h05-media-management

- 00-chapter-intro.md
- 01-sonarr.md
- 02-radarr.md
- 03-prowlarr.md
- 04-bazarr.md
- 05-nzbget.md
- 06-whisparr.md
- 07-lazylibrarian.md
- 08-mylar3.md
- 09-audiobookrequest.md
- 10-profilarr.md
- 11-dispatcharr.md
- 12-pulsarr.md
- 13-flaresolverr.md

00-chapter-intro.md

kstack: book: Centerpoint Home Lab chapter: Media Management page: Chapter Introduction tags: [media-management, arr, sonarr, radarr, prowlarr, nzbget]

Overview

This chapter documents the automated media acquisition and management stack — commonly referred to as the *arr ecosystem. These services work together to monitor, search, download, rename, and organise media content automatically, then deliver it to Plex and Jellyfin.

Download Pipeline

```
Seerr (request) ──► Sonarr / Radarr / Whisparr / LazyLibrarian
                          |
                          ▼
                    Prowlarr (indexer search)
                          |
                          ▼
                    NZBGet (Usenet download)
                          |
                          ▼
          /media/jeeves/1TB_Vol2/downloads  (Ceph OSD — temp)
```

```
                    |
              Post-processing
                    |
                    ▼
/mnt/Multimedia  (NFS → UnRAID — permanent)
                    |
              Plex / Jellyfin
```

Downloads land temporarily on the Ceph OSD volume (`nvme2n1`, mounted at `/media/jeeves/1TB_Vol2`) for high-speed write performance. After post-processing, completed media is moved/hardlinked to the UnRAID NFS share at `/mnt/Multimedia` for permanent storage.

Shared Networks

All services in this chapter are on two Docker networks:

Network	Purpose
`traefik-net`	Web UI access via Traefik
`media-network`	Internal service-to-service communication (no Traefik hop)

`media-network` allows *arr services to reach each other and NZBGet directly by container hostname without going through Traefik.

Services in This Chapter

Service	Container	Status	Purpose
Sonarr	`sonarr`	Active	TV show monitoring and management
Radarr	`radarr`	Active	Movie monitoring and management
Prowlarr	`prowlarr`	Active	Indexer aggregator for all *arr apps
Bazarr	`bazarr`	Active	Subtitle management
NZBGet	`nzbget`	Active	Usenet download client
Whisparr	`whisparr`	Active	Adult content management (Sonarr fork)

Service	Container	Status	Purpose
LazyLibrarian	`lazylibrarian`	Active	Book and magazine management
Mylar3	`mylar3`	Active	Comics management
Audiobookrequest	`audiobookrequest`	Active	Audiobook request portal
Profilarr	`profilarr`	Active	Quality profile manager for *arr apps
Dispatcharr	`dispatcharr`	Active	IPTV channel and stream dispatcher
Pulsarr	`pulsarr`	Active	Watchlist automation and notifications
FlareSolverr	`flaresolverr`	Offline	Cloudflare bypass proxy for Prowlarr

---

Last Updated: 2026-06-16

01-sonarr.md

kstack: book: Centerpoint Home Lab chapter: Media Management page: Sonarr tags: [sonarr, tv, arr, media-management]

Overview

Sonarr monitors RSS feeds and indexers (via Prowlarr) for new TV show episodes, automatically searches for and downloads them via NZBGet, then renames and organises the files into the media library on UnRAID.

Access

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

Configuration

Image: `lscr.io/linuxserver/sonarr:latest` **Compose project:** `arr` stack (managed via Portainer)

Traefik Labels

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

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

traefik.http.services.sonarr-svc.loadbalancer.server.port: 8989
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/arr/Sonarr/config`	`/config`	Sonarr database, settings, and series index
`/media/jeeves/1TB_Vol2/downloads`	`/downloads`	NZBGet download staging (Ceph OSD)
`/mnt/Multimedia`	`/media`	Final media library (NFS from UnRAID)

Both `/downloads` and `/media` must be in the same container path namespace so Sonarr can hardlink completed downloads rather than copy them — essential for atomic moves between the download staging area and the media library.

