GUIDE

Build a Robust Docker Media Server with Plex, the 'Arr' Stack, and Traefik

> **What you'll have running:** A fully automated, secure, and accessible media server stack capable of managing, downloading, and streaming movies, TV shows, and music. **Estimated time:** ~6 hours **Difficulty:** INTERMEDIATE **Power usage:** N/A (highly dependent on chosen hardware)
>
> **Hardware needed:**

- A server (e.g., mini PC, old desktop, or dedicated server) with 8GB+ RAM, 4+ CPU cores
- Ample storage (e.g., 4TB+ HDD for media, SSD for OS/apps recommended)

Introduction: Why Self-Host Your Media?

Tired of subscription hopping, content disappearing from streaming services, or not having your entire media library at your fingertips? Self-hosting your media server gives you complete control. Imagine a world where your entire collection of movies, TV shows, and music is beautifully organized, automatically downloaded, and streamable to any device, anywhere, securely. This isn't just about saving a few bucks; it's about digital ownership, privacy, and the sheer satisfaction of building something powerful yourself.

This guide will walk you through setting up a robust, automated media server using Docker, the 'Arr' stack (Radarr, Sonarr, Lidarr, Prowlarr), qBittorrent, Plex Media Server, and Traefik for secure, external access with automatic SSL. It's an intermediate journey, but every step is designed to empower you with a truly impressive homelab setup.

Prerequisites: Hardware, Operating System (Linux), and Initial Setup

Before we dive into the software, let's ensure your foundation is solid.

Hardware Considerations

Your server is the heart of your media empire. While specific recommendations depend on your budget and desired scale, here's what to aim for:

- **CPU:** 4+ cores are recommended. Modern Intel CPUs (e.g., i3/i5 8th gen or newer) or AMD Ryzen equivalents are excellent, especially if they support hardware transcoding (Quick Sync for Intel, AMF/VCN for AMD) which Plex can leverage to reduce CPU load during streaming.
- **RAM:** 8GB is a good starting point. If you plan to run many other services or transcode multiple 4K streams simultaneously, 16GB+ would be safer.
- **Storage:** This is crucial.
    - **OS/Applications:** A fast SSD (120GB+ NVMe or SATA) for your operating system and Docker application volumes will make everything snappy.
    - **Media Storage:** Ample HDD space (4TB+ recommended). Consider multiple drives in a RAID configuration (software RAID like ZFS or mdadm) for data redundancy, especially if your media collection is precious.
- **Network:** A Gigabit Ethernet port is essential for fast media transfers and streaming.

> 💡 Tip: Look for mini PCs like Intel NUCs, Dell Optiplex Micro, HP Elitedesk Mini, or Lenovo Tiny series on the used market. They offer excellent performance-per-watt and are compact.

Operating System (Linux)

We'll be using **Ubuntu Server 24.04 LTS** for its stability, widespread community support, and excellent Docker compatibility.

1. **Install Ubuntu Server 24.04 LTS:** Follow the official Ubuntu installation guide to get a minimal server installation. Ensure you enable SSH during installation for remote access.
2. **Initial Server Access:** Once installed, connect via SSH from your local machine. Replace `your_username` and `your_server_ip` with your actual details.

```
ssh your_username@your_server_ip
```

1. **Update and Upgrade:** Always start with a fresh system update.

```
sudo apt update && sudo apt upgrade -y
```

```
**Verification:**
The `sudo apt upgrade -y` command will output a summary of updated packages.
If everything is up to date, it will simply return to the prompt.
```

```
All packages are up to date.
```

1. **Install Essential Utilities:** `curl` and `git` are handy.

```
sudo apt install curl git -y
```

1. **Configure Firewall (UFW):** Enable the Uncomplicated Firewall (UFW) and allow SSH. We'll add more rules later.

```
sudo ufw allow ssh
sudo ufw enable
```

```
**Verification:**
Check UFW status.
```

```
sudo ufw status
```

```
**Expected Output:**
```

```
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
```

```
22/tcp (v6)                ALLOW       Anywhere (v6)
```

Network Requirements

For a robust and accessible media server, a good network setup is key:

- **Dedicated Static IP:** Assign a static IP address to your server on your local network. This prevents its IP from changing, which would break port forwarding and internal service communication. You typically configure this in your router's DHCP reservation settings, associating your server's MAC address with a fixed IP.
- **Router with Port Forwarding:** You'll need to forward specific ports from your router's public IP to your server's static local IP. This is usually found under "NAT," "Port Forwarding," or "Virtual Servers" in your router's web interface. We'll forward ports 80 and 443 for Traefik.
- **Domain Name:** Essential for Traefik to obtain Let's Encrypt SSL certificates. You can purchase one from a registrar like Namecheap or Cloudflare.
- **Dynamic DNS (Optional but Recommended):** If your home's public IP address changes (most residential ISPs do this), a dynamic DNS service (like Cloudflare DNS, DuckDNS, No-IP) will automatically update your domain's A record to point to your current public IP. This ensures your server is always reachable via your domain name. Configure this either on your router (if it supports it) or using a Docker container like `ddclient`.

