

3. Docker Compose File

Okey then! So you have created an NFS share, created the Trash Guides folder structure inside of that NFS share then now it is time to use that to setup the docker compose file. The compose file comes from different websites and it is surely not the only way to do this. However, it is a simple one that creates one docker network called **internal proxy** and uses **gluetun** as a VPN container. Furthermore, we will use the open ports 4001 - 4012 for the applications that we are running. You can keep the original official port numbers but I like to keep it structured instead of random port numbers. With all of that said the compose file looks like this

compose.yml

```
services:
# Sonarr - TV Shows
  sonarr:
    image: lscr.io/linuxserver/sonarr:latest
    container_name: sonarr
    environment:
      - PUID=${PUID}
      - PGID=${PGID}
      - TZ=${TZ}
    volumes:
      - ./sonarr/config:/config
      - ${ROOT_MEDIA_PATH}:/data # Access to /data/downloads and /data/media/tv
    ports:
      - 4001:8989
    restart: unless-stopped
    networks:
      - internal-proxy

# Radarr - Movies
  radarr:
    image: lscr.io/linuxserver/radarr:latest
    container_name: radarr
    environment:
      - PUID=${PUID}
      - PGID=${PGID}
```

```
- TZ=${TZ}
volumes:
- ./radarr/config:/config
- ${ROOT_MEDIA_PATH}:/data # Access to /data/downloads and /data/media/movies
ports:
- 4002:7878
restart: unless-stopped
networks:
- internal-proxy
```

Prowlarr - Indexer centralized management

```
prowlarr:
  image: lscr.io/linuxserver/prowlarr:latest
  container_name: prowlarr
  environment:
    - PUID=${PUID}
    - PGID=${PGID}
    - TZ=${TZ}
  volumes:
    - ./prowlarr/config:/config
  ports:
    - 4003:9696
  restart: unless-stopped
  networks:
    - internal-proxy
```

Profilarr

```
profilarr:
  image: ghcr.io/dictionary-hub/profilarr:latest
  container_name: profilarr
  restart: unless-stopped
  ports:
    - "4004:6868"
  volumes:
    - ./config:/config
  environment:
    - PUID=${PUID}
    - PGID=${PGID}
    - UMASK=022
```

- TZ=\${TZ}

depends_on:

- radarr
- sonarr

FlareSolvrr - Bypasses Cloudflare protection for Indexers

flaresolvrr:

image: ghcr.io/flaresolvrr/flaresolvrr:latest

container_name: flaresolvrr

environment:

- LOG_LEVEL=\${LOG_LEVEL:-info}
- TZ=\${TZ}

ports:

- 4005:8191

restart: unless-stopped

networks:

- internal-proxy

bazarr for subtitles

bazarr:

image: lscr.io/linuxserver/bazarr:latest

container_name: bazarr

environment:

- PUID=\${PUID}
- PGID=\${PGID}
- TZ=\${TZ}

volumes:

- ./bazarr/config:/config
- \${ROOT_MEDIA_PATH}:/data

ports:

- 4006:6767

restart: unless-stopped

networks:

- internal-proxy

maintainerr:

image: ghcr.io/maintainerr/maintainerr:latest # or

ghcr.io/maintainerr/maintainerr:development (to test unreleased changes)

container_name: maintainerr

```
user: 1000:1000
volumes:
  - type: bind
    source: /mnt/data
    target: /opt/data
#       - type: bind # uncomment for the leftover-folder cleanup: your library, at the
same path Radarr/Sonarr report it at
#       source: /path/to/media
#       target: /path/to/media
environment:
  - TZ=Europe/Amsterdam
#       - BASE_PATH=/maintainerr # uncomment if you're serving maintainerr from a
subdirectory
#       - UI_HOSTNAME=: # uncomment if you want to listen on IPv6 instead (default
0.0.0.0)
#       - UI_PORT=4007 # uncomment to change the UI port (default 6246)
#       - GITHUB_TOKEN=ghp_yourtoken # Optional: GitHub Personal Access Token for
higher API rate limits (60/hr without, 5000/hr with token)
ports:
  - 4007:6246
restart: unless-stopped
healthcheck: # already baked into the image; included here so you can tune it
  test: ['CMD', '/opt/app/healthcheck.sh']
  interval: 30s
  timeout: 5s
  start_period: 40s
  retries: 3
networks:
  - internal-proxy

seerr:
  image: ghcr.io/seerr-team/seerr:latest
  init: true
  container_name: seerr
  environment:
    - PUID=${PUID}
    - PGID=${PGID}
    - LOG_LEVEL=debug
    - TZ=${TZ}
```