Integration Points

Service	Connection Method	Purpose
Prowlarr	API (`http://prowlarr:9696`) via `media-network`	Indexer search
NZBGet	API (`http://nzbget:6789`) via `media-network`	Download client
Bazarr	API (Bazarr polls Sonarr)	Subtitle fetching

Service	Connection Method	Purpose
Pulsarr	Sonarr webhook / API	Watchlist sync
Seerr	API	Receives show requests

Notes / Gotchas

- Sonarr's root folder for TV series must point to `/media/TV` (or equivalent) inside the container, which maps to `/mnt/Multimedia/TV` on the host via NFS.
- Quality profiles are managed centrally via Profilarr and pushed to Sonarr via its API — do not manually edit quality profiles in Sonarr if Profilarr is active.
- API key is stored in `/config/config.xml` — required to configure Prowlarr, Bazarr, Seerr, and Pulsarr integrations.

Last Updated: 2026-06-16

02-radarr.md

kstack: book: Centerpoint Home Lab chapter: Media Management page: Radarr tags: [radarr, movies, arr, media-management]

Overview

Radarr is the movie counterpart to Sonarr. It monitors for new and existing movie releases, searches indexers via Prowlarr, downloads via NZBGet, and moves completed movies into the media library on UnRAID.

Access

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

Configuration

Image: `lscr.io/linuxserver/radarr:latest` **Compose project:** `arr` stack

Traefik Labels

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

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

traefik.http.services.radarr-svc.loadbalancer.server.port: 7878
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/arr/Radarr/config`	`/config`	Radarr database and settings
`/media/jeeves/1TB_Vol2/downloads`	`/downloads`	NZBGet download staging (Ceph OSD)
`/mnt/Multimedia`	`/media`	Final media library (NFS from UnRAID)

Integration Points

Service	Connection Method	Purpose
Prowlarr	API (`http://prowlarr:9696`) via `media-network`	Indexer search
NZBGet	API (`http://nzbget:6789`) via `media-network`	Download client
Bazarr	API (Bazarr polls Radarr)	Subtitle fetching
Pulsarr	Radarr webhook / API	Watchlist sync
Seerr	API	Receives movie requests

Notes / Gotchas

- Radarr root folder must be set to the path inside the container that maps to `/mnt/Multimedia/Movies` on the host.
- Hardlinking between `/downloads` and `/media` requires both paths to be on the same filesystem. The current setup has downloads on Ceph OSD and the final library on NFS — hardlinking is **not** possible across these. Radarr performs a copy+delete instead. This is slower but correct.
- Quality profiles are managed via Profilarr — avoid manual edits in Radarr UI.

Last Updated: 2026-06-16

03-prowlarr.md

kstack: book: Centerpoint Home Lab chapter: Media Management page: Prowlarr tags: [prowlarr, indexer, arr, media-management]

Overview

Prowlarr is the indexer aggregation layer for the *arr stack. It manages all Usenet and torrent indexer connections in one place and syncs them automatically to Sonarr, Radarr, Whisparr, and LazyLibrarian — eliminating the need to configure indexers individually in each application.

Access

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

No external route — internal management only.