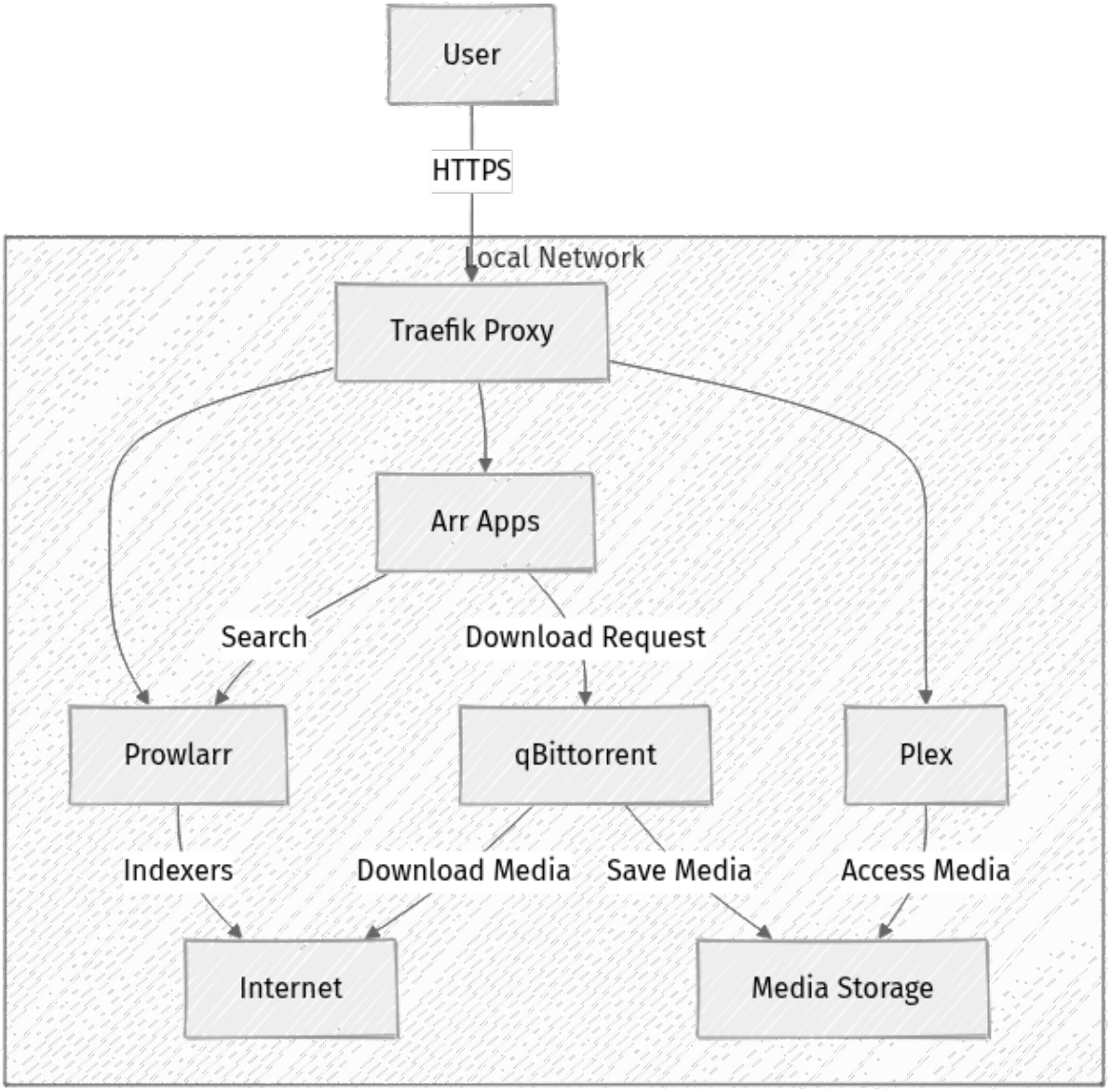
Core Components Overview: Plex, the 'Arr' Stack, Download Clients, and Traefik

Let's quickly demystify the components that will form your media server stack. Understanding their roles will help you grasp the overall architecture.

- **Plex Media Server:** The star of the show. Plex organizes your media, adds rich metadata (posters, synopses, cast info), and streams it to any Plex client (TVs, phones, tablets, web browsers). It also handles transcoding media on-the-fly to ensure smooth playback on various devices and network conditions.

- **The 'Arr' Stack:** A suite of applications designed for media automation.
    - **Radarr:** Automatically finds, downloads, and manages movies.
    - **Sonarr:** Automatically finds, downloads, and manages TV shows.
    - **Lidarr:** Automatically finds, downloads, and manages music.
    - **Prowlarr:** An indexer manager. It aggregates multiple torrent trackers and Usenet indexers, providing a single point of access for Radarr, Sonarr, and Lidarr to search for media.
- **Download Client (qBittorrent/Transmission):** These are the workhorses that actually download the media files found by the 'Arr' stack. We'll use qBittorrent in this guide due to its feature set and robust API.
- **Traefik Proxy:** Your secure gateway to the internet. Traefik is a modern reverse proxy and load balancer that integrates seamlessly with Docker. It automatically discovers your services, routes external traffic to them, and, crucially, handles automatic SSL certificate generation and renewal from Let's Encrypt. This means all your services will be accessible via `<https://your-service.your-domain.com>` without manual certificate management.

Here's a simplified view of how these components interact:



Installing Docker and Docker Compose

Docker is the foundation for running all our services in isolated containers. Docker Compose allows us to define and run multi-container Docker applications with a single command.

1. **Install Docker Engine:** First, ensure you remove any old versions of Docker.

```
    for pkg in docker.io docker-doc docker-compose docker-ce docker-ce-cli con
tainerd.io docker-buildx-plugin docker-compose-plugin; do sudo apt remove
$pkg; done
```

Then, install the necessary packages for Docker's repository.

```
    sudo apt update
    sudo apt install ca-certificates curl gnupg -y
    sudo install -m 0755 -d /etc/apt/keyrings
    curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /etc/apt/keyrings/docker.gpg
    sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

Add the Docker repository to Apt sources.

```
    echo \
      "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/
docker.gpg] https://download.docker.com/linux/ubuntu \
      "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
      sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
    sudo apt update
```

Finally, install Docker Engine, containerd, and Docker Compose (as a plugin).

```
    sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-
plugin docker-compose-plugin -y
```

Verification:
Check Docker's version and status.

```
    docker --version
    sudo systemctl status docker
```

Expected Output (versions may vary, but should be recent):

```
    Docker version 26.1.3, build xxx
```

```
    ● docker.service - Docker Application Container Engine
         Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor
preset: enabled)
```

```
         Active: active (running) since ...
```

1. **Add Your User to the Docker Group:** This allows you to run `docker` commands without `sudo`. You'll need to log out and back in for this to take effect.