- PORT=5055 #optional

ports:

- 4011:5055

volumes:

- ./seerrconfig:/app/config

healthcheck:

test: wget --no-verbose --tries=1 --spider

http://10.69.1.103:4011/api/v1/settings/public || exit 1

start_period: 20s

timeout: 3s

interval: 15s

retries: 3

restart: unless-stopped

networks:

- internal-proxy

qbittorrent:

image: lscr.io/linuxserver/qbittorrent

container_name: qbittorrent

network_mode: "service:gluetun" # Routes all traffic through Gluetun

depends_on:

gluetun:

condition: service_healthy # Wait for VPN to be up first

environment:

- PUID=1000

- PGID=1000

- TZ=\${TZ}

- WEBUI_PORT=4009

- WEBUI_ADDRESS=0.0.0.0

- WEBUI_EXTERNAL_ACCESS=true

volumes:

- ./qbittorrent:/config

- \${ROOT_MEDIA_PATH}:/data

restart: unless-stopped

nzbget:

image: lscr.io/linuxserver/nzbget:latest

container_name: nzbget

```
environment:
  - PUID=${PUID}
  - PGID=${PGID}
  - TZ=${TZ}
volumes:
  - /etc/localtime:/etc/localtime:ro
  - ./nzbget:/config
  - /mnt/data:/data
ports:
  - 4008:6789
restart: unless-stopped
networks:
  - internal-proxy
```