Configuration

Image: `lscr.io/linuxserver/prowlarr:latest` **Compose project:** `arr` stack

Traefik Labels

```
traefik.enable: "true"
traefik.http.routers.prowlarr.rule: Host(`prowlarr.home.local`)
traefik.http.routers.prowlarr.entrypoints: websecure
traefik.http.routers.prowlarr.tls.certresolver: step-ca
traefik.http.services.prowlarr.loadbalancer.server.port: 9696
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/arr/Prowlarr/config`	`/config`	Indexer definitions and settings

Integration Points

Prowlarr pushes its configured indexers to downstream *arr apps via their APIs. Each app is added in Prowlarr Settings → Apps:

App	API Endpoint (`media-network`)
Sonarr	`http://sonarr:8989`
Radarr	`http://radarr:7878`
Whisparr	`http://whisparr:6969`
LazyLibrarian	`http://lazylibrarian:5299`

When an indexer is added or updated in Prowlarr, it is automatically propagated to all connected apps. No manual indexer configuration in individual *arr apps is needed.

Notes / Gotchas

- FlareSolverr (currently offline) is used by Prowlarr to bypass Cloudflare-protected indexers. When FlareSolverr is down, any Cloudflare-protected indexer will fail. Configure the FlareSolverr proxy URL in Prowlarr Settings → Indexers → Proxies.
- Prowlarr's API key is required when adding Prowlarr as the indexer source in each *arr application. Find it in Prowlarr Settings → General → Security.
- Indexer categories must be correctly mapped in Prowlarr for each app (e.g. TV categories for Sonarr, movie categories for Radarr) — misconfigured categories result in wrong content being returned in searches.

Last Updated: 2026-06-16

04-bazarr.md

kstack: book: Centerpoint Home Lab chapter: Media Management page: Bazarr tags: [bazarr, subtitles, arr, media-management]

Overview

Bazarr automatically downloads subtitles for movies and TV shows managed by Radarr and Sonarr. It monitors both libraries and fetches subtitles from configured providers (OpenSubtitles, Subscene, etc.) whenever a new file is added or an existing file is missing subtitles.

Access

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

No external route — internal management only.

Configuration

Image: `lscr.io/linuxserver/bazarr:latest` **Compose project:** `arr` stack

Traefik Labels

```
traefik.enable: "true"
traefik.http.routers.bazarr.rule: Host(`bazarr.home.local`)
traefik.http.routers.bazarr.entrypoints: websecure
traefik.http.routers.bazarr.tls.certresolver: step-ca
traefik.http.services.bazarr.loadbalancer.server.port: 6767
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/arr/Bazarr/config`	`/config`	Bazarr database and settings
`/mnt/Multimedia`	`/media`	Media library for subtitle placement (NFS from UnRAID)

The `/media` mount mirrors the Sonarr/Radarr mount so Bazarr can write subtitle files directly alongside the video files in the library.

Integration Points

Service	Connection	Purpose
Sonarr	API (`http://sonarr:8989`) via `media-network`	TV library sync
Radarr	API (`http://radarr:7878`) via `media-network`	Movie library sync

Notes / Gotchas

- Subtitle provider API keys (OpenSubtitles, etc.) are stored in Bazarr's settings. These are rate-limited — avoid aggressive subtitle searches on large libraries.
- Bazarr writes subtitle `.srt` or `.ass` files next to the video files in `/mnt/Multimedia`. Plex and Jellyfin pick these up automatically on the next library scan.
- If Sonarr or Radarr is restarted, Bazarr may take a few minutes to re-sync its library view via the API.

Last Updated: 2026-06-16

05-nzbget.md

kstack: book: Centerpoint Home Lab chapter: Media Management page: NZBGet tags: [nzbget, usenet, download-client, arr, media-management]

Overview

NZBGet is the Usenet download client for the homelab. It receives download jobs from Sonarr, Radarr, Whisparr, and LazyLibrarian, downloads from configured Usenet providers, unpacks archives, and notifies the requesting *arr application when complete.

Downloads are staged to the Ceph OSD volume (`1TB_Vol2`) for high-speed local writes before being processed and moved to UnRAID.

Access

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

Configuration

Image: `lscr.io/linuxserver/nzbget:latest` **Compose project:** `arr` stack

Traefik Labels

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

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

traefik.http.services.nzbget-svc.loadbalancer.server.port: 6789
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/arr/NZBGet/config`	`/config`	NZBGet settings and queue database
`/media/jeeves/1TB_Vol2/downloads`	`/downloads`	Download staging directory (Ceph OSD)
`/mnt/Multimedia`	`/media`	Media library (used by post-process scripts)

Download Directory Layout

NZBGet organises downloads under `/downloads` with category subdirectories:

```
/media/jeeves/1TB_Vol2/downloads/
├── usenet/
│   ├── intermediate/    ← In-progress downloads
```

```
│   └── complete/
│       ├── tv/          ← Completed TV episodes → Sonarr picks up
│       ├── movies/      ← Completed movies → Radarr picks up
│       ├── books/       ← Completed books → LazyLibrarian picks up
│       └── adult/       ← Completed adult content → Whisparr picks up
```

