Music Streaming

- Navidrome
  - Install Navidrome and Stream Your Own Library using the Airsonic API
- MStream Music Streamer
  - MStream - Open Source Music Streaming at your Fingertips
- Ampache
  - Ampache Music Streaming Server
- Identifying Music
  - Install MusicBrainz in Docker
- Music Assistant
  - Install Music Assistant

Navidrome

Navidrome

Install Navidrome and Stream Your Own Library using the Airsonic API

https://www.youtube.com/embed/K3DtR3g2S8Y

Subsonic / Airsonic music streaming servers are an incredible way to self host your own Google Music, iTunes, Apple Music, Spotify, Pandora, and any other streaming service out there, by using your own music, and keeping it all local and under your control. The best thing is these are fairly straightforward to setup, and there are some really great mobile apps out there for free to use to stream once your server is setup.

What you'll need

- A machine you want to use as a music server
- Docker, Docker-Compose installed on that machine
- About 20 minutes of time.

Optional needs

If you want to access your music streaming server from outside your local network (over the internet) you'll need the following, otherwise these are optional:

- a domain name / subdomain with an A record pointing to your public IP address.
- NGinX Proxy Manager installed via Docker-compose (we'll install this with Docker and Docker-compose int eh next step.)
- The ability to setup port forwards on your network router for ports 80 and 443.

Docker, Docker-Compose, NGinX Proxy Manager Setup

I've got a script that will help you install all of these amazing bits of software at once if you have CentOS, Debian, or Ubuntu 18.04 or 20.04.

You just need to go to my github page and grab the right script for your OS.

https://github.com/bmcgonag/docker_installs

Find your OS and version, click the link to view the code, then copy the script code.

Create a new file called "docker-install.sh":

```
nano docker-install.sh
```

Now paste the copied code into the file you just created with CTRL + Shift + V.

Save the file wtih CTRL + O, then press Enter to confirm. Now use CTRL + X to exit the nano editor.

Next, make the file executable with:

```
chmod +x docker-install.sh
```

Finally, run the script with:

```
./docker-install.sh
```

Allow the script to run, and enter your sudo password if / when prompted.

> Reverse Proxy Setup further down - for outside access back to your music server.

Navidrome Setup:

1. create a new folder on your server called "navidrome".

```
mkdir navidrome
```

1. change into that new directory:

```
cd navidrome
```

1. create a new file called "docker-compose.yml"

```
nano docker-compose.yml
```

1. Copy the following into that file: You can use CTRl + Shift + V to paste into the nano editor.

```
version: "3.3"
services:
  navidrome:
    image: deluan/navidrome:latest
    ports:
      - "4533:4533"
    environment:
      ND_SCANINTERVAL: 30m
      ND_LOGLEVEL: info
      ND_BASEURL: ""
    volumes:
      - "./data:/data"
      - "/path/to/your/music/here:/music:ro"
```

NOTES:

You can change the left side of the port mapping above, as this is the host port you'll access the Navidrome web server on. If port 4533 is already in use for another application you'll need to change it. **DO NOT**, however, change the right side, as this is the port on the container, and this is where the application is expected to run.

You also need to change the `path/to/your/music/here` portion to be the actual file / folder path to your music collection.

For instance, my music collection path is:

`/mnt/video/arr/data/media/music`

where /music is the directory in which all of my music resides.

Start the container with the command

`docker-compose up -d`

Allow the image to download, and the container to spin up, and you can go to your site at your server's IP address and port 4533. For instance, I went to `http://192.168.7.125:4533`

Once there, you'll see the option to create an administrator account.

If you want to have access to your collection over the internet, then we have a bit more to do.

Setup Internet Access with NGinX Proxy Manager

Login to NGinx Proxy Manager at the IP of your server with port 81.

For example, I go to `http://192.168.7.125:81`

NOTE: NGinX Proxy Manager must be allowed to use the host ports 80, 443, and 81.

NOTE 2: You should forward the ports 80, and 443 only through your router configuration to the local (LAN) IP address of your server running NGinX Proxy Manager. This makes all traffic on 80 and 443 go to NGinX Proxy Manager, then we let NGinX Proxy Manager handle directing traffic around your network to the applications you specify, vs. opening a bunch of ports to the internet which is arguably less secure.

