MovieServer

Documentation

By Anirudh Sevugan

Contents

Chapter 1
Installation

MovieServer is a free, open-source tool available on GitHub that serves as a template for a fully functional movie server which streams movies on LAN, and with some extra configuration, anywhere in the world.

However, the tool is a bit complicated at first glance. So we're going to simplify things for you.

1. **Use MovieServer Installer (Recommended)**
   The MovieServer installer is a piece of software that installs all dependencies ***and*** the source code as well (pulled from GitHub), that is available for Windows, Mac, and Debian-based distros of Linux (e.g, Ubuntu). The dependencies are Node.js (***Required***) and Jellyfin (Optional),
   Visit https://github.com/A-Star100/MovieServer/releases/, and choose the release with **Latest** right next to it. Then, choose the installer for your platform (for Windows, it is a batch script, for macOS and Linux (separate installers), it is a shellscript). On **Windows only**, you will need to bypass the Windows Smartscreen warning displayed by clicking More… > Run Anyway.

2. **Install tools manually**
   First, visit https://brew.sh if you are on macOS or visit https://chocolatey.org if you are on Windows, and view installation instructions.

   macOS/Linux Steps
   Once Homebrew is installed, type
   ```
   brew install node
   ```
   to install Node.js, which is a mandatory dependency. Optionally, you may also install ffmpeg (for HLS and MPEG-DASH stream creation, and offline video and audio file conversion), and Jellyfin.

   To do so, type
   ```
   brew install ffmpeg
   ```
   and
   ```
   brew install --cask jellyfin
   ```

   Windows Steps

```
choco install nodejs
```

to install Node.js, which is a mandatory dependency. Optionally, you may also install ffmpeg (for HLS and MPEG-DASH stream creation, and offline video and audio file conversion)
To do so, type:

```
choco install ffmpeg
```

and

```
choco install jellyfin
```

Chapter 2
Setup

Now that you have installed MovieServer's dependencies, let's take a look at some basic setup you'll need to do.

Open the MovieServer directory (~/MovieServer on macOS/Linux, C:\Users\YourUsername\MovieServer on Windows - **Note:** Replace *YourUsername* with your username on Windows), and go to the ***server.js*** file. There you will find many file paths. You will ***need*** to change these file paths according to your setup.

An example for the file path that will work with your setup in *server.js* is:
/Users/*YourUsername*/*example.txt* **(on macOS/Linux)**
or
C:\Users\*YourUsername**example.txt* **(on Windows)**

We assume you know what file paths to replace, but if you don't, here is an example for the file paths to replace for .key and .crt files (assuming you have them in a *certs* directory located outside of the MovieServer directory:

/Users/*YourUsername*/*certs/example.key* **(on macOS/Linux)**
/Users/*YourUsername*/*certs/example.crt* **(on macOS/Linux)**
or
C:\Users\*YourUsername**example.key* **(on Windows)**
C:\Users\*YourUsername**example.crt* **(on Windows)**

Now that you've changed most of the file paths, you'll need to change the IP addresses. Let's address these one by one.

Inside the MovieServer directory, open ***index.html***, and try to find mentions of IP addresses by hitting Command+F on Mac or hitting Control+F on Windows, specifically the IP address *192.168.254.137* (the IP address that was used in creation of the repository) and replace them
On Mac,

Open **System Preferences** from the Apple menu > **System Preferences**.
Go to **Network**.
Select the active connection (Wi-Fi or Ethernet) on the left panel.

The **IP address** will be displayed on the right under **Status**.

Look for the **IPv4 Address** (usually formatted like XXX.XXX.XXX.XXX) under your network adapter (typically "Ethernet" or "Wi-Fi").

On Windows,
Open **Settings**: Start > Settings > Network & Internet.
Select either **Wi-Fi** or **Ethernet** (depending on your connection).
Click on the connected network, and you'll see the IP address listed under **Properties**.

Look for the **IPv4 Address** (usually formatted like XXX.XXX.XXX.XXX) under your network adapter (typically "Ethernet" or "Wi-Fi").

Replace any mentions of *192.168.254.137* with the IPv4 address you found in *index.html*.

Then, inside the MovieServer directory, go inside the *misc* > *websocket-chat* directory, for the WebSocket chat server that you will use.

Then go to the ***server.js*** file inside there, and replace any mentions of *192.168.254.137* with the IPv4 address you found.

Then, inside the websocket-chat directory, go inside the *public* directory, and do the same for the ***chat.js*** and ***index.html*** files.

