GUIDE

Build Your Essential Home Media Server (Plex/Jellyfin)

What you'll have running: A fully functional home media server accessible from your devices, capable of streaming your personal media collection securely. **Estimated time:** ~4 hours **Difficulty:** BEGINNER **Power usage:** ~15-30W idle (~$15-30/year for a mini PC)

Hardware needed:

- Mini PC (e.g., Intel NUC, Beelink, ZimaBoard) or an old desktop computer with an Intel CPU (8th gen or newer for hardware transcoding support) and at least 8GB RAM
- Large capacity Hard Disk Drive (HDD) for media storage (e.g., 4TB+)
- USB drive (8GB+) for OS installation
- Ethernet cable
- Monitor, keyboard, mouse (for initial setup)

Introduction: Why a Home Media Server?

Ever get tired of juggling streaming subscriptions, or finding that your favorite movie isn't available anywhere? Building your own home media server puts you back in control of your entertainment. Imagine having your entire collection of movies, TV shows, home videos, and music, all organized beautifully and ready to stream to any device in your house – or even securely when you're away from home.

This guide will walk you through setting up a powerful, yet energy-efficient, media server using Jellyfin (a fantastic open-source alternative to Plex) on Ubuntu Linux. We'll leverage Docker for easy installation and management, and Tailscale for secure remote access without fumbling with complex router settings. It's a rewarding project that gives you ownership over your digital media and a great entry point into the world of homelabbing.

Hardware Requirements & Recommendations

Choosing the right hardware is crucial for a smooth media server experience. While you can technically use anything, a good foundation prevents headaches down the line, especially with video transcoding.

The Brain: Your Server PC

For a beginner-friendly media server, I highly recommend a mini PC or a repurposed old desktop.

- **Mini PC (Recommended):** Devices like an Intel NUC, Beelink, or ZimaBoard are perfect. They are small, quiet, and most importantly, energy-efficient. Look for one with an **Intel CPU (8th generation or newer)**. This is key for **hardware transcoding** with Intel Quick Sync Video, which dramatically reduces CPU load when streaming to devices that can't play the original file directly (e.g., converting a 4K file to 1080p for a phone).
    - **RAM:** At least **8GB RAM** is a good starting point. 16GB offers more headroom if you plan to run other services later.
- **Old Desktop Computer:** If you have an old desktop lying around, it can work too. Again, prioritize an Intel CPU (8th gen or newer) if possible. Desktops consume more power and are often louder, but they offer more expansion slots for hard drives.

> 💡 **Tip:** If your CPU is older or not Intel, software transcoding will still work, but it will be much more CPU-intensive and might struggle with multiple simultaneous streams or high-bitrate content.

The Storage: Your Media Vault

This is where your media lives. You'll need ample space.

- **Large Capacity Hard Disk Drive (HDD):** Start with at least a **4TB HDD**. Media files are big, and your collection will grow. HDDs offer the best cost-per-gigabyte.
    - **Internal vs. External:** An internal 3.5-inch SATA HDD is generally preferred for reliability and speed, but a good quality external USB 3.0 HDD can also work.

- **USB Drive (8GB+):** This is for installing Ubuntu. Any standard USB stick will do.

Connectivity & Peripherals

- **Ethernet Cable:** Always connect your server via Ethernet for stable and fast network performance. Wi-Fi can be unreliable for streaming high-bitrate video.
- **Monitor, Keyboard, Mouse:** You'll need these for the initial setup of Ubuntu. Once configured, you'll rarely need them again, as you'll manage the server remotely via SSH.

Power Cost Estimation

A mini PC typically idles around 15-30W. Running 24/7, this translates to:

- 15W * 24 hours * 365 days = 131.4 kWh/year
- 30W * 24 hours * 365 days = 262.8 kWh/year At an average electricity cost of $0.10/kWh, this is roughly **$13 - $26 per year**. This is incredibly efficient for a dedicated server!