Now, go to the IP address and port 81 of your NGinX Proxy Manager server, and use the login credentials:

username: admin@example.com
password: changeme

Upon logging in the first time, you'll be prompted to change the Admin account details, as well as the admin password. You should absolutely make these different from the defaults.

Now, log out, and back in with your new credentials to make sure it's all working.

Finally, let's create a proxy host to our Navidrome instance. This allows us to give our Navidrome install a proper sub-domain / domain name that we can access from outside of our network.

Inside NGinX Proxy Manager click the Proxy tab. Click to add a new proxy, and in the pop-up window enter the domain / sub-domain you setup with the A-record. Press tab, or Enter to accept the name and make it a "chip".

Now go to the IP address field. Here we need the IP of our Navidrome gateway in docker. (Alternatively, if you setup a docker network, and assign both NGinX Proxy Manager and Navidrome to the same network, you can simply use the Navidrome container name in this field.)

NOTE: If you are running Navidrome on a different physical server than your NGinX Proxy Manager, you'll simply use the LAN IP of the server host you are running Navidrome on.

To find the gateway IP in Docker, enter the following in the terminal:

```
docker network list
```

This will list all of the docker networks. You are likely just using the default network, but make sure.

If you don't see a network specifically created for Navidrome (either by you, or by docker), then simply do the following for the default network:

```
docker network inspect <network name>
```

Where you replace < network name > above with the actual name of your default docker network or the network created for navidrome in Docker.

You should see a bunch of JSON output in the terminal. Just scroll up, searching for "gateway". Get that IP (Usually 172.17.0.1) and put it in the field in the proxy host window for IP address.

Now, move to the port field and put in 4533, unless you changed this in the docker-compose file. If you did, then please put that port number here.

Enable the "Block Common Exploits", and "Webrtc" options, then save.

You should see your new proxy entry listed in a row. Click the url in the row to open it in a new tab, and make sure you are taken to your navidrome install.

Once you see the Navidrome login, close the browser tab, and go back to NGinX Proxy Manager.

Next, we want SSL.

Click on the 3 dot symbol at the right end of the row for your navidrome proxy entry, and select 'Edit' from the pop-up menu.

In the proxy entry form, select the SSL tab, and from the first drop down list, select "Request a New SSL Certificate". Next, enable the "Force SSL" option, and then enter your email address if it's not already filled in. Finally, enable the "Accept Terms of Service" option.

Click 'Save'. This will prompt NGinX Proxy Manager to ask letsEncrypt for a new SSL certificate. LetsEncrypt will in turn try to reach your site on Port 80 (that's why we wanted to make sure it opened and got to the Navidrome site using our domain / sub-domain). After a few seconds (maybe a minute), the pop-up window should close with no errors.

Now click on the name in the row you see, and it should open your site in a new tab, with SSL enabled. Now create your admin user (if you haven't already), and make sure your music is showing up.

You're done! Congratulations! You can now install a subsonic capable music player on your smart device, enter your SSL encrypted domain, your user credentials, and start streaming your own music to your mobile device with no third party involvement!

Support my Channel and Content

Support my Channel and ongoing efforts through Patreon:

https://www.patreon.com/bePatron?u=234177

MStream Music Streamer

MStream - Open Source Music Streaming at your Fingertips

https://www.youtube.com/embed/1CrPCxThlRY

MStream is an open source, self hosted music streaming server with an Android companion app, and an iOS app on the way. You can stream through your web browser, as well as use the streaming service in Jukebox mode. An alternative to the Subsonic / Airsonic API based servers, it's got a clean, intuitive interface, and is very straight-forward to setup and use.

What You'll Need

- Docker-CE
- Docker-Compose
- About 10 minutes of time
- The Path to your music library
- (Optional) NGinX Proxy Manager
- (Optional) a domain / Sub-domain to access your music from outside your home lab, or via a URL instead of an IP address.

Installation

We are going to install the MStream server using a modified version of the MStream team's docker-compose file.

For the best organization, we want to keep our Docker and Docker Compose files in a "root" level folder called "docker".

If you haven't setup a "docker" folder, why not start here.

```
mkdir docker
```

Now move into that docker folder:

```
cd docker
```

Next, let's make a folder called MStream.