```
    sudo usermod -aG docker $USER
```

```
**Verification:**
Log out of your SSH session (`exit`) and log back in. Then try running a
Docker command without `sudo`.
```

```
    docker run hello-world
```

```
**Expected Output:**
```

```
    Hello from Docker!
    This message shows that your installation appears to be working correctly.
    ...
```

Structuring Your Project and `docker-compose.yml`

A well-organized project structure is crucial for maintainability. We'll create a main directory for our media server, with subdirectories for configurations and media.

1. **Create Project Directories:**

```
    mkdir -p ~/docker/media-stack
    cd ~/docker/media-stack
    mkdir -p config/{traefik,plex,radarr,sonarr,lidarr,prowlarr,qbittorrent}
    mkdir -p media/{movies,tvshows,music,downloads}
```

```
This creates a `~/docker/media-stack` directory. Inside, `config` holds
application-specific configuration files, and `media` holds your actual media
```

```
content. The `downloads` folder within `media` will be used by qBittorrent for
temporary downloads.
```

1. **User ID and Group ID (PUID/PGID):** Docker containers often run as `root` by default, which can lead to permission issues when writing to host-mounted volumes. Many `linuxserver.io` containers (which we'll use) support `PUID` (Process User ID) and `PGID` (Process Group ID) environment variables. This allows the container's internal process to run as a specific user/group on the host, matching your media directory permissions.

   Find your user's PUID and PGID:

```
    id $USER
```

```
**Expected Output (example):**
```

```
    uid=1000(your_username) gid=1000(your_username)
groups=1000(your_username),4(adm),27(sudo),999(docker)
```

```
Note down your `uid` (PUID) and `gid` (PGID), typically `1000` for the first
user.
```

1. **Set Permissions for Media and Config Directories:** Ensure your user owns these directories so Docker containers running as your user can write to them.

```
    sudo chown -R $USER:$USER ~/docker/media-stack/config
    sudo chown -R $USER:$USER ~/docker/media-stack/media
    sudo chmod -R 775 ~/docker/media-stack/config
    sudo chmod -R 775 ~/docker/media-stack/media
```

```
> ⚠️ Warning: Incorrect permissions are the #1 cause of issues in Docker
homelab setups. Always double-check your `PUID`/`PGID` and directory
ownership.
```

Comprehensive `docker-compose.yml`

Now, let's create our main `docker-compose.yml` file. This defines all our services, their images, volumes, environment variables, and network configurations. Create this file in `~/docker/media-stack/docker-compose.yml`.

```
version: '3.8'

# Define global environment variables for PUID/PGID and timezone
# Replace 1000 with your actual PUID/PGID if different
# Replace Europe/London with your timezone
x-environment: &default-environment
  TZ: Europe/London
  PUID: 1000
  PGID: 1000

# Define common labels for Traefik, including network and restart policy
x-traefik-labels: &traefik-labels
  restart: unless-stopped
  networks:
    - proxy

services:
  # Traefik Reverse Proxy
  traefik:
    image: traefik:v2.11
    container_name: traefik
    command:
      - --api.dashboard=true
      - --api.insecure=false # Set to true for local testing, but false for
production
      - --providers.docker=true
      - --providers.docker.exposedbydefault=false
      - --providers.docker.network=proxy
      - --entrypoints.web.address=:80
      - --entrypoints.websecure.address=:443
      - --entrypoints.web.http.redirections.entrypoint.to=websecure
      - --entrypoints.web.http.redirections.entrypoint.scheme=https
      - --certificatesresolvers.letsencrypt.acme.email=your-email@example.com
# REPLACE WITH YOUR EMAIL
      - --certificatesresolvers.letsencrypt.acme.storage=/etc/traefik/
acme.json
      - --certificatesresolvers.letsencrypt.acme.tlschallenge=true # Use TLS
challenge for Let's Encrypt
      # - --certificatesresolvers.letsencrypt.acme.caserver=https://acme-
staging-v02.api.letsencrypt.org/directory # Uncomment for testing Let's
Encrypt
    ports:
      - "80:80"
      - "443:443"
      # - "8080:8080" # Uncomment to expose Traefik dashboard locally
(insecure)
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - ~/docker/media-stack/config/traefik/acme.json:/etc/traefik/acme.json
# Stores Let's Encrypt certs
    labels:
      <<: *traefik-labels
      # Traefik Dashboard (optional)
      - "traefik.enable=true"
      - "traefik.http.routers.traefik-dashboard.rule=Host(`traefik.your-
domain.com`)" # REPLACE WITH YOUR DOMAIN
      - "traefik.http.routers.traefik-dashboard.entrypoints=websecure"
      - "traefik.http.routers.traefik-dashboard.service=api@internal"
      - "traefik.http.routers.traefik-dashboard.middlewares=authelia@docker"
# Example for Authelia
```

```
      - "traefik.http.routers.traefik-dashboard.tls.certresolver=letsencrypt"
      - "traefik.http.services.traefik-
dashboard.loadbalancer.server.port=8080" # Traefik's internal API port
    networks:
      - proxy

  # Plex Media Server
  plex:
    image: lscr.io/linuxserver/plex:latest
    container_name: plex
    environment:
      <<: *default-environment
      VERSION: docker
    volumes:
      - ~/docker/media-stack/config/plex:/config
      - ~/docker/media-stack/media/movies:/movies
      - ~/docker/media-stack/media/tvshows:/tvshows
      - ~/docker/media-stack/media/music:/music
      - ~/docker/media-stack/media/downloads:/data # For temporary download
access by Plex (e.g., optimized versions)
    labels:
      <<: *traefik-labels
      - "traefik.enable=true"
      - "traefik.http.routers.plex.rule=Host(`plex.your-domain.com`)" #
REPLACE WITH YOUR DOMAIN
      - "traefik.http.routers.plex.entrypoints=websecure"
      - "traefik.http.routers.plex.tls.certresolver=letsencrypt"
      - "traefik.http.services.plex.loadbalancer.server.port=32400" # Plex's
internal HTTP port
    networks:
      - proxy
      # - host # Uncomment if you prefer host networking for Plex (simpler,
but less isolated)

  # Radarr (Movies)
  radarr:
    image: lscr.io/linuxserver/radarr:latest
    container_name: radarr
    environment:
      <<: *default-environment
    volumes:
      - ~/docker/media-stack/config/radarr:/config
      - ~/docker/media-stack/media/movies:/movies # Final destination for
movies
      - ~/docker/media-stack/media/downloads:/downloads
# qBittorrent download directory
    labels:
      <<: *traefik-labels
      - "traefik.enable=true"
      - "traefik.http.routers.radarr.rule=Host(`radarr.your-domain.com`)" #
REPLACE WITH YOUR DOMAIN
      - "traefik.http.routers.radarr.entrypoints=websecure"
      - "traefik.http.routers.radarr.tls.certresolver=letsencrypt"
      - "traefik.http.services.radarr.loadbalancer.server.port=7878" #
Radarr's internal HTTP port
    networks:
      - proxy

