instances — there is no automated synchronisation.

Notes / Gotchas

- The `work/data` directory requires root-level permissions — the container runs as root. This is expected behaviour.
- If port `53` is already in use on the host (e.g. `systemd-resolved`), AdGuard will fail to start. On Ubuntu 24.04, disable the stub listener:

```
sudo systemctl disable --now systemd-resolved
```

- Client-specific rules (blocking, bypass, custom upstream) are stored in `AdGuardHome.yaml` and are preserved across container upgrades as long as the `conf/` bind mount is intact.
- AdGuard Home should be pinned to a specific version tag rather than `latest` to avoid schema migrations breaking the config on unexpected upgrades.
- When adding a new `*.home.local` service, no DNS change is needed (the wildcard covers it). New custom hostnames outside the wildcard pattern must be added to both the primary and secondary AdGuard instances.

FUTURE WORK

- Update the DNS blocklist to include more sites
- Provide a summary of sites blocked via AI for review and/or include in Grafana for trend analysis to know if the numbers go up or down over time- find rogue machines

---

Last Updated: 2026-06-16

CrowdSec

Overview

CrowdSec is a collaborative intrusion detection and prevention system integrated directly into Traefik. It analyses Traefik access logs in real time, detects attack patterns (brute force, CVE exploitation, bad bots, etc.), and instructs Traefik to block flagged IPs before they reach any backend service.

The stack consists of two containers:

- **`crowdsec`** — the Security Engine (LAPI + agent). Ingests logs, applies detection scenarios, maintains a decision list, and shares signals with the CrowdSec community.
- **`crowdsec-bouncer-traefik`** — the Traefik bouncer. Traefik forwards every incoming request to it via ForwardAuth; the bouncer checks the IP against CrowdSec's local decision database and returns allow/deny.

Access

Neither CrowdSec container has a web UI. Management is via the `cscli` CLI inside the `crowdsec` container:

```
docker exec -it crowdsec cscli decisions list
docker exec -it crowdsec cscli alerts list
docker exec -it crowdsec cscli metrics
```

Configuration

Compose project: `crowdsec` (managed via Portainer)

Containers

`crowdsec` — Security Engine

Image: `crowdsecurity/crowdsec:latest`

Property	Value
Network	`traefik-net`
Restart policy	`unless-stopped`

Key environment variables:

Variable	Value / Notes
`GID`	`1000`
`COLLECTIONS`	`crowdsecurity/traefik crowdsecurity/http-cve crowdsecurity/whitelist-good-actors`
`BOUNCER_KEY_TRAEFIK`	**REDACTED** — shared secret used by the bouncer to authenticate with LAPI

Collections installed:

Collection	Purpose
`crowdsecurity/traefik`	Detects attack patterns in Traefik access logs
`crowdsecurity/http-cve`	Detects exploitation of known HTTP CVEs
`crowdsecurity/whitelist-good-actors`	Whitelists known-good crawlers and services

`crowdsec-bouncer-traefik` — Traefik Bouncer

Image: `fbonalair/traefik-crowdsec-bouncer:latest`

Property	Value
Network	`traefik-net`
Internal port	`8080` (ForwardAuth endpoint)

Key environment variables:

Variable	Value / Notes
`GIN_MODE`	`release`
`CROWDSEC_AGENT_HOST`	`crowdsec:8080` — CrowdSec LAPI endpoint
`CROWDSEC_BOUNCER_API_KEY`	**REDACTED** — must match `BOUNCER_KEY_TRAEFIK`

How the Bouncer Integrates with Traefik

The middleware is defined in `/home/jeeves/docker/traefik/config/crowdsec.yml`:

```
http:
  middlewares:
    crowdsec-bouncer:
      forwardAuth:
        address: http://crowdsec-bouncer-traefik:8080/api/v1/forwardAuth
        trustForwardHeader: true
```

This middleware is referenced on external Traefik routes as `crowdsec-bouncer@file`.