Notes / Gotchas

- Usenet provider credentials (server address, port, username, password) are stored in NZBGet's settings — these are **REDACTED** in all documentation.
- NZBGet's built-in web UI has its own username/password in addition to the Authentik ForwardAuth layer on the external route. Both must be configured.
- The Ceph OSD download volume provides fast local NVMe performance for downloads before they are moved to the slower NFS path. Ensure adequate free space is maintained on `1TB_Vol2`.
- Each *arr app connects to NZBGet via `http://nzbget:6789` on the `media-network` using the NZBGet API credentials configured in the *arr download client settings.

Last Updated: 2026-06-16

06-whisparr.md

kstack: book: Centerpoint Home Lab chapter: Media Management page: Whisparr tags: [whisparr, arr, media-management, adult-content]

Overview

Whisparr is a Sonarr fork purpose-built for managing adult content. It integrates with Prowlarr for indexer searches and NZBGet for downloads, following the same *arr workflow as Sonarr but with metadata sources appropriate for its content type. Downloaded content is delivered to the same media volume that Stash monitors.

Access

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

No external route — internal management only.

Configuration

Image: `ghcr.io/hotio/whisparr:v3` **Compose project:** `arr` stack

Traefik Labels

```
traefik.enable: "true"
traefik.http.routers.whisparr.rule: Host(`whisparr.home.local`)
traefik.http.routers.whisparr.entrypoints: websecure
traefik.http.routers.whisparr.tls.certresolver: step-ca
traefik.http.services.whisparr.loadbalancer.server.port: 6969
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/arr/Whisparr/config`	`/config`	Whisparr database and settings
`/media/jeeves/1TB_Vol2/downloads`	`/downloads`	NZBGet download staging (Ceph OSD)
`/mnt/Multimedia`	`/data`	Media library including adult content (NFS from UnRAID)

Integration Points

Service	Connection Method	Purpose
Prowlarr	API (`http://prowlarr:9696`) via `media-network`	Indexer search
NZBGet	API (`http://nzbget:6789`) via `media-network`	Download client

Notes / Gotchas

- The `hotio/whisparr:v3` image is a community-maintained build. Monitor the repository for updates as this is not an official Servarr project.
- Whisparr's library root folder should point to the same path that Stash monitors (`/mnt/Multimedia/ST`) so content is automatically picked up by Stash after import.
- Prowlarr indexer categories for adult content must be configured correctly in Prowlarr and mapped to Whisparr in the Apps settings.

Last Updated: 2026-06-16

07-lazylibrarian.md

kstack: book: Centerpoint Home Lab chapter: Media Management page: LazyLibrarian tags: [lazylibrarian, books, ebooks, magazines, arr, media-management]

Overview

LazyLibrarian manages ebook, audiobook, and magazine acquisition. It monitors authors and publications, searches for releases via Prowlarr, downloads via NZBGet, and organises the completed files into the books library on UnRAID. Despite its internal name, it is accessed externally via the `readarr.jeevesconsults.ca` domain.

Access

Type	URL	Notes
Internal	`https://readarr.home.local`	LAN access via Step-CA TLS
External	`https://readarr.jeevesconsults.ca`	Authentik SSO + GeoBlock + CrowdSec

> The external domain uses `readarr.jeevesconsults.ca` (not `lazylibrarian.*`).

Configuration

Image: `lscr.io/linuxserver/lazylibrarian:latest` **Compose project:** `arr` stack

Traefik Labels

```
# External route
traefik.http.routers.lazylibrarian-external.rule: Host(`readarr.jeevesconsults.ca`)
traefik.http.routers.lazylibrarian-external.entrypoints: websecure
traefik.http.routers.lazylibrarian-external.tls.certresolver: letsencrypt
traefik.http.routers.lazylibrarian-external.middlewares: authentik-auth@docker,plex-
geoblock@file,crowdsec-bouncer@file
traefik.http.routers.lazylibrarian-external.service: lazylibrarian-svc