  # Sonarr (TV Shows)
  sonarr:
    image: lscr.io/linuxserver/sonarr:latest
    container_name: sonarr
```

```
    environment:
      <<: *default-environment
    volumes:
      - ~/docker/media-stack/config/sonarr:/config
      - ~/docker/media-stack/media/tvshows:/tvshows
# Final destination for TV shows
      - ~/docker/media-stack/media/downloads:/downloads
# qBittorrent download directory
    labels:
      <<: *traefik-labels
      - "traefik.enable=true"
      - "traefik.http.routers.sonarr.rule=Host(`sonarr.your-domain.com`)" #
REPLACE WITH YOUR DOMAIN
      - "traefik.http.routers.sonarr.entrypoints=websecure"
      - "traefik.http.routers.sonarr.tls.certresolver=letsencrypt"
      - "traefik.http.services.sonarr.loadbalancer.server.port=8989" #
Sonarr's internal HTTP port
    networks:
      - proxy

  # Lidarr (Music)
  lidarr:
    image: lscr.io/linuxserver/lidarr:latest
    container_name: lidarr
    environment:
      <<: *default-environment
    volumes:
      - ~/docker/media-stack/config/lidarr:/config
      - ~/docker/media-stack/media/music:/music # Final destination for music
      - ~/docker/media-stack/media/downloads:/downloads
# qBittorrent download directory
    labels:
      <<: *traefik-labels
      - "traefik.enable=true"
      - "traefik.http.routers.lidarr.rule=Host(`lidarr.your-domain.com`)" #
REPLACE WITH YOUR DOMAIN
      - "traefik.http.routers.lidarr.entrypoints=websecure"
      - "traefik.http.routers.lidarr.tls.certresolver=letsencrypt"
      - "traefik.http.services.lidarr.loadbalancer.server.port=8686" #
Lidarr's internal HTTP port
    networks:
      - proxy

  # Prowlarr (Indexer Manager)
  prowlarr:
    image: lscr.io/linuxserver/prowlarr:latest
    container_name: prowlarr
    environment:
      <<: *default-environment
    volumes:
      - ~/docker/media-stack/config/prowlarr:/config
    labels:
      <<: *traefik-labels
      - "traefik.enable=true"
      - "traefik.http.routers.prowlarr.rule=Host(`prowlarr.your-domain.com`)"
# REPLACE WITH YOUR DOMAIN
      - "traefik.http.routers.prowlarr.entrypoints=websecure"
      - "traefik.http.routers.prowlarr.tls.certresolver=letsencrypt"
      - "traefik.http.services.prowlarr.loadbalancer.server.port=9696" #
Prowlarr's internal HTTP port
    networks:
      - proxy
```

```
  # qBittorrent (Download Client)
  qbittorrent:
    image: lscr.io/linuxserver/qbittorrent:latest
    container_name: qbittorrent
    environment:
      <<: *default-environment
      WEBUI_PORT: 8080 # Internal web UI port for qBittorrent
    volumes:
      - ~/docker/media-stack/config/qbittorrent:/config
      - ~/docker/media-stack/media/downloads:/downloads # Main downloads
folder
    labels:
      <<: *traefik-labels
      - "traefik.enable=true"
      - "traefik.http.routers.qbittorrent.rule=Host(`qbittorrent.your-
domain.com`)" # REPLACE WITH YOUR DOMAIN
      - "traefik.http.routers.qbittorrent.entrypoints=websecure"
      - "traefik.http.routers.qbittorrent.tls.certresolver=letsencrypt"
      - "traefik.http.services.qbittorrent.loadbalancer.server.port=8080" #
qBittorrent's internal WEBUI_PORT
    networks:
      - proxy

  # Portainer CE (Optional - Docker Management GUI)
  # Uncomment if you want a web UI to manage your Docker containers
  # portainer:
  #   image: portainer/portainer-ce:latest
  #   container_name: portainer
  #   command: -H unix:///var/run/docker.sock
  #   volumes:
  #     - /var/run/docker.sock:/var/run/docker.sock
  #     - ~/docker/media-stack/config/portainer_data:/data
  #   labels:
  #     <<: *traefik-labels
  #     - "traefik.enable=true"
  #     - "traefik.http.routers.portainer.rule=Host(`portainer.your-
domain.com`)" # REPLACE WITH YOUR DOMAIN
  #     - "traefik.http.routers.portainer.entrypoints=websecure"
  #     - "traefik.http.routers.portainer.tls.certresolver=letsencrypt"
  #     - "traefik.http.services.portainer.loadbalancer.server.port=9000" #
Portainer's internal HTTP port
  #   networks:
  #     - proxy

networks:
  proxy:
    external: true # Traefik will manage this network
```

Before running `docker-compose up`:

- **REPLACE `your-email@example.com`** with your actual email in the `traefik` service.
- **REPLACE `your-domain.com`** with your actual domain name for each service's `Host()` rule. For example, if your domain is `mydomain.com`, `plex.your-domain.com` becomes `plex.mydomain.com`.

- Ensure the `PUID` and `PGID` in `x-environment` match your user's IDs.
- Adjust `TZ` to your correct timezone (e.g., `America/New_York`).
- If you plan to use Portainer, uncomment its section.

Example `docker-compose.override.yml` for Local Adjustments

Sometimes you need temporary changes or local-only configurations without altering your main `docker-compose.yml`. This is where `docker-compose.override.yml` comes in handy. Docker Compose automatically merges this file with your main `docker-compose.yml`.

Create `~/docker/media-stack/docker-compose.override.yml`:

```
version: '3.8'

services:
  traefik:
    ports:
      - "8080:8080" # Expose Traefik dashboard only for local debugging
    command:
      - --api.insecure=true # Enable insecure API for local dashboard access

  plex:
    environment:
      PLEX_CLAIM: "claim-xxxxxxxxxxxx" # Temporarily claim Plex server (find
this on plex.tv/claim)
```

> 💡 Tip: Use `docker-compose.override.yml` for things like exposing additional ports for debugging, enabling insecure APIs temporarily, or setting one-off environment variables. Remember to remove or comment out these overrides when you're done!

Configuring Traefik for Reverse Proxy, SSL (Let's Encrypt), and Middleware

Traefik is the unsung hero of this setup, providing secure, automated access to all your services.

1. **Create the Traefik Network:** Traefik needs a dedicated Docker network to communicate with your services.