Operating System Installation (Ubuntu Desktop LTS)

We're going with Ubuntu Desktop LTS (Long Term Support). It's stable, widely supported, and has a user-friendly interface for initial setup, while still being a powerful Linux distribution under the hood.

1. Download Ubuntu Desktop LTS

Go to the official Ubuntu website and download the latest LTS version of Ubuntu Desktop. As of today, that would be a recent release like 22.04 LTS or 24.04 LTS.

> 💡 **Tip:** Always choose the "LTS" (Long Term Support) version. These releases receive updates and support for several years, meaning less frequent reinstallation or major upgrades.

2. Create a Bootable USB Drive

You'll need a tool to "burn" the Ubuntu ISO image onto your USB drive.

- **Windows:** Use Rufus or Balena Etcher.
- **macOS:** Use Balena Etcher.

- **Linux:** Use `dd` command or Balena Etcher.

Using Balena Etcher (Cross-Platform):

1. Download and install Balena Etcher from their official website.
2. Insert your 8GB+ USB drive.
3. Open Etcher, click "Flash from file" and select the Ubuntu ISO you downloaded.
4. Click "Select target" and choose your USB drive (double-check you're selecting the correct drive to avoid data loss!).
5. Click "Flash!" and wait for the process to complete.

> ⚠️ **Warning:** Ensure you select the correct USB drive in Etcher. Flashing the wrong drive will erase all data on it!

3. Install Ubuntu Desktop LTS on Your Server

1. Plug the bootable USB drive, monitor, keyboard, and mouse into your server PC.
2. Power on the PC and repeatedly press the key to enter the boot menu (often `F2`, `F10`, `F12`, or `Del` – check your PC's manual or on-screen prompts).
3. Select your USB drive as the boot device.
4. When Ubuntu loads, choose **"Try or Install Ubuntu"**.
5. On the desktop, double-click **"Install Ubuntu [version]"**.

6. Follow the on-screen prompts:
    - **Keyboard Layout:** Choose your preferred layout.
    - **Updates and Other Software:** Select "Normal installation" and check "Download updates while installing Ubuntu" and "Install third-party software for graphics and Wi-Fi hardware" (if applicable).
    - **Installation Type:** Select **"Erase disk and install Ubuntu"**. This will wipe the entire drive and install Ubuntu.

        > ⚠️ **Warning:** This option will delete all data on the selected drive. Make sure you're installing to the correct drive (the SSD/HDD you want to be your server's boot drive, not your large media HDD if it's already connected).

    - **Where are you?** Select your time zone.
    - **Who are you?** Enter your name, server's name (e.g., `mediaserver`), username (e.g., `homelabuser`), and a strong password. **Make a note of this username and password!** Choose "Require my password to log in".
7. The installation will proceed. This can take some time.
8. Once complete, you'll be prompted to restart. Remove the USB drive when instructed.

Initial Server Configuration & Updates

After Ubuntu reboots, log in with the user you created. Now we'll do some essential post-installation tasks.

1. Update Your System

It's crucial to ensure your system is up-to-date with the latest security patches and software.

```
sudo apt update
sudo apt upgrade -y
sudo apt autoremove -y
```

Verification: After running `sudo apt upgrade -y`, you should see output indicating packages were upgraded or that the system is already up to date.

```
# Expected output might look like:
# Reading package lists... Done
# Building dependency tree... Done
# Reading state information... Done
# Calculating upgrade... Done
# 0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

2. Set a Static IP Address

A static IP address ensures your server always has the same network address, making it easy to find and connect to. We'll configure this on your router.

1. **Find your server's current IP address:** Open a terminal on your server (Ctrl+Alt+T) and run:

```
ip a
```

```
Look for the `inet` address under your active network interface (usually
`enpXsX` or `eth0`). It will look something like `192.168.1.100/24`. Note this
down. Also note your router's IP (often the `gateway` listed in `ip r` or
`192.168.1.1` or `192.168.0.1`).
```

1. **Access your router's administration page:** Open a web browser on any computer connected to your home network and navigate to your router's IP address (e.g., `<http://192.168.1.1>`). Log in with your router's admin credentials (often found on a sticker on the router itself, or in its manual).
2. **Configure a Static IP (DHCP Reservation):**
    - Look for a section like "DHCP Reservation," "Static Lease," or "Address Reservation." This is usually under "LAN Settings," "Network Settings," or "Advanced."
    - You'll need your server's **MAC address**. You can find this by running `ip a` again and looking for the `link/ether` address next to your `inet` address (e.g., `aa:bb:cc:dd:ee:ff`).
    - Add a new reservation, associating your server's MAC address with a specific IP address outside your router's normal DHCP range (e.g., if DHCP gives out `192.168.1.100-192.168.1.200`, pick `192.168.1.50`).
    - Save the settings and restart your router if prompted.
    - Restart your Ubuntu server or run `sudo systemctl restart NetworkManager` to pick up the new IP.

Verification: After rebooting the server, run `ip a` again. The IP address for your server should now be the static one you assigned in your router.

```
ip a
```

```
# Expected output (example):
# 2: enp0s31f6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
state UP group default qlen 1000
#     link/ether aa:bb:cc:dd:ee:ff brd ff:ff:ff:ff:ff:ff
#     inet 192.168.1.50/24 brd 192.168.1.255 scope global dynamic
noprefixroute enp0s31f6
```

Notice `192.168.1.50` (or whatever you set) is now listed.

3. Enable SSH for Remote Access

SSH (Secure Shell) allows you to control your server from another computer without a monitor, keyboard, or mouse.

```
sudo apt install openssh-server -y
sudo systemctl enable ssh
sudo systemctl start ssh
```

Verification: From another computer on your network, open a terminal (or PuTTY on Windows) and try to connect:

```
ssh your_username@192.168.1.50 # Replace with your username and server IP
```

You'll be prompted to accept the host key and then enter your password. If successful, you'll see a command prompt for your server.

> 💡 **Tip:** For enhanced security, consider disabling password authentication for SSH and using SSH keys instead. This is a more advanced topic but highly recommended for any server exposed to the network.

4. Basic Firewall Setup (UFW)

Ubuntu comes with UFW (Uncomplicated Firewall). Let's enable it and allow essential services.

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

```
sudo ufw allow https
sudo ufw status verbose
```

Explanation:

- `sudo ufw enable`: Turns on the firewall. This will block all incoming connections by default, except those you explicitly allow.
- `sudo ufw allow ssh`: Allows incoming connections on port 22 (SSH).
- `sudo ufw allow http`: Allows incoming connections on port 80 (standard web traffic, though Jellyfin uses a different port).
- `sudo ufw allow https`: Allows incoming connections on port 443 (secure web traffic).
- `sudo ufw status verbose`: Shows the current firewall rules.

Verification: The `sudo ufw status verbose` command should show:

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

# To                         Action      From
# --                         ------      ----
# 22/tcp                     ALLOW IN    Anywhere
# 80/tcp                     ALLOW IN    Anywhere
# 443/tcp                    ALLOW IN    Anywhere
# 22/tcp (v6)                ALLOW IN    Anywhere (v6)
# 80/tcp (v6)                ALLOW IN    Anywhere (v6)
# 443/tcp (v6)               ALLOW IN    Anywhere (v6)
```

Installing Docker & Docker Compose

Docker simplifies running applications like Jellyfin by packaging them into isolated containers. Docker Compose allows us to define and run multi-container Docker applications with a single command.

1. Install Docker Engine

First, let's remove any old Docker installations and then install the latest version.

```
# Uninstall old versions (if any)
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-
docker containerd runc; do sudo apt remove $pkg; done

# Add Docker's official GPG key
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

# Install Docker Engine, containerd, and Docker Compose (CLI)
sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin do
cker-compose-plugin -y
```

Verification: Check the Docker version and run a test container.

```
docker --version
```

```
# Expected output (version may vary):
# Docker version 26.1.3, build 885c327
```

```
sudo docker run hello-world
```

```
# Expected output:
# Unable to find image 'hello-world:latest' locally
# latest: Pulling from library/hello-world
# ... (download progress) ...
# Hello from Docker!
# This message shows that your installation appears to be working correctly.
# ...
```

2. Add Your User to the Docker Group

To run Docker commands without `sudo`, add your user to the `docker` group.