`mkdir mstream`

and move into that folder:

`cd mstream`

In here, we'll make a new file called "docker-compose.yml"

`nano docker-compose.yml`

In this file, we need to paste the following block of yaml code, and modify it to suit your needs and setup. We'll go through the parts you may need to change after the code block:

```
version: "2.1"
services:
  mstream:
    image: lscr.io/linuxserver/mstream
    container_name: mstream
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=America/Chicago
    volumes:
      - /home/brian/docker/mstream/config:/config
      - /home/brian/Music:/music
    ports:
      - 3000:3000
    restart: unless-stopped
```

As always, the above is a yaml file, and yaml is very (and I mean very!) specific about spaces, so be cautious as you make changes.

First, you'll want to check your User ID and Group ID. so, save the file with CTRL + O, and press Enter to confirm, the exit the nano editor with CTRL + X. Next, in the terminal type

`id`

You'll see a list of ID numbers for your user, group, and a few others. Make note of your user and group ids, and we'll use those back in the docker-compose.yml file. Open the file again:

`nano docker-compose.yml`

and replace the PUID and PGID with your user and group id's respectively (if they are different from what is there already).

Next, set your timezone. In my case it's "America/Chicago", but make sure to se the correct timezone for where you are.

Under the "volumes" section, you'll want to set the path to where your mstream folder is. **Remember to only change teh left side of the colon ":"**. The right side is always preset by the container, and should not be changed. Use the full path, as this just generally works regardless of the docker and docker-compose version.

For the music location, make sure, to again, put the full path your where your music is stored. Again, changing only the left side of the colon.

Finally, in the "ports" section, feel free to leave it as 3000 on the left (this is the host machine port that you'll use to access the MStream server). If, however, port 3000 is already in use on this host, then feel free to change the left side of the port mapping to any port that is open and free. In my case, I ended up setting it to 8215, so the port mapping for me looks like:

```
ports:
  - 8215:3000
```

Once you've made the adjustments, save the file again with CTrl+O, then press Enter to confirm, and use CTRL + X to exit.

Run Our Server

To run the server, and get it going, we'll use the following command:

`docker-compose up -d`

After you see the 'done' and are brought back to the prompt in the terminal, you can check the logs with the command:

`docker-compose logs -f mstream`

This will show you the logs as they update, and once you see success or done messages in the logs, you can use CTRL + C to end the logging output, and open the browser.

In the browser go to the IP address of your MStream server, and the port you set.

In my case, I went to

`http://192.168.10.25:8215`

You should see the MStream UI with a list of all of your music on the screen. On the left you'll see a menu of options, center is a list of available music, and on the right is a playlist window. click on on items in the music list, will move them to the playlist in the order they are clicked. Think of this as a quick (ad-hoc) playlist.

You are now up and running.

Check out the video for the Graphical User Interface walk-through, and enjoy your music!

Support my Channel and Content

Support my Channel and ongoing efforts through Patreon:

https://patreon.com/awesomeopensource

Ampache

Ampache Music Streaming Server

https://www.youtube.com/embed/PpUkgQSi_RQ

Why Run Your Own Music Streaming Server?

I've been an iTunes user for more than a decade, in fact going on two decades now. I liked it, and still like it somewhat, but with the move to split everything out, the interface is painful to me. Getting music to my local machine is a hassle. I also have become less and less pleased with the thought of someone else holding my data (in this case my music). I paid for it, I own it, and I've collected it for more than 35 years now. Why should I not have it locally?

Before I get comments, yes, I know I can download it, but that goes right back to the hassle.

I've bean looking for solutions to host and stream from my own home server. It wasn't hard to find several viable options, but Ampache kept popping up. So here are my show notes with Installation, first run wizard, and a few admin items you'll likely want to take care of early on as you setup your own Ampache streaming server.

Installation

WE will be using Docker to install Ampache from the official ampache docker image. Docker is an excellent tool for running extremely minimal virtual servers so there's not all the bloat of a full virtual OS install.

I'm using Ubuntu 18.04 (Zorin 15.2), so I'll be using the Docker-CE installation instructions found on Digital Ocean's site to install Docker. If you don't have Docker installed yet, feel free to jump over there and get it installed first. If you have a different OS (Newer or older, different Linux distro, Windows, or MacOS) just google around for the instructions to install Docker-CE on your OS. :

Installing Ampache, once Docker is installed is really quite straight-forward. A few things we want to keep in mind:

1. We want to make sure we setup proper port mapping from our host to our docker container where Ampache will be running. Ampache runs on port 80 (the common port for most web sites). In the Ampache instructions for their docker they only give information to map port 80.
2. This, however is an unsecured (no SSL) port, so we also want to map a port from our host to our docker container for port 443.
3. Finally, we need to map a volume (a storage location on our host) to `/media` on our docker container. This is where we will keep our music.