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

traefik.http.services.lazylibrarian-svc.loadbalancer.server.port: 5299
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/arr/lazylibrarian/config`	`/config`	LazyLibrarian database and settings
`/media/jeeves/1TB_Vol2/downloads/usenet/complete/books`	`/downloads`	Completed book downloads (Ceph OSD)
`/mnt/Multimedia/Books/Lazylibrarian`	`/books`	Final book library (NFS from UnRAID)

Integration Points

Service	Connection Method	Purpose
Prowlarr	API (`http://prowlarr:9696`) via `media-network`	Indexer search
NZBGet	API (`http://nzbget:6789`) via `media-network`	Download client

Notes / Gotchas

- LazyLibrarian covers ebooks, audiobook metadata management (separate from Audiobookshelf's podcast/streaming role), and magazines. Configure each content type under its own library section within LazyLibrarian.
- The external domain (`readarr.jeevesconsults.ca`) was chosen for cleaner branding — update DNS and Traefik labels if the domain ever changes.
- Calibre integration can be enabled in LazyLibrarian settings if a Calibre server is available on the network for ebook format conversion.

Last Updated: 2026-06-16

08-mylar3.md

kstack: book: Centerpoint Home Lab chapter: Media Management page: Mylar3 tags: [mylar3, comics, cbz, arr, media-management]

Overview

Mylar3 is the automated comic book downloader and manager for the homelab. It monitors comic series, searches for new issues via Usenet indexers (through NZBGet), and organises downloaded CBZ/CBR files into the comics library on UnRAID.

Access

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

No external route — internal management only.

Configuration

Image: `lscr.io/linuxserver/mylar3:latest` **Compose project:** `arr` stack

Traefik Labels

```
traefik.enable: "true"
traefik.http.routers.mylar3.rule: Host(`mylar.home.local`)
traefik.http.routers.mylar3.entrypoints: websecure
traefik.http.routers.mylar3.tls.certresolver: step-ca
traefik.http.services.mylar3.loadbalancer.server.port: 8090
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/arr/mylar/config`	`/config`	Mylar3 database and settings
`/media/jeeves/1TB_Vol2/downloads/usenet/complete/books`	`/downloads`	Completed downloads (Ceph OSD, shared with books)
`/mnt/Multimedia/Books/Comics`	`/comics`	Comics library (NFS from UnRAID)

Integration Points

Service	Connection Method	Purpose
NZBGet	API (`http://nzbget:6789`) via `media-network`	Download client

> Mylar3 connects directly to NZBGet rather than through Prowlarr. Indexers are configured directly in Mylar3 settings.

Notes / Gotchas

- Mylar3 uses ComicVine as its metadata source — a free API key from `comicvine.gamespot.com` is required and stored in the Mylar3 config.
- The `/downloads` path is shared with LazyLibrarian (both point to the books completion directory). Ensure Mylar3's category in NZBGet is distinct to avoid processing conflicts.
- Mylar3's database (`mylar.db`) stores the full comic watchlist and download history. Back it up before upgrades.

Last Updated: 2026-06-16

09-audiobookrequest.md

kstack: book: Centerpoint Home Lab chapter: Media Management page: Audiobookrequest tags: [audiobookrequest, audiobooks, requests, media-management]

Overview

Audiobookrequest is a self-hosted audiobook request portal — similar in concept to Seerr but for audiobooks. Users can search for and request audiobooks, which are then queued for download via the configured acquisition tools (LazyLibrarian).

Access

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

No external route — LAN and Tailscale access only.

Configuration

Image: `markbeep/audiobookrequest:1` **Compose project:** `arr` stack

Traefik Labels