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

You'll need to **log out and log back in** (or simply reboot the server) for this change to take effect.

Verification (after relogging/rebooting): Try running `docker run hello-world` again without `sudo`.

```
docker run hello-world
```

If it runs successfully, your user is now in the Docker group.

Setting Up Jellyfin (or Plex) with Docker Compose

We'll use Jellyfin for this guide, as it's open-source and free, offering a fantastic media server experience. The principles apply similarly if you choose Plex.

1. Prepare Directory Structure

It's good practice to keep your Docker configurations and media separate.

```
# Create a directory for your Docker Compose projects
mkdir -p ~/docker/jellyfin
cd ~/docker/jellyfin

# Create directories for Jellyfin configuration and cache
mkdir -p ./config
mkdir -p ./cache

# Create a directory for your media (if not already mounted/created)
# This assumes your large HDD is mounted at /mnt/media or similar
sudo mkdir -p /mnt/media/movies
sudo mkdir -p /mnt/media/tvshows
sudo mkdir -p /mnt/media/music
sudo mkdir -p /mnt/media/homevideos
```

> 💡 **Tip:** If your large HDD isn't automatically mounted or formatted, you'll need to do that first. Use `lsblk` to identify the drive, then `sudo fdisk /dev/sdX` (replace sdX) to partition, and `sudo mkfs.ext4 /dev/sdX1` to format. Finally, add an entry to `/etc/fstab` to auto-mount it on boot. For a beginner, it might be easier to connect and format the drive before installing Ubuntu, and let Ubuntu handle the mounting during installation if possible, or use the Disks utility in Ubuntu Desktop.

2. Set Permissions for Media Directories

Docker containers run as a specific user ID (UID) and group ID (GID). For Jellyfin to access your media, its user inside the container needs permissions on your media folders. The `linuxserver/jellyfin` image often runs as `PUID=1000` and `PGID=1000` by default, which usually matches your first Ubuntu user. However, it's safer to explicitly set permissions or ensure the user running Jellyfin has access.

```
# Change ownership of media directories to your user (PUID/PGID=1000)
# Replace 'your_username' with the user you created during Ubuntu installation
sudo chown -R your_username:your_username /mnt/media

# Grant read/write permissions to the owner, and read-only to others on media
sudo chmod -R 755 /mnt/media
```

> ⚠️ **Warning:** Be careful with `chmod` and `chown`. Incorrect usage can lock you out of files or expose them. For media, `755` for directories and `644` for files is generally safe.

3. Create the `docker-compose.yml` for Jellyfin

Now, let's create the `docker-compose.yml` file that defines our Jellyfin service.

```
nano ~/docker/jellyfin/docker-compose.yml
```

Paste the following content, making sure to replace `your_username` with your actual Ubuntu username if you didn't use `homelabuser` and adjust `/mnt/media` if your media path is different.

```
version: "2.1"
services:
  jellyfin:
    image: lscr.io/linuxserver/jellyfin:latest
    container_name: jellyfin
    environment:
      - PUID=1000 # Your user ID (run 'id -u' on your server to find it)
      - PGID=1000 # Your group ID (run 'id -g' on your server to find it)
      - TZ=Europe/London # Replace with your timezone (e.g., America/New_York)
      - JELLYFIN_PublishedServerUrl=http://192.168.1.50:8096 # Optional: Set
if you want to advertise a specific URL
    volumes:
      - ./config:/config
      - ./cache:/cache
      - /mnt/media/movies:/data/movies:ro # Read-only access for media
      - /mnt/media/tvshows:/data/tvshows:ro
      - /mnt/media/music:/data/music:ro
      - /mnt/media/homevideos:/data/homevideos:ro
    ports:
      - 8096:8096 # Jellyfin web interface
      - 8920:8920 # Jellyfin HTTPS (if enabled)
      - 7359:7359/udp # Auto-discovery
      - 1900:1900/udp # DLNA discovery
    devices:
      # Uncomment the following lines for Intel Quick Sync Video hardware
transcoding
      # Make sure your user is in the 'render' group: sudo usermod -aG render
$USER
      # Then reboot your server.
```

```
      # - /dev/dri:/dev/dri
    restart: unless-stopped
```