```
docker network create proxy
```

```
**Verification:**
```

```
    docker network ls
```

```
**Expected Output (look for 'proxy'):**
```

```
    NETWORK ID     NAME      DRIVER    SCOPE
    ...
    xxxxxxxxxxxx   proxy     bridge    local
    ...
```

1. **DNS Configuration:** For Traefik to obtain SSL certificates, your domain's A records must point to your server's public IP address.
    - Log in to your domain registrar or DNS provider (e.g., Cloudflare, Namecheap).
    - Create `A` records for each service you want to expose:
        - `traefik.your-domain.com` -> `your_public_ip` (for Traefik dashboard, optional)
        - `plex.your-domain.com` -> `your_public_ip`
        - `radarr.your-domain.com` -> `your_public_ip`
        - `sonarr.your-domain.com` -> `your_public_ip`
        - `lidarr.your-domain.com` -> `your_public_ip`
        - `prowlarr.your-domain.com` -> `your_public_ip`
        - `qbittorrent.your-domain.com` -> `your_public_ip`
    - If using Cloudflare, ensure the proxy status (orange cloud) is **off** for these records initially, as Traefik needs direct access for the TLS challenge. You can enable it later if you understand the implications.

2. **Router Port Forwarding:** You need to forward incoming traffic on ports 80 and 443 from your public IP to your server's **static local IP address**.
    - **Where to find it:** Log in to your router's web interface (usually `192.168.1.1` or `192.168.0.1`). Look for sections like "NAT," "Port Forwarding," "Virtual Servers," or "Firewall."
    - **What to change:**
        - **External Port:** 80, **Internal Port:** 80, **Protocol:** TCP, **Internal IP:** `your_server_static_ip`
        - **External Port:** 443, **Internal Port:** 443, **Protocol:** TCP, **Internal IP:** `your_server_static_ip`
3. **Start Traefik and Services:** Navigate to your `~/docker/media-stack` directory and bring up all containers.

```
    cd ~/docker/media-stack
    docker compose up -d
```

```
The `-d` flag runs containers in detached mode (background).

**Verification:**
Check if Traefik is running and its logs.
```

```
    docker compose ps
    docker compose logs traefik
```

```
**Expected Output for `docker compose ps` (all services should be
'running'):**
```

```
    NAME              COMMAND                  SERVICE
STATUS              PORTS
    lidarr            "/init"                  lidarr
running
    plex              "/init"                  plex
running
    prowlarr          "/init"                  prowlarr
running
    qbittorrent       "/init"                  qbittorrent
running
    radarr            "/init"                  radarr
running
    sonarr            "/init"                  sonarr
running
    traefik           "/entrypoint.sh --api…"  traefik
```

```
running             0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp
```

```
**Expected Output for `docker compose logs traefik` (look for successful
certificate acquisition):**
```

```
    traefik    | time="2026-06-18T12:00:00Z" level=info msg="Certificates
obtained successfully" domain="plex.your-domain.com" providerName=letsencrypt
ACME CA="https://acme-v02.api.letsencrypt.org/directory"
```

```
It might take a minute or two for Traefik to acquire the certificates. If you
see errors, double-check your DNS A records and port forwarding.