jellydash:

```
image: ghcr.io/themartz90/jellydash:latest
restart: unless-stopped
ports:
  - "${APP_PORT:-8080}:80"
environment:
  PUID: "${PUID}"
  PGID: "${PGID}"
  TZ: "${TZ}"
  APP_ENV: "${APP_ENV:-production}"
  APP_DEBUG: "${APP_DEBUG:-false}"
  APP_TIMEZONE: "${APP_TIMEZONE:-UTC}"
  # Bundled MariaDB below. Using your own database server instead? Point
  # DB_HOST at it and drop the db service in docker-compose.override.yml.
  DB_HOST: "${DB_HOST:-db}"
  DB_PORT: "${DB_PORT:-3306}"
  DB_DRIVER: mysql
  DB_NAME: "${DB_NAME:-jellydash}"
  DB_USER: "${DB_USER:-jellydash}"
  DB_PASS: "${DB_PASS:-change-me}"
  JELLYFIN_URL: "${JELLYFIN_URL}"
  JELLYFIN_API_TOKEN: "${JELLYFIN_API_TOKEN}"
  JELLYFIN_VERIFY_SSL: "${JELLYFIN_VERIFY_SSL:-true}"
  LIBRARIES_CACHE_TTL: "${LIBRARIES_CACHE_TTL:-300}"
  # Jellyseerr (optional; the page hides until these are set)
```

```
JELLYSEER_URL: "${JELLYSEER_URL:-}"
JELLYSEER_API_TOKEN: "${JELLYSEER_API_TOKEN:-}"
JELLYSEER_VERIFY_SSL: "${JELLYSEER_VERIFY_SSL:-true}"
SEERR_POLL_INTERVAL: "${SEERR_POLL_INTERVAL:-120}"
SEERR_NOTIFY_ENABLED: "${SEERR_NOTIFY_ENABLED:-true}"
# Background poller (history recording, cache warming, alerts)
POLLER_ENABLED: "${POLLER_ENABLED:-true}"
POLL_INTERVAL: "${POLL_INTERVAL:-30}"
# Web Push notifications (optional; generate keys once, see README)
PUSH_ENABLED: "${PUSH_ENABLED:-true}"
VAPID_PUBLIC_KEY: "${VAPID_PUBLIC_KEY:-}"
VAPID_PRIVATE_KEY: "${VAPID_PRIVATE_KEY:-}"
VAPID_SUBJECT: "${VAPID_SUBJECT:-mailto:admin@example.com}"
PUSH_IGNORE_USERS: "${PUSH_IGNORE_USERS:-}"
# Public base URL (optional): external alert links point back here.
APP_URL: "${APP_URL:-}"
# Extra notification channels; each is active once its config is set.
TELEGRAM_BOT_TOKEN: "${TELEGRAM_BOT_TOKEN:-}"
TELEGRAM_CHAT_ID: "${TELEGRAM_CHAT_ID:-}"
PUSHOVER_APP_TOKEN: "${PUSHOVER_APP_TOKEN:-}"
PUSHOVER_USER_KEY: "${PUSHOVER_USER_KEY:-}"
DISCORD_WEBHOOK_URL: "${DISCORD_WEBHOOK_URL:-}"
# Libraries hidden from Trending/Most Watched (also editable in Settings)
TRENDING_EXCLUDE_LIBRARIES: "${TRENDING_EXCLUDE_LIBRARIES:-}"
# Optional login gate (recommended when internet-facing)
AUTH_ENABLED: "${AUTH_ENABLED:-false}"
AUTH_ADMIN_USER: "${AUTH_ADMIN_USER:-}"
AUTH_ADMIN_PASSWORD: "${AUTH_ADMIN_PASSWORD:-}"
depends_on:
  db:
    condition: service_healthy
volumes:
  - ./jellydash_cache:/var/www/html/cache
  - ./jellydash_runtime_cache:/var/www/html/var/cache
  - ./jellydash_logs:/var/www/html/var/log
  - ./jellydash_uploads:/var/www/html/public/uploads
# Drop-in feature modules (see docs/MODULES.md), e.g.:
# - ./my-modules/downloads:/var/www/html/modules/downloads:ro
networks:
```

- internal-proxy

db:

image: mariadb:11

restart: unless-stopped

environment:

MARIADB_RANDOM_ROOT_PASSWORD: "yes"

MARIADB_DATABASE: "\${DB_NAME:-jellydash}"

MARIADB_USER: "\${DB_USER:-jellydash}"

MARIADB_PASSWORD: "\${DB_PASS:-change-me}"

PUID: "\${PUID}"

PGID: "\${PGID}"

TZ: "\${TZ}"

healthcheck:

test: ["CMD", "healthcheck.sh", "--connect", "--innodb_initialized"]

interval: 5s

timeout: 5s

retries: 12

volumes:

- ./jellydash_data:/var/lib/mysql

networks:

- internal-proxy

gluetun:

image: qmcgaw/glutun:latest

container_name: glutun

cap_add:

- NET_ADMIN

environment:

- PUID=\${PUID}

- PGID=\${PGID}

- TZ=\${TZ}

- VPN_SERVICE_PROVIDER=\${VPN_SERVICE_PROVIDER}

- VPN_TYPE=\${VPN_TYPE}

- WIREGUARD_PRIVATE_KEY=\${WIREGUARD_PRIVATE_KEY}

- WIREGUARD_ADDRESSES=\${WIREGUARD_ADDRESSES}

- VPN_PORT_FORWARDING=\${VPN_PORT_FORWARDING}

ports:

- 8888:8888 # Health check and control

```
- 4009:4009
# - 4009:6881 # qBittorrent P2P port
# - 4009:6881/udp
restart: unless-stopped
networks:
  - internal-proxy

networks:
  internal-proxy:
    driver: bridge
```

and the associated environment file looks like this

.env

```
# User and Group IDs for Permissions
PUID=1000
PGID=1000
TZ=Europe/Amsterdam

# Global Paths for Download Client and *Arr Apps
DOWNLOADS_PATH=/mnt/data
MEDIA_PATH=/mnt/data
ROOT_MEDIA_PATH=/mnt/data

# Network Settings
PROXY_NETWORK=internal-proxy

VPN_SERVICE_PROVIDER=protonvpn
VPN_TYPE=wireguard
WIREGUARD_PRIVATE_KEY=private-key
WIREGUARD_ADDRESSES=x.x.x.x/32
VPN_PORT_FORWARDING=yes

# Jellydash configuration.
# -----
```

```
# Basics
# -----

# Leave as production. The local mode is only for development.
APP_ENV=production
APP_DEBUG=false

# The port Jellydash runs on, so http://your-host:8080
APP_PORT=4010

# Your timezone (IANA name, e.g. Europe/Prague). Used for timestamps
# and the daily statistics boundaries.
APP_TIMEZONE=Europe/Amsterdam

# Public address of your dashboard. Optional, can stay empty.
# When set, notification alerts include a link back to the app.
APP_URL=

#Database

DB_HOST=db
DB_PORT=3306
DB_NAME=jellydash
DB_USER=jellydash
# Required. Pick any password, the bundled database is created with it.
DB_PASS=9u2983fjosdjhf9823jf9jisodf

# -----
# Jellyfin (required)
# -----

# Required. The address of your Jellyfin server.
JELLYFIN_URL=https://jellyfin.example.org

# Required. An API key from Jellyfin > Dashboard > API Keys. Use an admin
# key, some statistics need to read library paths.
JELLYFIN_API_TOKEN=jellyfin-api-token

# Set false only when your Jellyfin runs on self-signed HTTPS.
```

```
JELLYFIN_VERIFY_SSL=true
```

```
# How long the Libraries page cache lives, in seconds. The default is fine.
```

```
LIBRARIES_CACHE_TTL=300
```

```
# -----
```

```
# Jellyseerr (optional, can stay empty)
```

```
# -----
```

```
# Fill these and the Jellyseerr page appears in the menu, including alerts
```

```
# for new requests. API key: Jellyseerr > Settings > General > API Key.
```

```
JELLYSEER_URL=
```

```
JELLYSEER_API_TOKEN=
```

```
JELLYSEER_VERIFY_SSL=true
```

```
# Seconds between request syncs, and whether new requests send an alert.
```

```
SEERR_POLL_INTERVAL=120
```

```
SEERR_NOTIFY_ENABLED=true
```

```
# -----
```

```
# Background poller
```

```
# -----
```

```
# Records play history even when nobody has the dashboard open, and sends
```

```
# the notifications. You want this on.
```

```
POLLER_ENABLED=true
```

```
# Seconds between Jellyfin checks. 30 is a good balance.
```

```
POLL_INTERVAL=30
```

```
# -----
```

```
# Notifications (all optional, can stay empty)
```

```
# -----
```

```
# Alerts when someone starts playing and when a new Jellyseerr request
```

```
# comes in. A channel is on as soon as its values are filled. Setup steps
```

```
# for every channel are in the README.
```

```
# Master switch for all notification channels.
```

```
PUSH_ENABLED=true
```

```
# Jellyfin usernames that never trigger an alert, comma-separated.
# Typically your own. Also editable later in Settings inside the app.
PUSH_IGNORE_USERS=

# -----
# Login (optional)
# -----
# Off by default. Turn it on when the dashboard is reachable from the
# internet. The admin user is created automatically on the next start.

AUTH_ENABLED=false
AUTH_ADMIN_USER=jkodfk
# The password needs at least 8 characters, shorter ones are rejected.
AUTH_ADMIN_PASSWORD=tJT9off2KfCN2Ug0x8CI

# -----
# Statistics
# -----

# Jellyfin libraries to hide from Trending and Most Watched, comma-separated
# names. Also editable later in Settings inside the app.
TRENDING_EXCLUDE_LIBRARIES=
```

I keep every tunable value in a `.env` file rather than hardcoding it in the Compose file. Ports, paths, and the user IDs all live in one place, which makes the Compose file readable and easy to reuse. I also run every service on an **external** network called `media_network`, which means the network is created once, outside of Compose, and the stack attaches to it. That keeps the network stable even when I tear the stack down and bring it back up.