Explanation of the `docker-compose.yml`:

- `version: "2.1"`: Specifies the Docker Compose file format version.
- `services:`: Defines the different applications in your stack.
- `jellyfin:`: Our single service, named `jellyfin`.
- `image: lscr.io/linuxserver/jellyfin:latest`: Tells Docker to pull the latest Jellyfin image from `linuxserver.io`, which provides excellent, well-maintained Docker images.
- `container_name: jellyfin`: Gives the container a friendly name.
- `environment:`:
  - `PUID=1000`, `PGID=1000`: These are crucial. They tell the container to run as the user with UID 1000 and GID 1000 on your host system. This usually matches your first user created in Ubuntu. You can verify your user's PUID/PGID by running `id -u` and `id -g` respectively.
  - `TZ=Europe/London`: Set your correct timezone. This affects logging and scheduled tasks.
  - `JELLYFIN_PublishedServerUrl`: Optional, but can help clients find your server.
- `volumes:`: Maps directories from your host system into the container.
  - `./config:/config`: The `config` directory inside your `jellyfin` project folder on the host is mapped to `/config` inside the container. This stores all Jellyfin's settings, database, and metadata. **This is vital for persistence!**
  - `./cache:/cache`: For temporary cache files.
  - `/mnt/media/movies:/data/movies:ro`: Your actual media folders are mapped into the container. `:ro` means "read-only," which is a good security practice for media folders, preventing the server from accidentally modifying your original files.

- `ports:`: Maps ports from the container to your host system.
  - `8096:8096`: Jellyfin's main web interface. The first `8096` is the host port, the second is the container port.
  - `8920:8920`: For HTTPS if you configure it later.
  - `7359:7359/udp`, `1900:1900/udp`: For network discovery protocols like DLNA.
- `devices:`: This section is critical for hardware transcoding.
  - `- /dev/dri:/dev/dri`: This maps your Intel integrated GPU's device files into the container, allowing Jellyfin to use Quick Sync Video. **Remember to uncomment this line and add your user to the `render` group (`sudo usermod -aG render $USER` followed by a reboot) if you have an Intel CPU with Quick Sync.**
- `restart: unless-stopped`: Ensures Jellyfin automatically restarts if the server reboots or the container crashes, unless you manually stop it.

Save the file and exit nano (Ctrl+X, then Y, then Enter).

4. Start Your Jellyfin Server

Navigate to your `~/docker/jellyfin` directory and start the Docker Compose stack.

```
cd ~/docker/jellyfin
docker compose up -d
```

- `docker compose up`: Starts the services defined in `docker-compose.yml`.
- `-d`: Runs the containers in "detached" mode, meaning they run in the background, and you get your terminal prompt back.

Verification: Check if the Jellyfin container is running.

```
docker ps
```

```
# Expected output:
# CONTAINER ID   IMAGE                                COMMAND
CREATED        STATUS
PORTS

                      NAMES
# <some_id>      lscr.io/linuxserver/jellyfin:latest "/init"      X seconds
```

```
ago  Up X seconds  0.0.0.0:1900->1900/udp, 0.0.0.0:7359->7359/udp,
0.0.0.0:8096->8096/tcp, 0.0.0.0:8920->8920/tcp   jellyfin
```

If you see `jellyfin` listed with `Status: Up`, it's running!

Organizing Your Media Library

A well-organized media library is key to a pleasant experience with Jellyfin. Consistent naming and folder structures help Jellyfin (and Plex) correctly identify your media and download metadata.