Now, you should be able to access your Traefik dashboard (if enabled) at
`<https://traefik.your-domain.com`> and other services via their respective
subdomains.
```

Setting Up Plex Media Server

Plex is typically the first service you'll configure.

1. **Access Plex Web UI:** Open your web browser and navigate to `<https://plex.your-domain.com>`.
2. **Claim Your Server:**
    - You'll be prompted to sign in or create a Plex account. Do so.
    - Plex should automatically detect your new server. Give it a friendly name (e.g., "Homelab Plex").
    - Ensure "Allow me to access my media outside my home" is checked if you want remote access.
    - Click "Next."

3. **Add Libraries:**
    - You'll be asked to add your media libraries. Click "Add Library."
    - **Type:** Select "Movies."
    - **Name:** "Movies."
    - **Folders:** Click "Browse for media folder" and select `/movies` (this maps to `~/docker/media-stack/media/movies` on your host).
    - Repeat for "TV Shows" (pointing to `/tvshows`) and "Music" (pointing to `/music`).

Verification: After adding libraries, Plex will start scanning your folders. If you have existing media, it should begin populating. If not, you'll see empty libraries. The key is that the libraries are created and pointing to the correct paths.

> 💡 Tip: If Plex doesn't detect your server, try accessing it directly via its local IP and port (e.g., `<http://your_server_static_ip:32400/web>`) before it's claimed, then sign in. Once claimed, access through Traefik should work.

Integrating the 'Arr' Stack (Radarr, Sonarr, Lidarr, Prowlarr)

The 'Arr' stack works together to automate your media acquisition. We'll set up Prowlarr first, then connect the others.

Prowlarr Setup

1. **Access Prowlarr Web UI:** Go to `<https://prowlarr.your-domain.com>`.

2. **Initial Setup:**
    - Set an admin username and password.
    - Go to **Settings > Indexers**.
    - Click the `+` button to add new indexers. Search for your preferred public or private torrent trackers (e.g., "Nyaa" for anime, "RARBG" for general content) or Usenet providers. Follow the specific instructions for each indexer (API key, URL, etc.).
    - Go to **Settings > Apps**.
    - Click the `+` button and add `Radarr`, `Sonarr`, and `Lidarr`. For each, select the appropriate application, provide the API key (you'll find this in each 'Arr' app's settings once they're configured), and test. Prowlarr will then push the configured indexers to these apps.

Radarr Setup

1. **Access Radarr Web UI:** Go to `<https://radarr.your-domain.com>`.

2. **Initial Setup:**
    - **Settings > General:** Set a username and password for the web UI.
    - **Settings > Media Management:**
        - **Root Folders:** Add `/movies` as a root folder.
        - **Advanced Settings:** Enable "Advanced Settings" at the top.
        - **File Management:** Enable "Rename Movies" and configure a naming scheme (e.g., `{Movie Title} ({Release Year}) {Quality Full}`).
    - **Settings > Download Clients:**
        - Click `+`, select "qBittorrent."
        - **Host:** `qbittorrent` (this is the Docker service name, allowing internal Docker network communication).
        - **Port:** `8080`.
        - **Username/Password:** Set these in qBittorrent later.
        - **Category:** `radarr` (important for keeping downloads organized).
        - **Save & Test.**
    - **Settings > Indexers:** Prowlarr should have pushed your indexers here. If not, go back to Prowlarr and ensure Radarr is configured as an app.
    - **Settings > Profiles:** Configure your preferred quality profiles (e.g., 1080p, 4K).
3. **Add a Movie:** Go to "Movies," click "Add New," search for a movie, select a quality profile, and add it. Radarr will then send a request to Prowlarr to search and then to qBittorrent to download.

Sonarr Setup (Similar to Radarr)

1. **Access Sonarr Web UI:** Go to `<https://sonarr.your-domain.com>`.

2. **Initial Setup:**
    - **Settings > General:** Set a username and password.
    - **Settings > Media Management:**
        - **Root Folders:** Add `/tvshows` as a root folder.
        - **Advanced Settings:** Enable "Advanced Settings."
        - **File Management:** Enable "Rename Episodes" and configure a naming scheme (e.g., `{Series Title} - S{season:00}E{episode:00} - {Episode Title}`).
    - **Settings > Download Clients:**
        - Click `+`, select "qBittorrent."
        - **Host:** `qbittorrent`, **Port:** `8080`.
        - **Category:** `sonarr`.
        - **Save & Test.**
    - **Settings > Indexers:** Verify Prowlarr pushed indexers.
    - **Settings > Profiles:** Configure quality profiles.
3. **Add a TV Show:** Go to "Series," click "Add New," search for a show, select a root folder and quality profile.

Lidarr Setup (Similar to Radarr/Sonarr)

1. **Access Lidarr Web UI:** Go to `<https://lidarr.your-domain.com>`.

2. **Initial Setup:**
    - **Settings > General:** Set a username and password.
    - **Settings > Media Management:**
        - **Root Folders:** Add `/music` as a root folder.
        - **Advanced Settings:** Enable "Advanced Settings."
        - **File Management:** Enable "Rename Tracks" and configure a naming scheme.
    - **Settings > Download Clients:**
        - Click `+`, select "qBittorrent."
        - **Host:** `qbittorrent`, **Port:** `8080`.
        - **Category:** `lidarr`.
        - **Save & Test.**
    - **Settings > Indexers:** Verify Prowlarr pushed indexers.
    - **Settings > Profiles:** Configure quality profiles.
3. **Add an Artist:** Go to "Artists," click "Add New," search for an artist, select a root folder and quality profile.

Configuring Your Download Client (qBittorrent/ Transmission)

We're using qBittorrent for its robust features and API.

1. **Access qBittorrent Web UI:** Go to `<https://qbittorrent.your-domain.com>`.
2. **Initial Login:** The default username is `admin` and the default password is `adminadmin`. **Change this immediately!**
    - Go to **Tools > Options > Web UI**.
    - Change the **Authentication** username and password.
    - Click "Save."

3. **Configure Paths:**
    - Go to **Tools > Options > Downloads**.
    - **Save files to location:** `/downloads` (this is the host path `~/docker/media-stack/media/downloads`).
    - **Keep incomplete files in:** `/downloads/incomplete`.
    - **Automatically add torrents from:** Check this and point it to a folder like `/downloads/watch`. The 'Arr' apps will place `.torrent` files here.
    - Ensure "Append .!qB extension to incomplete files" is checked.
4. **Set Categories:**
    - In the "Downloads" section, scroll down to "Torrent Management."
    - Enable "Enable Categories."
    - The categories `radarr`, `sonarr`, `lidarr` that you set in the 'Arr' apps will automatically be created here when the first download is initiated.
    - For each category, set the "Save path" to `/downloads`. The 'Arr' apps will then handle the final move and renaming from this `/downloads` directory to `/movies`, `/tvshows`, or `/music`.

    Verification: After saving, attempt to add a test torrent manually or through one of the 'Arr' apps. Observe that it starts downloading and is placed in `/downloads/incomplete` with the `.!qB` extension, then moves to `/downloads` upon completion. The 'Arr' app should then detect, move, and rename it.

Storage Management and Best Practices (Mounts, Permissions, Data Integrity)

Proper storage management is fundamental for a reliable media server.

Docker Volume Mounts

In our `docker-compose.yml`, we're using **bind mounts**. This means a directory on your host machine (e.g., `~/docker/media-stack/media/movies`) is directly mounted into the container (e.g., `/movies`).

- **Pros of Bind Mounts:** Simple, direct access to host filesystem, easy to manage permissions from the host.
- **Cons of Bind Mounts:** Less portable than Docker volumes, host path must exist.

> 💡 Tip: For configuration data (`/config` in our setup), bind mounts are generally preferred for easy access and backup. For media, bind mounts are also ideal as you want to manage the large media files directly on your host filesystem.

Permissions (PUID/PGID Revisited)

The `PUID` and `PGID` environment variables are critical. If your Docker containers try to write to a host-mounted directory with different user IDs than the directory's owner, you'll encounter "Permission denied" errors.

- Ensure the `PUID` and `PGID` in your `docker-compose.yml` match the user ID (`uid`) and group ID (`gid`) of the user who owns your `~/docker/media-stack/media` and `~/docker/media-stack/config` directories.
- The `chmod -R 775` command ensures that the owner and group have read/write/execute permissions, and others have read/execute. This is generally safe for media directories in a homelab.

Data Integrity

Your media collection is valuable. Consider these options for protecting it:

- **RAID (Redundant Array of Independent Disks):**
  - **Software RAID (mdadm):** A common choice on Linux. You can create RAID1 (mirroring) or RAID5/6 (striping with parity) arrays from multiple drives.
  - **ZFS:** A powerful filesystem that integrates volume management, snapshots, and data integrity checks. It's more complex but highly recommended for critical data. Requires more RAM (8GB+ dedicated to ZFS for larger pools).
  - **Btrfs:** Another modern Linux filesystem with features like snapshots, checksums, and built-in RAID capabilities.

> ⚠️ Warning: RAID is not a backup! It protects against drive failure, but not against accidental deletion, ransomware, or catastrophic server failure. Always implement a separate backup strategy.

Security Hardening: Firewall, VPN, and Access Control

Exposing services to the internet requires careful security measures.

Firewall (UFW)

We already enabled UFW. Let's refine the rules.

1. **Allow Traefik Ports:** Ensure ports 80 and 443 are open for Traefik.