Sweet! So you want to install the arr stack on your self hosted device. Well, as with any app you need decide which app workflow deployment you want to use. Here, we focus on the docker compose deployment using a dedicated VM. Now, personally I like to run the arr stack in a dedicated VM and Jellyfin in another Docker VM. This gives you the possibility to move your Jellyfin

instance around easily to different hosts and if something is wrong with your arr stack then your users can still enjoy the already downloaded media. Eventually, it is up to you and the amount of resources you have at your disposal what your decision will be.

1. Create a new docker vm by using the proxmox helper scripts ;) Or if you want to continue using your existing docker vm then you need

2.

Create a project directory and the `.env` file:

Hottio Decluttar

Whisparr is the Sonarr/Radarr equivalent for adult content

Whisper-AI generates subtitles on the fly for your media

compose.yml

```
services:
# Sonarr - TV Shows
  sonarr:
    image: lscr.io/linuxserver/sonarr:latest
    container_name: sonarr
    environment:
      - PUID=${PUID}
      - PGID=${PGID}
      - TZ=${TZ}
    volumes:
      - ./sonarr/config:/config
      - ${ROOT_MEDIA_PATH}:/data # Access to /data/downloads and /data/media/tv
    ports:
      - 8989:8989
    restart: unless-stopped
    networks:
      - internal-proxy

# Radarr - Movies
```

```
radarr:
  image: lscr.io/linuxserver/radarr:latest
  container_name: radarr
  environment:
    - PUID=${PUID}
    - PGID=${PGID}
    - TZ=${TZ}
  volumes:
    - ./radarr/config:/config
    - ${ROOT_MEDIA_PATH}:/data # Access to /data/downloads and /data/media/movies
  ports:
    - 7878:7878
  restart: unless-stopped
  networks:
    - internal-proxy
```

```
networks:
  internal-proxy:
    driver: bridge
```

Prowlarr - Indexer centralized management

```
prowlarr:
  image: lscr.io/linuxserver/prowlarr:latest
  container_name: prowlarr
  environment:
    - PUID=${PUID}
    - PGID=${PGID}
    - TZ=${TZ}
  volumes:
    - ./prowlarr/config:/config
  ports:
    - 9696:9696
  restart: unless-stopped
  networks:
    - internal-proxy
```

Profilarr & Decluttarr (Config only, no media access needed usually)

```
profilarr:
  image: santiagosayshey/profilarr:latest
```

```
container_name: profilarr
environment:
  - TZ=${TZ}
volumes:
  - ./profilarr/config:/config
ports:
  - 6868:6868
restart: unless-stopped
networks:
  - internal-proxy
depends_on:
  - sonarr
  - radarr
```

FlareSolvrr - Bypasses Cloudflare protection for Indexers

```
flaresolvrr:
  image: ghcr.io/flaresolvrr/flaresolvrr:latest
  container_name: flaresolvrr
  environment:
    - LOG_LEVEL=${LOG_LEVEL:-info}
    - TZ=${TZ}
  ports:
    - 8191:8191
  restart: unless-stopped
  networks:
    - internal-proxy
```

□

Gluetun - VPN Client

```
gluetun:
  image: qmcgaw/glutun:latest
  container_name: glutun
  cap_add:
    - NET_ADMIN
  environment:
    - PUID=${PUID}
    - PGID=${PGID}
    - TZ=${TZ}
    - VPN_SERVICE_PROVIDER=${VPN_SERVICE_PROVIDER}
    - VPN_TYPE=${VPN_TYPE}
```

- WIREGUARD_PRIVATE_KEY=\${WIREGUARD_PRIVATE_KEY}
- WIREGUARD_ADDRESSES=\${WIREGUARD_ADDRESSES}
- VPN_PORT_FORWARDING=\${VPN_PORT_FORWARDING}

ports:

- 8888:8888 # Health check and control
- 6881:6881 # qBittorrent P2P port
- 6881:6881/udp

restart: unless-stopped

networks:

- internal-proxy

If using Gluetun, use qbittorrent from linuxserver io, NOT the hotio image below!

qbittorrent:

image: lscr.io/linuxserver/qbittorrent

network_mode: "service:gluetun" # The "Magic" line

depends_on:

gluetun:

condition: service_healthy # Wait for VPN to be up first

□

Lidarr - Music

lidarr:

image: lscr.io/linuxserver/lidarr:latest

container_name: lidarr

environment:

- PUID=\${PUID}
- PGID=\${PGID}
- TZ=\${TZ}

volumes:

- ./lidarr/config:/config
- \${ROOT_MEDIA_PATH}:/data

ports:

- 8686:8686

restart: unless-stopped

networks:

- internal-proxy

Bazarr - Subtitles

bazarr:

image: lscr.io/linuxserver/bazarr:latest

```
container_name: bazarr
environment:
  - PUID=${PUID}
  - PGID=${PGID}
  - TZ=${TZ}
volumes:
  - ./bazarr/config:/config
  - ${ROOT_MEDIA_PATH}:/data
ports:
  - 6767:6767
restart: unless-stopped
networks:
  - internal-proxy
```

whisper-ai - Local Subtitle Generation (Requires GPU)

whisper-ai:

```
image: ${WHISPER_IMAGE_URL} # Use a pre-built image like "jellyfin-whisper-
lab/whisper-container"
```

```
container_name: whisper-ai
```

```
environment:
```

```
  - PUID=${PUID}
  - PGID=${PGID}
  - TZ=${TZ}
```

```
volumes:
```

```
  - ./whisper/config:/config
  - ${ROOT_MEDIA_PATH}:/data
```

Enable GPU access if you have one, or comment out the 'deploy' section for CPU-only:

deploy:

resources:

reservations:

devices:

- driver: nvidia

count: all

capabilities: [gpu]

```
restart: unless-stopped
```

```
networks:
```

```
  - internal-proxy
```