Recommended Structure

Stick to a simple, logical hierarchy.

```
/mnt/media/
├── movies/
│   ├── Movie Title (Year)/
│   │   └── Movie Title (Year).ext
│   ├── Another Movie (2023)/
│   │   └── Another Movie (2023).mkv
├── tvshows/
│   ├── TV Show Name/
│   │   ├── Season 01/
│   │   │   └── TV Show Name - S01E01 - Episode Title.ext
│   │   ├── Season 02/
│   │   │   └── TV Show Name - S02E01 - Episode Title.ext
│   ├── Another Show/
│   │   └── Season 01/
│   │       └── Another Show - S01E01 - Pilot.mp4
├── music/
│   ├── Artist Name/
│   │   ├── Album Title (Year)/
│   │   │   └── 01 - Song Title.mp3
├── homevideos/
│   ├── Family Vacation 2024/
│   │   └── Video 1.mp4
```

Naming Conventions

- **Movies:** `Movie Title (Year).ext` (e.g., `The Matrix (1999).mkv`)
- **TV Shows:** `Show Name - SXXEXX - Episode Title.ext` (e.g., `The Office (US) - S01E01 - Pilot.mp4`)
- **Music:** `Artist Name/Album Title (Year)/01 - Song Title.mp3`

> 💡 **Tip:** Use tools like FileBot or Tiny Media Manager on your desktop to help automate renaming and organizing your media before moving it to the server.

Accessing Your Media Locally

Now that Jellyfin is running, let's access its web interface and configure your media libraries.

1. Access the Jellyfin Web Interface

Open a web browser on any computer or device on your home network and navigate to:

```
http://192.168.1.50:8096
```

(Replace `192.168.1.50` with your server's static IP address).

2. Initial Jellyfin Setup

1. **Welcome Screen:** Choose your preferred language.
2. **Create Your User:** Create an admin username and a strong password. This is for logging into Jellyfin, separate from your Ubuntu user.
3. **Add Media Library:**
    - Click the `+` icon to add a new library.
    - **Content Type:** Select "Movies," "TV Shows," "Music," etc.
    - **Display Name:** Give your library a friendly name (e.g., "My Movies").
    - **Folders:** Click the `+` icon and navigate to the corresponding folder inside the container. Remember the `volumes` mapping from `docker-compose.yml`?
        - For Movies, add `/data/movies`
        - For TV Shows, add `/data/tvshows`
        - For Music, add `/data/music`
        - For Home Videos, add `/data/homevideos`
    - **Preferred Metadata Language/Country:** Set these as desired.
    - Click "OK."
4. **Finish:** Complete the wizard. Jellyfin will now start scanning your libraries. This might take some time depending on the size of your collection.

3. Install Jellyfin Clients

Jellyfin has client apps for almost every platform:

- Web browser (already using this)
- Android (phone, tablet, Android TV)
- iOS (iPhone, iPad, Apple TV)
- Roku
- Fire TV
- And more!

Download the appropriate app for your device, enter your server's local IP address (`<http://192.168.1.50:8096>`), and log in with your Jellyfin credentials.

Secure Remote Access with Tailscale

Accessing your media server from outside your home network usually involves complex and insecure port forwarding. Tailscale solves this by creating a secure, private network (a "mesh VPN") between your devices, no matter where they are.

1. What is Tailscale?

Tailscale is a Zero Trust VPN that uses the WireGuard protocol. It allows your devices to securely communicate with each other directly, as if they were on the same local network, even if they're thousands of miles apart. This means you don't need to open any ports on your router, dramatically improving security.

2. Install Tailscale on Your Server

Follow the official Tailscale installation guide for Ubuntu.

```
curl -fsSL https://tailscale.com/install.sh | sh
```

Verification: Check the Tailscale version.

```
tailscale version
```

```
# Expected output (version may vary):
# 1.66.4
#   go version: go1.22.3
#   ...
```

3. Connect Your Server to Your Tailscale Network

```
sudo tailscale up
```

This command will output a URL. Copy this URL and paste it into a web browser on your desktop. You'll be prompted to log in with a Google, Microsoft, GitHub, or other identity provider account. This links your server to your personal Tailscale network.