I mapped `/home/brian/Music/ampache` to `/media` and this has worked well for me. So let's begin.

First we'll setup the directory for our music storage.

Starting from my home directory on my server, `/home/brian`, I'll change directory into Music, then make a new directory called `ampache`.

```
cd Music
```

```
mkdir ampache
```

Done! Not too bad, right?

Next, we want to put some of our music in this folder. You can use the `cp` command to copy music from one folder to another, or the `mv` command to move it (cut / paste), or `scp` to copy music from a remote machine. Of course, if you are running a GUI (Graphical User Interface) on your server, you can use all of the normal methods for getting music to the `ampache` directory you just made.

Now we need to get Ampache installed. For this we use a very straight-forward one liner in Docker. Depending on whether you've added your user to the docker group or not, you may need to use `sudo` to run these commands. I'll put them with sudo, but if you are part of the docker group, feel free to leave it off.

```
sudo docker run -d --name=ampache -v /home/<your user>/Music/ampache:/media -p 8051:80 -p 8543:443 ampache/ampache
```

What we are doing above is explained like this:

`sudo` -> do this as a super user with root permissions

`docker run -d` -> fun this docker container as a daemon (a service that runscontinuously in the background)

`--name=ampache` -> call this container "ampache" so I can find it in a list of containers.

`-v /home/<your user>/Music/ampache:/media` -> map my host machine folder (and yes you need to put your actual username for the host server where it says "<your user>") to the "/media" directory in my container.

`-p 8051:80` -> map my host port 8051 to the container port 80. NOTE: this step isn't required if you aren't running anything else on this server that uses port 80 already.

`-p 8543:443` -> map my host port 8543 to the container port 443. NOTE: this step isn't required if you aren't running anything else on this server that uses port 443 already.

On the two NOTES above, keep in mind, it may still be useful to point a non-standard port to 80 and 443 in the container as it leaves you room to use NginX or another Proxy server when you want to reach your music server more easily and securely from outside your home network.

`ampache/ampache` -> pull down the official ampache image to build our container from.

Extra Flags for Docker

You may want to add one more flag and option.

`--restart=unless-stopped` -> this option will restart the container if it should crash for some reason, or if the system should do an automatic reboot during the night. It won't restart on it's own, however, if you intentionally stopped it using the `docker stop <container name>` command.

First Run Wizard

After the docker command finishes running you shoudl be able to reach your server at it's IP address and whatever port you mapped to the container port 80. For me it is http://192.168.7.125:8051 (your's will likely be different).

The first time you visit the running site, you'll run through a "First Run Wizard". This wizrd helps you setup an administrative useer, the database (mysql included in the image we pulled) connection, and just makes sure everything we need is installed.

1. Select Language
2. Make sure all checks are good - but there will be a warning about files larger than 20 MB not being allowed - don't sweat that one.
3. MySQL Setup - Leave the defaults all the way to the checkboxes. You want to check the option for "Create Database User". When you check it you'll get two new fields. You can change the username if you want, but remember it. Then add a password and remember it.
4. On the next screen, you need to fill your user you added, and the password for that user. Scroll down and choose if you want to allow transcoding. You can check all the API boxes, or just take the defautls. Then click 'Create Config' (bottom center).

5. Make your first user (admin level account). Click 'Create Account'
6. Scroll to the bottom, and click 'Update Now'
7. You should get 'No Update Needed'. Click the link for Return to Main Screen' and login with your admin user from step 5.

Add Music

Click on Admin >> Add Catalog. From here, give you catalog a name, then move down and make sure 'local' is selected. Finally, give the path of `/media` for the location of your music. Click 'Go' and the music should begin filling your Ampache system pretty quickly.

You can now navigate to 'Songs' on the main view (the little headphones icon), and see your list of music as it's being built.

Conclusion

This is a very brief overview of what Ampache can do, and how to install and get it running. So before you invest hours or days in getting your music and playlists setup, make sure to get everything else setup just like you want it in your server. That way if you bork it, you can nuke and pave an dnot lose tons of effort in the process.