Then, go back to the MovieServer directory, and **optionally**, do the same for the ***ifofflineworker.js*** file, if it even exists.

If you're planning to use **Docker**, do the same for the ***dockerserver.js*** file and go to the next chapter

If you're **NOT** using **Docker**, skip the next chapter and go to the one after that.

⚠️ The Docker setup will not work well on Windows, you may need to modify the *dockerfile* to use a Windows VM instead of a Linux one. ⚠️

Chapter 3
Docker Guide

First, make sure Docker is installed on your computer (if not, visit https://docker.com and go through the installation). Open Terminal and type in:

`cd ~/MovieServer` -(or wherever your MovieServer directory is located)-

```
docker build -t movieserver-image .
```

Now that you have built your Docker image out of the *dockerfile*, we need to build the container for MovieServer to run in. Type in:

```
docker run -d \ -p 8080:8080 \ -p 8000:8000 \ -p 4443:4443 \ -p
3000:3000 \ --name movieserver-container \ movieserver-image
```

You have now exposed ports 8080, 8000, 4443, and 3000 (ports used with MovieServer) for your Docker container.

Then type:

```
docker exec -it movieserver-container /bin/bash
```

Now use the following to figure out the IPv4 address of your container:

```
ip addr show eth0
```

Go ahead and install *nano* and *vim* in your Docker Container by using:

```
apt-get update && apt-get install nano vim
```

Choose your preferred editor (this example uses *vim*), and type:

```
vim dockerserver.js
```

or if you are using *nano*:

```
nano dockerserver.js
```

Go ahead and replace any file paths or IP addresses however you please, then save your edited file.
Now start the servers by typing:

```
node server.js
```

You should be ready to go.

Jellyfin Media Server and/or Jellyfin Client is not included with the dockerfile. You will have to modify it in order to add support.

Chapter 4
Jellyfin Setup

⚠️ Before you start, *please* note that Jellyfin has some recurring issues that have not been fixed. Some issues straight-up hide full films. *Please* visit https://jellyfin.org/contact to visit the Jellyfin troubleshooting page (Jellyfin will ignore your issue on GitHub Issues if you troubleshoot, etc), and it is also very hard to transfer Jellyfin metadata, so if you are close to getting a new computer, wait until that happens to avoid a huge amount of issues.⚠️

Now that we've covered that, let's start using Jellyfin!
Visit https://jellyfin.org and use the version of Jellyfin for your operating system.

Once you've installed the Jellyfin Media server app (or compiled it from source if that's what you wanted), go ahead and start it and go through the Jellyfin wizard.

Once you encounter a selection dialog with a **Folders** label on top, click on the + icon, and choose the file path of wherever your full-length movies are stored.

Make sure to disable metadata fetching if the full-length movies are your personal media, as ratings, images, titles, etc, will be found from the web, and your movies may be treated and displayed as other movies if their name matches a movie on ImDb, etc.

Once you're done, you should see every movie. If you don't, try these steps.

1. **Change filenames**
   Jellyfin often hides trailers at random, so if you have any folder or file with *trailer* in its name, try renaming that portion to something like *preview*, then click on the hamburger menu, click *Dashboard*, go to *Media Libraries*, then click *Scan All Libraries*. This will seek for changes of filenames, and it *should* display the previously hidden movies. If that doesn't work, follow the next step.

2. **Remove excessive nested folders**
   Jellyfin often has trouble finding files in many nested folders, it is better to keep the files in only *one* subfolder, or just keep them all in root (although that can easily generate clutter). If even ***that*** doesn't work, visit https://jellyfin.org/contact and locate their troubleshooting server. Create a Matrix account and start troubleshooting (again, Jellyfin

will ignore issues on GitHub).

Once you've succeeded in getting all of your movies on Jellyfin, you can go ahead and replace any links related to the Jellyfin server. (mostly in ***index.html*** (the one outside of the chat directory), and ***chat.html***)