Once authorized, your server will join your Tailscale network and be assigned a unique Tailscale IP address (e.g., `100.x.y.z`).

Verification: Check your server's Tailscale status.

```
tailscale status
```

```
# Expected output:
# 100.x.y.z   mediaserver   your_username@example.com   linux   active; relay
"syd"
```

This shows your server is connected and its Tailscale IP.

4. Access Jellyfin Remotely

1. **Install Tailscale on Your Client Device:** Download and install the Tailscale app on your phone, laptop, or other device you want to use for remote access.
2. **Log in to Tailscale:** Log into the Tailscale app on your client device using the same identity provider account you used for your server.
3. **Access Jellyfin:** Once your client device is connected to Tailscale, it can now reach your server's Tailscale IP address as if it were local. Open your Jellyfin client app or web browser and use your server's **Tailscale IP address** and Jellyfin port:

```
    http://100.x.y.z:8096
```

```
(Replace `100.x.y.z` with your server's Tailscale IP from `tailscale status`).
```

You now have secure, remote access to your media server without any port forwarding!

Basic Maintenance & Updates

Keeping your server and services updated is crucial for security, performance, and new features.

1. Update Ubuntu

Regularly update your operating system. I usually do this once a month or whenever I hear about a major security patch.

```
sudo apt update
sudo apt upgrade -y
sudo apt autoremove -y
sudo reboot # If a kernel update or major system component was updated
```

2. Update Docker Images

Update your Docker containers to get the latest versions of Jellyfin and other services.

```
cd ~/docker/jellyfin # Navigate to your docker-compose.yml directory
docker compose pull # Downloads the latest images
docker compose up -d # Recreates containers with the new images, keeping data
docker image prune -f # Removes old, unused Docker images to save space
```

3. Backup Your Configuration

Your `docker-compose.yml` and the `./config` directory for Jellyfin are the most important things to back up.

```
# Create a backup archive of your Jellyfin configuration
cd ~/docker/jellyfin
tar -czvf jellyfin_config_backup_$(date +%Y%m%d).tar.gz ./config ./docker-
compose.yml

# Move the backup to a safe location (e.g., another drive, cloud storage)
# Example: cp jellyfin_config_backup_$(date +%Y%m%d).tar.gz /mnt/backup_drive/
```

> 💡 **Tip:** Automate backups with a cron job! You could schedule the `tar` command to run weekly and copy the archive to a network share or cloud storage.

Troubleshooting Common Issues

Even experienced homelabbers run into issues. Here are some common problems and their solutions.

1. Docker Container Not Starting / Exiting

Error Message:

```
Error response from daemon: driver failed programming external connectivity on
endpoint jellyfin (...): Error starting userland proxy: listen tcp
0.0.0.0:8096: bind: address already in use
```

Cause: Another service on your server is already using port `8096`. This often happens if you tried to install Jellyfin directly on the host before using Docker, or another application uses that port. **Fix:**

1. Identify the process using the port:

```
sudo lsof -i :8096
```

1. Stop the conflicting service or change the host port in your `docker-compose.yml` (e.g., `8097:8096`). If you change the host port, remember to access Jellyfin at `<http://192.168.1.50:8097>`.

Error Message:

```
ERROR: for jellyfin  Cannot start service jellyfin: OCI runtime create failed:
container_linux.go:380: starting container process caused:
process_linux.go:545: container init caused: process_linux.go:528: setting
cgroup config for procHooks process caused: failed to write "pids.max":
unknown.
```

Cause: This can sometimes happen on older kernel versions or specific Docker installations. **Fix:** Try updating your Docker daemon and kernel. If it persists, it might be a cgroup v1 vs v2 issue; ensure your system is using cgroup v1 or Docker is configured for v2. For most modern Ubuntu LTS installations, this shouldn't be an issue.