```
    sudo ufw allow 80/tcp comment 'Allow HTTP for Traefik'
    sudo ufw allow 443/tcp comment 'Allow HTTPS for Traefik'
```

1. **Restrict SSH Access (Optional but Recommended):** If you have a static public IP for your administrative workstation, you can restrict SSH access to only that IP.

```
    sudo ufw allow from your_admin_workstation_ip to any port 22 comment 'Allo
w SSH from admin workstation'
    sudo ufw delete allow ssh # Removes the "Anywhere" rule
```

```
**Verification:**
```

```
    sudo ufw status verbose
```

```
**Expected Output (example):**
```

```
    Status: active
    Logging: on (low)
    Default: deny (incoming), allow (outgoing), deny (routed)
    New profiles: skip

    To                         Action      From
    --                         ------      ----
    80/tcp                     ALLOW IN    Anywhere
    443/tcp                    ALLOW IN    Anywhere
    22/tcp                     ALLOW IN    your_admin_workstation_ip # Or
Anywhere if not restricted
```

VPN for Download Client

To protect your privacy when downloading torrents, route your qBittorrent traffic through a VPN. This is typically done by running the qBittorrent container within a VPN container, or using a specialized image. The `linuxserver/qbittorrent` image supports VPN integration directly via environment variables.

1. **VPN Provider:** Choose a reputable VPN provider that supports WireGuard or OpenVPN (e.g., ProtonVPN, Mullvad, NordVPN, Private Internet Access).
2. **Configure qBittorrent VPN:** You'll need to modify your `docker-compose.yml` for qBittorrent. This is an example for WireGuard; consult your VPN provider's documentation for specific details.

```
    # ... inside qbittorrent service definition
    environment:
      <<: *default-environment
      WEBUI_PORT: 8080
      VPN_ENABLED: true
      VPN_CLIENT: wireguard # or openvpn
      VPN_ENDPOINT: "wg.your-vpn-provider.com:51820" # Replace with your VPN
endpoint
      VPN_PUBKEY: "YOUR_VPN_PUBLIC_KEY" # Replace with your VPN public key
      VPN_ADDRESS: "10.10.10.2/32" # Replace with your VPN client IP address
      VPN_PRESHARED_KEY: "YOUR_VPN_PRESHARED_KEY"
# Optional, if your VPN uses it
      VPN_DNS: "1.1.1.1,1.0.0.1" # Recommended: Cloudflare DNS
      LAN_NETWORK: "192.168.1.0/24" # Your local network subnet, e.g.,
192.168.1.0/24
    cap_add:
      - NET_ADMIN # Required for VPN to function
    sysctls:
      - net.ipv4.conf.all.src_valid_lables=1
    # ...
```

```
**After updating `docker-compose.yml`, restart the qBittorrent container:**
```

```
    docker compose restart qbittorrent
```

```
**Verification:**
Access qBittorrent UI, go to **Tools > Options > Connection**. Under "Network
Interface," you should see an interface like `wg0` or `tun0`. You can also try
a torrent client IP leak test to confirm traffic is routed through the VPN.
```

Access Control (Traefik Basic Auth Middleware)

To add a layer of security to your services (especially the 'Arr' apps and Traefik dashboard), you can use Traefik's Basic Auth middleware.

1. **Generate Hashed Password:** You need an htpasswd compatible hash of your username:password.

```
    echo $(htpasswd -nb your_username your_password)
```

```
Copy the output (e.g., `your_username:$apr1$xxxxxxx`).
```

1. **Create Traefik Middleware File:** Create a file `~/docker/media-stack/config/traefik/dynamic.yml`:

```
    http:
      middlewares:
        secured-auth:
          basicAuth:
            users:
              - "your_username:$apr1$xxxxxxx" # REPLACE WITH YOUR GENERATED HASH
              # - "another_user:$apr1$yyyyyyy" # Add more users if needed
```

1. **Update Traefik Configuration:** Modify the `traefik` service in your `docker-compose.yml` to load this dynamic configuration and apply the middleware to desired services.

```
    # ... inside traefik service
    command:
      # ... existing commands ...
      - --providers.file.directory=/etc/traefik/dynamic # Point to dynamic config directory
      - --providers.file.watch=true # Watch for changes
    volumes:
      # ... existing volumes ...
      - ~/docker/media-stack/config/traefik/dynamic.yml:/etc/traefik/dynamic/dynamic.yml:ro # Mount dynamic config
    labels:
      # ... existing labels ...
      # Example: Apply middleware to Radarr
      - "traefik.http.routers.radarr.middlewares=secured-auth@file" # Apply the middleware
    # ...
```

```
Apply the `secured-auth@file` middleware label to any service you want to
protect.

**Restart Traefik:**
```

```
    docker compose restart traefik
```

```
**Verification:**
Access `<https://radarr.your-domain.com`>. You should be prompted for a
username and password before reaching the Radarr UI.
```

Maintenance, Updates, and Backup Strategies

A self-hosted server requires ongoing care.

System Updates

Regularly update your host operating system.

```
sudo apt update && sudo apt upgrade -y
sudo apt autoremove -y
sudo reboot # If kernel or critical packages were updated
```

Docker Container Updates

You have two main approaches:

1. **Manual Updates:** Navigate to your `~/docker/media-stack` directory.