Chapter 5
HLS and MPEG-DASH with ffmpeg

FFmpeg command for making an adaptive bitrate (360p, 480p, 720p, 1080p) HLS stream with master playlist:

```
ffmpeg -i input.mp4 \
  -filter_complex "\
    [0:v]split=4[1080p][720p][480p][360p];\
    [1080p]scale=-2:1080[v1080p];\
    [720p]scale=-2:720[v720p];\
    [480p]scale=-2:480[v480p];\
    [360p]scale=-2:360[v360p]" \
  -map "[v1080p]" -c:v:0 libx264 -b:v:0 5000k -maxrate 5350k -bufsize
7500k -c:a aac -b:a 128k -ac 2 -y -hls_time 6 -hls_playlist_type vod
-hls_segment_filename "hls/1080p/segment_%03d.ts" hls/1080p.m3u8 \
  -map "[v720p]" -c:v:1 libx264 -b:v:1 3500k -maxrate 3850k -bufsize
5250k -c:a aac -b:a 128k -ac 2 -y -hls_time 6 -hls_playlist_type vod
-hls_segment_filename "hls/720p/segment_%03d.ts" hls/720p.m3u8 \
  -map "[v480p]" -c:v:2 libx264 -b:v:2 2000k -maxrate 2200k -bufsize
3000k -c:a aac -b:a 128k -ac 2 -y -hls_time 6 -hls_playlist_type vod
-hls_segment_filename "hls/480p/segment_%03d.ts" hls/480p.m3u8 \
  -map "[v360p]" -c:v:3 libx264 -b:v:3 1000k -maxrate 1100k -bufsize
1500k -c:a aac -b:a 128k -ac 2 -y -hls_time 6 -hls_playlist_type vod
-hls_segment_filename "hls/360p/segment_%03d.ts" hls/360p.m3u8 \
  # Create HLS Master Playlist
  -f hls -y hls/master.m3u8
```

FFmpeg command for making an adaptive bitrate (360p, 480p, 720p, 1080p) MPEG-DASH stream with master manifest:

```
ffmpeg -i input.mp4 \
  -filter_complex "\
    [0:v]split=4[1080p][720p][480p][360p];\
    [1080p]scale=-2:1080[v1080p];\
    [720p]scale=-2:720[v720p];\
    [480p]scale=-2:480[v480p];\
    [360p]scale=-2:360[v360p]" \
  -map "[v1080p]" -c:v:0 libx264 -b:v:0 5000k -maxrate 5350k -bufsize
7500k -c:a aac -b:a 128k -ac 2 -y -f dash -dash_segment_filename
"dash/1080p/segment_%03d.m4s" dash/1080p.mpd \
  -map "[v720p]" -c:v:1 libx264 -b:v:1 3500k -maxrate 3850k -bufsize
5250k -c:a aac -b:a 128k -ac 2 -y -f dash -dash_segment_filename
"dash/720p/segment_%03d.m4s" dash/720p.mpd \
  -map "[v480p]" -c:v:2 libx264 -b:v:2 2000k -maxrate 2200k -bufsize
3000k -c:a aac -b:a 128k -ac 2 -y -f dash -dash_segment_filename
"dash/480p/segment_%03d.m4s" dash/480p.mpd \
  -map "[v360p]" -c:v:3 libx264 -b:v:3 1000k -maxrate 1100k -bufsize
1500k -c:a aac -b:a 128k -ac 2 -y -f dash -dash_segment_filename
"dash/360p/segment_%03d.m4s" dash/360p.mpd
```

More…

What *you* can do

There are ***so many things*** you could add to MovieServer. If your changes work well, you could even try making a pull request on GitHub and see if I (the author of this book and the software) agrees. I would love to see your additions!

You could, for example, make a **mobile chatting app** for the chat server, which I struggled to do, with Flutter, App Inventor, Kodular, everything I tried either half-worked, half-didn't, or failed completely. Maybe you'll have better luck, especially if you're a mobile app developer.

You could also enhance the UI, by adding support for more buttons.

You could even ***add a load balancer,*** which will be very helpful with handling lots of load, since MovieServer is a server-side app, tools like Hostinger, Wix, GitHub Pages, etc, won't work.

You'll probably need **Google Cloud**, or you'll need to configure **port forwarding or a proxy** (to expose the server anyway).

Just ***imagine*** the possibilities.

Anyway, that's it! Hope you enjoyed reading this book and I hope you enjoy using MovieServer to serve your movies on the internet.

MovieServer

Documentation

By Anirudh Sevugan

In this book, you'll learn how to use ***MovieServer***, an open-source, easy-to-use setup that has the capability to serve thousands of movies on-demand, with a bit of configuration. Learn how to use MovieServer on your device, in Docker, and even setup software like Jellyfin to give an even better movie-streaming experience.

You'll also get versatile compatibility with HLS and MPEG-DASH streaming technologies out-of-the-box, *and* learn how to use it!

Learn more about MovieServer at https://github.com/A-Star100/MovieServer!