```
traefik.enable: "true"
traefik.http.routers.audiobookrequest.rule: Host(`audiobookseer.home.local`)
traefik.http.routers.audiobookrequest.entrypoints: websecure
traefik.http.routers.audiobookrequest.tls.certresolver: step-ca
traefik.http.services.audiobookrequest.loadbalancer.server.port: 8000
```

Key Environment Variables

Variable	Value	Purpose
`TZ`	`America/Toronto`	Timezone
`ABR_APP__PORT`	`8000`	Application listen port
`ABR_APP__FORCE_LOGIN_TYPE`	`forms`	Forces form-based login
`ABR_APP__VERSION`	`1.10.5`	Application version

`ABR_APP__FORCE_LOGIN_TYPE=forms` disables any SSO/header-based login and enforces the standard username/password login form.

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/arr/audiobookrequest/config`	`/config`	Request database and settings

Notes / Gotchas

- User accounts and request history are stored in the `/config` bind mount.
- Audiobookrequest is pinned to image tag `1` — check for newer tagged releases before upgrading.
- Integration with LazyLibrarian (for fulfilling requests) must be configured in Audiobookrequest settings with the LazyLibrarian API endpoint and key.

Last Updated: 2026-06-16

10-profilarr.md

kstack: book: Centerpoint Home Lab chapter: Media Management page: Profilarr tags: [profilarr, quality-profiles, arr, media-management]

Overview

Profilarr is a quality profile and custom format manager for the *arr stack. It maintains a centralised set of quality profiles and custom format definitions and syncs them across Sonarr, Radarr, and other *arr apps via their APIs — ensuring consistent quality standards across all services without manual duplication.

Access

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

No external route — internal management only.

Configuration

Image: `santiagosayshey/profilarr:latest` **Compose project:** `arr` stack

Traefik Labels

```
traefik.enable: "true"
traefik.http.routers.profilarr.rule: Host(`profilarr.home.local`)
traefik.http.routers.profilarr.entrypoints: websecure
traefik.http.routers.profilarr.tls.certresolver: step-ca
traefik.http.services.profilarr.loadbalancer.server.port: 6868
```

Environment Variables

Variable	Value	Purpose
`TZ`	`America/Toronto`	Timezone

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/arr/profilarr/config`	`/config`	Profile definitions and sync state

Notes / Gotchas

- Quality profiles in Sonarr and Radarr should not be edited manually when Profilarr is managing them — changes will be overwritten on the next sync.
- Profilarr can import community-maintained profile packs (e.g. TRaSH Guides profiles) and push them to all connected *arr apps.
- API keys for each connected *arr application must be configured in Profilarr settings.

Last Updated: 2026-06-16

11-dispatcharr.md

kstack: book: Centerpoint Home Lab chapter: Media Management page: Dispatcharr tags: [dispatcharr, iptv, m3u, media-management]

Overview

Dispatcharr is an IPTV playlist and stream management tool. It organises M3U channel lists, manages IPTV stream sources, and acts as a companion to Threadfin — providing a more structured interface for managing channel groups, EPG mappings, and stream prioritisation before they are published to Plex and Jellyfin via Threadfin.

Dispatcharr runs in **all-in-one (`aio`) mode**, bundling its application server, Celery task worker, and Redis instance into a single container.

Access

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

No external route — internal management only.

Configuration

Image: `ghcr.io/dispatcharr/dispatcharr:latest` **Compose project:** `arr` stack

Traefik Labels