2. Jellyfin Not Seeing Media Files

Cause: Incorrect permissions or incorrect volume mapping in `docker-compose.yml`. **Fix:**

1. **Check `docker-compose.yml`:** Ensure your `volumes` section correctly points to the host paths (e.g., `/mnt/media/movies`) and the container paths (e.g., `/data/movies`).
2. **Check Host Permissions:** Log into your server via SSH and verify that the user ID (`PUID`) and group ID (`PGID`) used in your `docker-compose.yml` (e.g., `1000`) have read access to your media folders.

```
    ls -la /mnt/media/movies
```

```
If permissions are wrong, re-run `sudo chown -R your_username:your_username /
mnt/media` and `sudo chmod -R 755 /mnt/media`.
```

1. **Rescan Library:** In the Jellyfin web UI, go to **Admin Dashboard > Libraries**, select your library, and click "Scan Library."

3. Transcoding Issues (Stuttering, High CPU)

Cause: Software transcoding is being used, or hardware transcoding isn't configured correctly. **Fix:**

1. **Verify Hardware Transcoding:**
    - Ensure your CPU supports Intel Quick Sync Video (8th gen or newer Intel CPU).
    - Check that `/dev/dri` is mapped in your `docker-compose.yml` (uncomment the `devices` section).
    - Verify your user is in the `render` group: `groups $USER`. If not, `sudo usermod -aG render $USER` and reboot.
    - In Jellyfin web UI, go to **Admin Dashboard > Playback > Transcoding**. Ensure "Enable hardware acceleration" is checked and "Intel Quick Sync (QSV)" is selected for "Hardware acceleration API."
2. **Check Server Resources:** Use `htop` (install with `sudo apt install htop`) to monitor CPU usage during playback. If CPU is consistently at 100% during transcoding, your hardware might be underpowered for the content.

4. Tailscale Connectivity Issues

Cause: Tailscale service not running, client not logged in, or firewall blocking access. **Fix:**

1. **Server Status:** On your server, run `sudo tailscale status`. Ensure it shows `active`. If not, try `sudo tailscale up`.
2. **Client Status:** On your remote device, ensure the Tailscale app is running and you are logged in.
3. **Firewall:** On your server, ensure UFW is not blocking Tailscale. Tailscale usually punches its own holes, but if you have custom rules, ensure they aren't interfering. You can temporarily disable UFW (`sudo ufw disable`) for testing (then re-enable: `sudo ufw enable`).

Next Steps & Expanding Your Server

Congratulations! You've built a fully functional home media server. But the homelab journey rarely ends here. Here are some ideas for expanding your setup:

- *The Arr Suite (Sonarr, Radarr, Lidarr): These applications automate finding, downloading, and organizing TV shows, movies, and music. They integrate seamlessly with Jellyfin. You can run them in separate Docker containers alongside Jellyfin using the same `docker-compose.yml` or a new one.
- **More Storage:** As your media library grows, you'll inevitably need more disk space. Consider adding another large HDD to your server.
- **VPN for Downloads:** If you plan to automate media acquisition, integrating a VPN client (e.g., OpenVPN, WireGuard) into a separate Docker container for your download client (like qBittorrent) is a common and secure setup.
- **Monitoring:** Tools like Portainer (for Docker management) or Grafana/Prometheus (for system metrics) can give you deeper insights into your server's health and performance.
- **Reverse Proxy with SSL:** For more advanced remote access (e.g., using a custom domain name and HTTPS without Tailscale), you could set up a reverse proxy like Nginx Proxy Manager or Caddy. This is more complex and involves opening ports on your router, but provides a more "public" facing service.
- **Unattended Upgrades:** For Ubuntu, you can enable unattended upgrades to automatically install security updates, reducing manual intervention.

```
    sudo apt install unattended-upgrades -y
    sudo dpkg-reconfigure --priority=low unattended-upgrades
```

Enjoy your new media kingdom! This server is a versatile foundation for many other self-hosted services. The world of homelabbing is vast, and you've just taken a significant first step.