Identifying Music

Install MusicBrainz in Docker

https://www.youtube.com/embed/FVRSoAmNLRU

Getting your music collection tagged properly, and finding the album art was a tedious, manual process for many years, but also a source of pride in owning a well defined digital music collection. Today it's been made much easier thanks to tools like MusicBrainz Picard. This revolutionary tool can scan your music collection, match it up using the wave forms of the music, and identify it with a high degree of accuracy. Using massive databases of music it's able to add the proper ID3 tags, and help find the right album art for the album a song came from, even if it's just 1 song. The other benefit to albums, artists, and other data is it offers another way to easily and quickly view your music in organized groupings.

Combine this with a streaming application server like Navidrome, Ampache, AirSonic, FunkWhale, or MStream, and you have an incredible set of tools for enjoying your music on your terms.

Installation of Docker and Docker Compose via a Simple Script

You can easily install Docker-CE, Docker-Compose, Portainer-CE, and NGinX Proxy manager by using this quick install script I created and maintain on Github. Just use the command:

```
wget https://gitlab.com/bmcgonag/docker_installs/-/raw/main/install_docker_nproxyman.sh
```

To download the script to your desired host.

Change the permissions to make the script executable:

```
chmod +x ./install_docker_nproxyman.sh
```

and then run the script with the command:

```
./install_docker_nproxyman.sh
```

When run, the script will prompt you to select your host operating system, then will ask you which bits of software you want to install.

Simply enter 'y' for each thing you want to install.

At some point, you may be asked for your super user (sudo) password as well.

Allow the script to complete installation.

At this point, you might want to log out and back in, as this will allow you to use the `docker` and `docker-compose` commands without the need of sudo in front of them.

What You'll Need

- A Server or Machine with your Music Collection on it
- Docker and Docker-Compose (if you want to run it via Docker like I do)
- About 10 minutes of your time

Create our Docker Compose

First, let's create a folder for our installation:

`mkdir -p docker/picard`

Now, let's move into that folder, and create our docker-compose.yml file that will help us pull down the image and get everything running so we can access MusicBrainz in the browser.

`cd docker/picard`

`nano docker-compose.yml`

Now, copy the code snippet below, and paste it into your new docker-compose.yml file.

```
version: '3.3'
services:
    musicbrainz:
        container_name: musicbrainz
        ports:
            - '7120:5800'
        volumes:
            - './config:/config:rw'
            - '/mnt/data/media/music:/storage:rw'
        image: mikenye/picard
```

In the file, you'll want to change the left side of the volume mapping for '/storage' to be the path to your data location. In my case my music is stored in /mnt/data/media/music. Your path is very likely to be different, so definitely make that change, then save and exit.

Save the file with CTRL + O, then press Enter to confirm, and exit the nano editor with CTRL + X.

Now we can run our application with the command:

```
docker compose up -d
```

If you have an older version of docker compose, you may need to add a hyphen between docker and compose to get the command to run...like this:

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

Once running, you'll see the image being pulled down, the container will start, and you'll be put back at the prompt in your terminal.

Now, open your favorite modern browser, and navigate to the IP address of your host machine running docker, and the port 7120.

You should be greeted with the MusicBrainz Picard user interface.

You may see a pop-up in the interface about a new version being available. This is ok, just click the Cancel button for now.

Make sure to check out the video to see how to select, identify, and save the new data for your music collection using MusicBrainz Picard!

Support My Channel and Content

Support my Channel and ongoing efforts through Patreon:
https://www.patreon.com/awesomeopensource

Music Assistant

Install Music Assistant

https://www.youtube.com/embed/dIk-6wDtTB8

Setup a Server

In most self hosted applications these days, you'll need a machine to act as a server. This can be a machine in your home / business (such as an old laptop or desktop, a Raspberry PI or Single Board Com#puter (SBC), or even the computer you're reading this article from), a VM / Container hosted on one of your machines, or a VPS (Virtual Private Server) hosted by companies like RackNerd, Digital Ocean, Linode, Vultr, and so many more. Regardless of which option you choose, you'll want to do a few things to get the server setup properly.