```
traefik.enable: "true"
traefik.http.routers.dispatcharr.rule: Host(`dispatcharr.home.local`)
traefik.http.routers.dispatcharr.entrypoints: websecure
traefik.http.routers.dispatcharr.tls.certresolver: step-ca
traefik.http.services.dispatcharr.loadbalancer.server.port: 9191
```

Key Environment Variables

Variable	Value	Purpose
`DISPATCHARR_ENV`	`aio`	All-in-one mode (app + worker + Redis)
`DISPATCHARR_LOG_LEVEL`	`info`	Logging verbosity
`REDIS_HOST`	`localhost`	Internal Redis (bundled in AIO mode)
`CELERY_BROKER_URL`	`redis://localhost:6379/0`	Celery task queue

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/arr/dispatcharr/data`	`/data`	Dispatcharr database and config

Notes / Gotchas

- In `aio` mode, Redis runs inside the container and does not persist state between restarts beyond what is stored in the `/data` bind mount. Task queue state is lost on container restart.
- Dispatcharr integrates with Threadfin as the downstream consumer of its managed M3U playlists. Configure the Dispatcharr output URL in Threadfin as the M3U source.

- `ghcr.io/dispatcharr/dispatcharr:latest` is an actively developed project — check release notes before pulling updates as breaking changes may affect the channel database schema.

Last Updated: 2026-06-16

12-pulsarr.md

kstack: book: Centerpoint Home Lab chapter: Media Management page: Pulsarr tags: [pulsarr, watchlist, notifications, arr, media-management]

Overview

Pulsarr is a watchlist automation and notification bridge for the *arr stack. It monitors Plex and Jellyfin watchlists, automatically adds requested movies and TV shows to Radarr and Sonarr, and sends notifications when content is downloaded and available. It also integrates with TMDB for metadata-enriched notifications.

Access

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

Configuration

Image: `lakker/pulsarr:latest` **Compose project:** `arr` stack

Traefik Labels

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

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

traefik.http.services.pulsarr-svc.loadbalancer.server.port: 3003
```

Key Environment Variables

Variable	Value	Purpose
`TZ`	`America/Toronto`	Timezone
`port`	`3003`	Application listen port
`tmdbApiKey`	**REDACTED**	TMDB API key for metadata and notification enrichment

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/arr/pulsarr/config`	`/app/data`	Pulsarr database and configuration

Integration Points

Service	Purpose
Sonarr	Adds TV shows from watchlists; receives completion events
Radarr	Adds movies from watchlists; receives completion events
Plex	Reads watchlists from Plex accounts
TMDB	Fetches poster art and metadata for notifications

Notes / Gotchas

- The TMDB API key (`tmdbApiKey`) is visible in the container environment — treat it as a secret. Rotate at `themoviedb.org` if exposed.
- Pulsarr's watchlist sync requires Plex account API tokens configured in the Pulsarr UI settings.
- Pulsarr is healthy-checked (`(healthy)` status in `docker ps`) — the healthcheck endpoint is configured internally.
- Notification channels (Discord, Telegram, etc.) are configured in Pulsarr settings and stored in the `/app/data` bind mount.

Last Updated: 2026-06-16

13-flaresolverr.md

kstack: book: Centerpoint Home Lab chapter: Media Management page: FlareSolverr (Offline) tags: [flaresolverr, cloudflare, prowlarr, offline, media-management]

Overview

> **Status: OFFLINE** — The FlareSolverr container is present but currently not running (exited 8 weeks ago). It can be restarted when Cloudflare-protected indexers need to be accessed via Prowlarr.

FlareSolverr is a proxy server that bypasses Cloudflare's anti-bot protection for web-based indexers. Prowlarr routes requests for Cloudflare-protected indexers through FlareSolverr, which uses a headless browser session to solve the challenge and return the page content.

Access

FlareSolverr has no web UI. It is accessed internally by Prowlarr via its HTTP API at `http://flaresolverr:8191`.

Configuration

Image: `ghcr.io/flaresolverr/flaresolverr:latest` **Status:** `exited`

Prowlarr Integration

Once running, configure in Prowlarr:

1. Settings → Indexers → Add Proxy
2. Type: **FlareSolverr**
3. Host: `http://flaresolverr:8191`
4. Tag: Apply the tag to any Cloudflare-protected indexer

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/flaresolver`	`/config`	FlareSolverr config

Notes / Gotchas

- FlareSolverr uses a headless Chromium browser. It consumes significant CPU and RAM during challenge solving — resource spikes are normal.
- Cloudflare regularly updates its bot detection. FlareSolverr may stop working after Cloudflare updates and require an image update.
- When FlareSolverr is offline, any Prowlarr indexer tagged with the FlareSolverr proxy will fail silently on search — no error surfaced in Sonarr/Radarr. Check Prowlarr indexer status if searches return no results.
- To restart: `docker start flaresolverr` or redeploy via Portainer.

Last Updated: 2026-06-16