Log Ingestion

CrowdSec reads Traefik access logs from two bind-mounted paths:

Host Path	Container Path	Notes
`/var/log/traefik`	`/var/log/traefik:ro`	Primary log location
`/home/jeeves/docker/traefik/logs`	`/logs/traefik:ro`	Secondary / rotated logs

The `acquis.yaml` config file at `/home/jeeves/docker/crowdsec/config/acquis.yaml` defines which log files to tail and in what format.

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/crowdsec/data`	`/var/lib/crowdsec/data`	Decision database and GeoIP data
`/home/jeeves/docker/crowdsec/config`	`/etc/crowdsec`	Scenarios, parsers, config
`/var/log/traefik`	`/var/log/traefik:ro`	Traefik access log (read-only)
`/home/jeeves/docker/traefik/logs`	`/logs/traefik:ro`	Traefik rotated logs (read-only)

Dependencies

- Traefik (must be running and writing access logs for CrowdSec to process)
- `crowdsec` LAPI must be healthy before `crowdsec-bouncer-traefik` starts
- Internet access required for CrowdSec community hub sync (pulling updated block lists)

Notes / Gotchas

- The `BOUNCER_KEY_TRAEFIK` / `CROWDSEC_BOUNCER_API_KEY` pair must match exactly. If mismatched, the bouncer will fail to authenticate with LAPI and Traefik will receive a 200 (passthrough) response from ForwardAuth — **effectively disabling the bouncer silently**.
- CrowdSec community sharing (`online_api_credentials.yaml`) requires an account at `app.crowdsec.net`. The credentials file is stored at `/home/jeeves/docker/crowdsec/config/online_api_credentials.yaml`.
- `crowdsec-bouncer@file` middleware must be applied to external routes only. Applying it to internal (`*.home.local`) routes will check LAN IPs against the community block list, which may block legitimate traffic unexpectedly.
- Manually ban an IP with:

  ```
  docker exec -it crowdsec cscli decisions add --ip <IP> --duration 24h --reason "manual"
  ```

- Remove a ban:

  ```
  docker exec -it crowdsec cscli decisions delete --ip <IP>
  ```

Last Updated: 2026-06-16

Dozzle

Overview

Dozzle is a lightweight, real-time log viewer for Docker containers. It provides a web UI to stream, search, and follow container logs without needing to SSH into the host and run `docker logs`. It is configured to monitor containers on both Centerpoint and Lusankya (the second homelab server at `192.168.1.114`).

Access

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

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

Configuration

Image: `amir20/dozzle:latest` (v10.6.0 at time of writing) **Compose project:** `dozzle` (managed via Portainer)

Environment Variables

Variable	Value	Purpose
`DOZZLE_HOSTNAME`	`Centerpoint`	Display name for the local host
`DOZZLE_REMOTE_HOST`	`tcp://192.168.1.114:2375|Lusankya`	Adds Lusankya as a remote Docker host
`DOZZLE_ENABLE_ACTIONS`	`false`	Disables container start/stop from UI
`DOZZLE_ENABLE_SHELL`	`true`	Enables shell access to containers

> `DOZZLE_ENABLE_SHELL=true` allows executing shell commands inside any container from the Dozzle UI. This is a powerful capability — ensure Dozzle is not accessible externally.

Traefik Labels

```
traefik.enable: "true"
traefik.http.routers.dozzle.rule: Host(`dozzle.home.local`)
traefik.http.routers.dozzle.entrypoints: websecure
traefik.http.routers.dozzle.tls.certresolver: step-ca
traefik.http.services.dozzle.loadbalancer.server.port: 8080
```

Internal-only route. No Authentik ForwardAuth on this route — access is controlled at the network level (LAN + Tailscale only).

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/var/run/docker.sock`	`/var/run/docker.sock:ro`	Read-only Docker socket access

Dozzle mounts the Docker socket read-only. No persistent data volume is required — Dozzle streams logs directly from the Docker daemon and does not store them.