Decluttarr - Queue Cleaner

```
decluttarr:
  image: hotio/decluttarr:latest
  container_name: decluttarr
  environment:
    - PUID=${PUID}
    - PGID=${PGID}
    - TZ=${TZ}
    - DECLUTTARR_CRON="*/15 * * * *"
    # API Keys/URLs for Sonarr, Radarr, etc. go here post-setup!
  volumes:
    - ./decluttarr/config:/config
  restart: unless-stopped
  networks:
    - internal-proxy
```

```
nzbget:
  image: lscr.io/linuxserver/nzbget:latest
  container_name: nzbget
  environment:
    - PUID=${PUID}
    - PGID=${PGID}
    - TZ=${TZ}
  volumes:
    - /etc/localtime:/etc/localtime:ro
    - ./nzbget:/config
    - /mnt/data:/data
  restart: unless-stopped
  networks:
    - internal-proxy
```

```
jellydash:
  image: ghcr.io/themartz90/jellydash:latest
  restart: unless-stopped
  ports:
    - "${APP_PORT:-8080}:80"
  environment:
    APP_ENV: "${APP_ENV:-production}"
    APP_DEBUG: "${APP_DEBUG:-false}"
    APP_TIMEZONE: "${APP_TIMEZONE:-UTC}"
```

```
# Bundled MariaDB below. Using your own database server instead? Point
# DB_HOST at it and drop the db service in docker-compose.override.yml.
DB_HOST: "${DB_HOST:-db}"
DB_PORT: "${DB_PORT:-3306}"
DB_DRIVER: mysql
DB_NAME: "${DB_NAME:-jellydash}"
DB_USER: "${DB_USER:-jellydash}"
DB_PASS: "${DB_PASS:-change-me}"
JELLYFIN_URL: "${JELLYFIN_URL}"
JELLYFIN_API_TOKEN: "${JELLYFIN_API_TOKEN}"
JELLYFIN_VERIFY_SSL: "${JELLYFIN_VERIFY_SSL:-true}"
LIBRARIES_CACHE_TTL: "${LIBRARIES_CACHE_TTL:-300}"
# Jellyseerr (optional; the page hides until these are set)
JELLYSEER_URL: "${JELLYSEER_URL:-}"
JELLYSEER_API_TOKEN: "${JELLYSEER_API_TOKEN:-}"
JELLYSEER_VERIFY_SSL: "${JELLYSEER_VERIFY_SSL:-true}"
SEERR_POLL_INTERVAL: "${SEERR_POLL_INTERVAL:-120}"
SEERR_NOTIFY_ENABLED: "${SEERR_NOTIFY_ENABLED:-true}"
# Background poller (history recording, cache warming, alerts)
POLLER_ENABLED: "${POLLER_ENABLED:-true}"
POLL_INTERVAL: "${POLL_INTERVAL:-30}"
# Web Push notifications (optional; generate keys once, see README)
PUSH_ENABLED: "${PUSH_ENABLED:-true}"
VAPID_PUBLIC_KEY: "${VAPID_PUBLIC_KEY:-}"
VAPID_PRIVATE_KEY: "${VAPID_PRIVATE_KEY:-}"
VAPID_SUBJECT: "${VAPID_SUBJECT:-mailto:admin@example.com}"
PUSH_IGNORE_USERS: "${PUSH_IGNORE_USERS:-}"
# Public base URL (optional): external alert links point back here.
APP_URL: "${APP_URL:-}"
# Extra notification channels; each is active once its config is set.
TELEGRAM_BOT_TOKEN: "${TELEGRAM_BOT_TOKEN:-}"
TELEGRAM_CHAT_ID: "${TELEGRAM_CHAT_ID:-}"
PUSHOVER_APP_TOKEN: "${PUSHOVER_APP_TOKEN:-}"
PUSHOVER_USER_KEY: "${PUSHOVER_USER_KEY:-}"
DISCORD_WEBHOOK_URL: "${DISCORD_WEBHOOK_URL:-}"
# Libraries hidden from Trending/Most Watched (also editable in Settings)
TRENDING_EXCLUDE_LIBRARIES: "${TRENDING_EXCLUDE_LIBRARIES:-}"
# Optional login gate (recommended when internet-facing)
AUTH_ENABLED: "${AUTH_ENABLED:-false}"
```

```

AUTH_ADMIN_USER: "${AUTH_ADMIN_USER:-}"
AUTH_ADMIN_PASSWORD: "${AUTH_ADMIN_PASSWORD:-}"
depends_on:
  db:
    condition: service_healthy
volumes:
  - ./jellydash_cache:/var/www/html/cache
  - ./jellydash_runtime_cache:/var/www/html/var/cache
  - ./jellydash_logs:/var/www/html/var/log
  - ./jellydash_uploads:/var/www/html/public/uploads
  # Drop-in feature modules (see docs/MODULES.md), e.g.:
  # - ./my-modules/downloads:/var/www/html/modules/downloads:ro

db:
  image: mariadb:11
  restart: unless-stopped
  environment:
    MARIADB_RANDOM_ROOT_PASSWORD: "yes"
    MARIADB_DATABASE: "${DB_NAME:-jellydash}"
    MARIADB_USER: "${DB_USER:-jellydash}"
    MARIADB_PASSWORD: "${DB_PASS:-change-me}"
  healthcheck:
    test: ["CMD", "healthcheck.sh", "--connect", "--innodb_initialized"]
    interval: 5s
    timeout: 5s
    retries: 12
  volumes:
    - ./jellydash_data:/var/lib/mysql

```

Gluetun

Source: [ThomasWildeTech Arr stack docker compose file](#)

Media Server Janitors

Revision #15

Created 2026-08-12 23:34:26 UTC by dpasn

Updated 2026-08-19 17:59:36 UTC by dpasn