```
    cd ~/docker/media-stack
    docker compose pull # Downloads latest images
    docker compose up -d # Recreates containers with new images
```

```
This is generally safe but requires you to be proactive.
```

1. **Automated Updates with Watchtower (Use with Caution):**
   Watchtower can automatically check for new Docker images and update your running containers. While convenient, it can sometimes introduce breaking changes without your intervention.

   Add Watchtower to your `docker-compose.yml` (e.g., at the bottom):

```
    # ...
    watchtower:
      image: containrrr/watchtower
      container_name: watchtower
      restart: unless-stopped
      volumes:
        - /var/run/docker.sock:/var/run/docker.sock
      environment:
        # Optional: update every 24 hours (86400 seconds)
        - WATCHTOWER_INTERVAL=86400
        # Optional: only update specific containers
        # - WATCHTOWER_LABEL_ENABLE=true
      # labels:
      #   - "com.centurylinklabs.watchtower.enable=true" # Uncomment and add
to other services to enable label filtering
```

```
If you use `WATCHTOWER_LABEL_ENABLE=true`, you'll need to add
`com.centurylinklabs.watchtower.enable=true` to the `labels` section of each
service you want Watchtower to manage.

**Start Watchtower:**
```

```
    docker compose up -d watchtower
```

```
> ⚠️ Warning: Automating updates can lead to unexpected issues if a new
version introduces breaking changes. Always review release notes for critical
applications before auto-updating.
```

Backup Strategies

Backups are non-negotiable for any homelab.

1. **Configuration Backups:** The `~/docker/media-stack/config` directory contains all your application settings, databases, and Traefik's SSL certificates (`acme.json`). This is crucial to back up.

```
    # Example: Create a compressed archive of your config
    tar -czvf ~/media-stack-config-backup-$(date +%Y%m%d).tar.gz ~/docker/
media-stack/config
```

```
Regularly copy this archive to an off-site location (cloud storage, another
machine).
```

1. **Media Backups:** Backing up large media libraries can be challenging.
    - **External HDD:** Periodically sync your media to an external drive.
    - **Cloud Storage:** Services like Backblaze B2, Google Drive, or S3 can be used, but costs scale with data volume.
    - **Another Server:** Sync to a NAS or another homelab server.

> 💡 Tip: Tools like `rsync` are excellent for incremental backups of large media libraries. For example: `rsync -av --delete /path/to/media/ /path/to/backup/destination/`

Troubleshooting Common Issues and Advanced Tips

Even the most robust setups encounter hiccups. Here's how to tackle common problems.

Troubleshooting Common Issues

1. **"Permission Denied" Errors:**
    - **Cause:** Docker containers running as a user that doesn't have write access to host-mounted volumes.
    - **Fix:**
        - Verify your `PUID` and `PGID` in `docker-compose.yml` match your host user's `uid` and `gid`.
        - Ensure the host directories have correct ownership and permissions:

```
            sudo chown -R $USER:$USER ~/docker/media-stack/media
            sudo chown -R $USER:$USER ~/docker/media-stack/config
            sudo chmod -R 775 ~/docker/media-stack/media
            sudo chmod -R 775 ~/docker/media-stack/config
```

```
    - Restart the affected container: `docker compose restart <service_name>`
- **Error Example:**
```

```
       plex    | [error] Failed to write to /config/Library/Application
Support/Plex Media Server/Logs/Plex Media Server.log: Permission denied
```

1. **Service Not Accessible via Domain / SSL Errors:**
    - **Cause:** Incorrect DNS, port forwarding, or Traefik configuration.
    - **Fix:**
        - **DNS:** Double-check your A records point to your server's public IP. Use `dig your-service.your-domain.com` to verify.
        - **Port Forwarding:** Confirm ports 80 and 443 are forwarded to your server's static local IP. Use an online port checker to see if they're open.
        - **Traefik Logs:** Check Traefik logs for certificate acquisition errors: `docker compose logs traefik`. Look for ACME (Let's Encrypt) errors.
        - **Traefik Dashboard:** If `traefik.your-domain.com` is working, check the dashboard for router/service status.
        - **Firewall:** Ensure UFW isn't blocking ports 80/443.
2. **'Arr' App Can't Connect to Download Client (qBittorrent):**
    - **Cause:** Incorrect host/port, wrong credentials, or network issues between containers.
    - **Fix:**
        - **Host:** Ensure the 'Arr' apps are configured to connect to `qbittorrent` (the Docker service name), not `localhost` or your server's IP.
        - **Port:** Confirm the port is `8080` (the internal `WEBUI_PORT` for qBittorrent).
        - **Credentials:** Double-check the username and password in both the 'Arr' app and qBittorrent settings.
        - **Network:** All services should be on the `proxy` network for them to communicate by service name.

3. **Media Not Moving/Renaming After Download:**
    - **Cause:** Permissions on media folders, incorrect path mappings in 'Arr' apps, or 'Arr' app not detecting completed downloads.
    - **Fix:**
        - **Permissions:** Re-verify `PUID`/`PGID` and host folder permissions for `~/docker/media-stack/media`.
        - **Paths:** In Radarr/Sonarr/Lidarr, ensure:
            - "Root Folders" point to `/movies`, `/tvshows`, or `/music`.
            - "Download Client" settings for qBittorrent have "Remote Path Mappings" configured correctly if your download client is on a different host (not needed if all in one `docker-compose.yml`).
            - The "Category" in the 'Arr' app matches the category in qBittorrent.
        - **Logs:** Check the logs of the specific 'Arr' app for clues: `docker compose logs radarr`.

Advanced Tips

- **Resource Limits:** For stability on lower-powered hardware, consider adding resource limits to your `docker-compose.yml` services, especially for Plex if transcoding is heavy.

```
    # ... inside a service definition (e.g., plex)
    deploy:
      resources:
        limits:
          cpus: '2.0' # Limit to 2 CPU cores
          memory: 4G # Limit to 4GB RAM
```

- **Healthchecks:** Add healthchecks to your `docker-compose.yml` to ensure Docker knows if your containers are truly healthy, not just running.

```
    # ... inside a service definition
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:32400/web || exit 1"] #
Example for Plex
      interval: 30s
      timeout: 10s
      retries: 3
```

```
      start_period: 30s
```

- **External Configuration for Traefik:** While we used a simple `dynamic.yml` for basic auth, Traefik can load complex configurations from external files, which is useful for more advanced routing, middlewares, or security policies.
- **Monitoring:** Consider adding monitoring tools like `Prometheus` and `Grafana` to track your server's resource usage, Docker container status, and application metrics.

This guide provides a solid foundation for your self-hosted media server. The journey of self-hosting is an ongoing learning experience. Embrace the challenges, leverage the vast community resources, and enjoy the freedom and control of your own digital domain!