Networks

Network	Purpose
`traefik-net`	Exposes the Dozzle web UI

Remote Hosts

Dozzle connects to Lusankya's Docker daemon at `tcp://192.168.1.114:2375`. This requires Lusankya's Docker daemon to have TCP exposure enabled. Logs from both Centerpoint and Lusankya containers are visible in a single Dozzle instance.

Dependencies

- Docker socket on Centerpoint (`/var/run/docker.sock`)
- Lusankya Docker daemon accessible at `tcp://192.168.1.114:2375`
- Traefik on `traefik-net` for the `dozzle.home.local` route

Notes / Gotchas

- `DOZZLE_ENABLE_SHELL=true` gives the ability to `exec` into any container from the browser. Treat this with the same care as SSH access to the host.
- Dozzle does not persist logs. Once a container is removed, its historical logs are no longer accessible from Dozzle (use `docker logs` or a log aggregation stack for archival).
- The `latest` image tag tracks the latest release. Dozzle updates frequently — check the changelog before pulling, especially for breaking UI changes.
- Lusankya's Docker TCP port (`2375`) is unencrypted. This is acceptable on the private LAN but should never be exposed externally.

Last Updated: 2026-06-16

Uptime-Kuma

Overview

Uptime-Kuma is a self-hosted uptime monitoring tool that tracks the availability of services and endpoints, sends alerts on downtime, and displays historical uptime statistics. It serves as the primary observability dashboard for the homelab — monitoring both internal services and external URLs.

Access

Type	URL	Notes
Internal	(via Tailscale or LAN direct)	Port `3001` on `192.168.1.85`
External	`https://uptime.jeeves5454.ddns.net`	Internet-facing, protected by Authentik + GeoBlock + CrowdSec

Configuration

Image: `louislam/uptime-kuma:2` **Compose project:** `uptime-kuma` (managed via Portainer; local file at `/home/jeeves/docker/uptime_kuma/docker-compose.yml`)

Ports

Port	Protocol	Purpose
`3001`	TCP	Web UI (also bound to host for direct access)

Traefik Labels

```
traefik.enable: "true"
traefik.http.routers.uptime.rule: Host(`uptime.jeeves5454.ddns.net`)
```

```
traefik.http.routers.uptime.entrypoints: websecure
traefik.http.routers.uptime.tls.certresolver: letsencrypt
traefik.http.routers.uptime.middlewares: authentik-auth@docker,plex-geoblock@file,crowdsec-bouncer@file,uptime-headers
traefik.http.middlewares.uptime-headers.headers.customrequestheaders.X-Forwarded-Proto: https
traefik.http.services.uptime.loadbalancer.server.port: 3001
```

The `uptime-headers` middleware injects `X-Forwarded-Proto: https` — required because Uptime-Kuma needs to know the original scheme for correct redirect handling.

External access is gated behind Authentik SSO, GeoBlock (CA/US/IN), and CrowdSec.

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/uptime_kuma/uptime-kuma-data`	`/app/data`	SQLite DB, config, logs

The data directory stores:

- `kuma.db` — SQLite database with monitor config, history, and incidents
- `error.log` — Uptime-Kuma application error log

Networks

Network	Purpose
`traefik-net`	External Traefik route
`uptime-kuma_monitoring`	Internal per-stack network

Dependencies

- Traefik on `traefik-net` for the external HTTPS route
- Authentik at `auth.jeevesconsults.ca` for SSO on the external route
- Network connectivity to all monitored endpoints

Notes / Gotchas

- Uptime-Kuma v2 uses SQLite. The `kuma.db` file is the entire state of the service — back it up before any upgrade.
- The `X-Forwarded-Proto` header middleware (`uptime-headers`) is specifically required for Uptime-Kuma to generate correct URLs in status page embeds and notifications.
- Monitor configurations (which services to check, intervals, notifications) are all stored in the UI and persisted in the SQLite database. There is no YAML-based config file.
- If Uptime-Kuma reports false positives for internal services, verify that DNS resolution works correctly from within the Uptime-Kuma container on `traefik-net`.
- Notification integrations (email, Telegram, Slack, etc.) are configured in the UI under Settings → Notifications.