Install updates to our Server

Ubuntu / Debian

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

RedHat / CentOS / Fedora / Alma / Rocky

```
sudo dnf update -y
```

Add a non-root / sudo user on the server

Generally, when you setup a new server, the VPS (Virtual Private Server) service sets up a default "root" user for you. It's considered unsafe to do everything as "root", so let's setup a non-root user who has super user (sudo) privileges.

`adduser <username>` You'll be prompted to enter and confirm a password for this user. You'll also be asked for some user information like Name, etc, but this is not required information. At the end, confirm the entries, and you'll have your new user.

Next, we need to add the user to the super user group.

Ubuntu / Debian

```
usermod -aG sudo <username>
```

RedHat / CentOS / Fedora / Alma / Rocky

```
usermod -aG wheel <username>
```

Now, you can log out of the system, and log back in as your new non-root super user.

Install Docker and Docker Compose

Fortunately, there is a single line command that will install both Docker and Docker Compose for us on most Linux based distributions.

We need the 'curl' utility to get this to work, so if you don't have it, you'll want to install it first with

Ubuntu / Debian

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

RedHat / CentOS / Fedora / Alma / Rocky

```
sudo dnf install curl -y
```

Next, we'll run the command to install Docker and Docker Compose:

```
curl https://get.docker.com | sh
```

You may be prompted to enter your super user password, so be ready for it. Once you do, the install should proceed.

Once complete. we want to add our user to the 'docker' group so we can do `docker` and `docker compose` commands without having to type in `sudo` each time.

```
sudo usermod -aG docker <username>
```

Now we'll logout / exit the session, and log right back in so the updated group will take effect.

Stand Alone Install of Music Assistant

First, let's create a folder on our server for our install:

```
mkdir -p docker/musicassistant
```

Now we'll move into that folder

```
cd docker/musicassistant
```

Now create a new file with a text editor (I like to use nano):

```
nano compose.yaml
```

Copy the yaml code block below, and paste it into the editor window.

```
---
services:
  music-assistant-server:
    image: ghcr.io/music-assistant/server:latest
    container_name: music-assistant-server
    restart: unless-stopped
    network_mode: host
    volumes:
      - ./music-assistant-server/data:/data/
      - </path/to/your/music/collection>:/media
    cap_add:
      - SYS_ADMIN
      - DAC_READ_SEARCH
    security_opt:
      - apparmor:unconfined
    environment:
      # default=info, possible=(critical, error, warning, info, debug)
      - LOG_LEVEL=info
```

In this code block, you'll want to make a couple of changes to make sure the application works for your music collection (if you are planning to use a local collection).

In this case, if you are wanting to use local music (whether on the same disk as the container you are running, or on a mounted volume such as NFS or SMB), then you should change the volumen mapping on the left side of the colon ':' where it says

`<path/to/your/music/collection>`

and enter the actual proper path to link your music collection to the container's internal folder labeled `/media` .

If you are not including locally hosted music, then you can remove that line completely.

Use CTRL + O to save your changes, press Enter to confirm, and exit the nano editor with CTRL + X.

Start Our Application in Docker Compose

First, let's pull down the images needed to run our container (tiny virtual machine).

```
docker compose pull
```

Once that pulls down, we can start the application with:

```
docker compose up -d && docker compose logs -f
```

This is actually two commands concatenated together with the `&&` symbol. The first command tells docker to bring up the container, and the second command says we want to follow `-f` the logs as the application starts. We are just looking quickly for any obvious error messages. Once the logging slows, we can stop following the logs with CTRL + C.

Now open your browser and type in the IP address of your host server, and the port 8095. For instance I went to:

http://192.168.10.36:8095

You should be greeted by the Music Assistant system, up and running.

!!! A Word About Security

Music Assistant does not use a user model, and there is no login screen for this application. That said, it would be a very bad idea to host your Music Assistant system directly on the open internet with a domain name and no other protections. This type of implementation would open your home music system, any device (player providers), your connected online music accounts (music providers), and your local music collection to the world! This is not a good thing.

If, however you run a reverse proxy like Pangolin, then you can mitigate this by setting up the Pangolin authentication in front of the proxied services / sites. I have not personally set this up, but would be willing to do a video on it if you are interested. Comment on the linked video to let me know.

Support My Channel and Content

Support my Channel and ongoing efforts through Patreon:

https://www.patreon.com/awesomeopensource

Buy me a Beer / Coffee:

https://paypal.me/BrianMcGonagill