Last Updated: 2026-06-16

Homepage.Dev

Overview

Homepage (gethomepage.dev) is the self-hosted service dashboard for Centerpoint. It provides a single-pane-of-glass view of all running services, organised into three tabs that reflect the different access methods available for each service. It also displays live Docker container status by connecting directly to the Docker socket.

Access

Type	URL	Notes
Internal	`https://centerpoint.home.local`	LAN access via Step-CA TLS
Direct	`http://192.168.1.85:3003`	Plain HTTP fallback

No external (internet-facing) Traefik route.

Configuration

Image: `ghcr.io/gethomepage/homepage:latest` (v1.13.2 at time of writing) **Compose project:** `homepage` (Portainer-managed; local file at `/home/jeeves/docker/homepage/docker-compose.yml`)

Ports

Port	Protocol	Purpose
`3003`	TCP	Web UI (mapped from internal `3000`)

Traefik Labels

```
traefik.enable: "true"
traefik.http.routers.homepage.rule: Host(`centerpoint.home.local`)
```

```
traefik.http.routers.homepage.entrypoints: websecure
traefik.http.routers.homepage.tls.certresolver: step-ca
traefik.http.services.homepage.loadbalancer.server.port: 3000
```

Internal-only route on `centerpoint.home.local`.

Environment Variables

Variable	Value
`PUID`	`1000`
`PGID`	`1000`
`LOG_TARGETS`	`stdout`
`HOMEPAGE_ALLOWED_HOSTS`	`gethomepage.dev,192.168.1.64:3003,localhost:3003`

Dashboard Structure

Homepage is configured with three tabs:

Tab	Service suffix	Description
External Secure	(plain names)	Services via `*.jeeves5454.ddns.net` or `*.jeevesconsults.ca`
Internal Secure	`(LAN)`	Services via `*.home.local` through Traefik
Internal Unsecured	`(IP)`	Services via direct `http://IP:port`

The default tab is set to **Internal Secure**.

Services can appear in multiple tabs if they have multiple access routes. Docker container status widgets (`server: centerpoint`, `container: <name>`) are applied to services where applicable.

Configuration Files

All config files live under `/home/jeeves/docker/homepage/config/`:

File	Purpose
`settings.yaml`	Dashboard title, theme, background, tab layout
`services.yaml`	All service entries grouped by tab and category

File	Purpose
`widgets.yaml`	Top-bar info widgets (system stats, weather, etc.)
`bookmarks.yaml`	Bookmark groups (if used)
`docker.yaml`	Docker socket connection config for container status

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/homepage/config`	`/app/config`	All dashboard config files
`/var/run/docker.sock`	`/var/run/docker.sock:ro`	Read-only Docker socket for container status

Dependencies

- Docker socket for live container status display
- Traefik on `traefik-net` for the `centerpoint.home.local` route
- API keys for service widgets (stored in `services.yaml` — secrets should be kept in a `secrets.yaml` file or environment variables, not committed to version control)

Notes / Gotchas

- Homepage requires group names to be **globally unique** across all tabs. The `(LAN)` and `(IP)` suffixes on group names solve this constraint for the multi-tab layout.
- `HOMEPAGE_ALLOWED_HOSTS` must include any hostname or IP:port used to access the dashboard; otherwise Homepage returns a 403.
- The Docker socket mount allows Homepage to show live container state. Homepage accesses it read-only and does not have the ability to start or stop containers.
- Config changes take effect immediately on file save — no container restart required.
- Quick Launch search only finds services listed in `services.yaml`. To enable search indexing for descriptions, set `searchDescriptions: true` in `settings.yaml`.

Last Updated: 2026-06-16