STREAMING LIVE VIDEO WITH A RASPBERRY PI CAMERA OVER RTMP

March 20, 2020

RTMP is a protocol for high performance streaming of audio, video, and data that was developed by Adobe (Macromedia at the time) for flash applications. It is now used by most popular live streaming platforms including Twitch, YouTube Live, and Facebook Live.

RTMP is implemented over 3 components:

- **encoder** - Captures and encodes a live stream before publishing to an RTMP server.
- **server** - Potentially transcodes the stream and hosts it for clients to fetch and render
- **client** - The reciever of the stream which renders it for the user

We will leverage a Raspberry Pi Camera as our source, gstreamer as our encoder, nginx + nginx-rtmp-module for our server, and video.js for the client. There are several other tutorials I found online that try to explain how to set this up with a Raspberry Pi, but none of them worked for me without modification. Here is what I ended up with.

Prerequisites

Hardware

- Raspberry Pi Zero W - should work with any other Pi though
- Raspberry Pi Camera Module V2

Software

- Raspbian Buster

I'll assume that you have Raspbian Buster installed on your Pi and you have SSH access to it.

Enable The Camera

Enter raspi-config and select: **Interfacing Options -> Camera -> Yes**

```
sudo raspi-config
```

Install GStreamer

```
sudo apt install gstreamer1.0-tools gstreamer1.0-plugins-bad gstreamer1.0-plugins-base
gstreamer1.0-plugins-good gstreamer1.0-omx-rpi gstreamer1.0-omx-rpi-config
```

Install NGINX

Normally the server would be hosted on a different machine than the encoder, but for our purposes we'll keep everything self-contained on the Raspberry Pi

```
sudo apt install libnginx-mod-rtmp nginx-full
```

Configure NGINX

We will enable the rtmp module for nginx and configure it to republish the stream over the HTTP Live Streaming protocol. HLS is a streaming protocol developed by Apple which is widely supported by all major desktop browsers and mobile devices.

Place the following into **/etc/nginx/modules-enabled/rtmp.conf**

```
# /etc/nginx/modules-enabled/rtmp.conf
# RTMP configuration
rtmp_auto_push on;
rtmp {
    access_log /var/log/nginx/rtmp_access.log;
    server {
        listen 1935; # Listen on standard RTMP port
        ping 30s;
        chunk_size 4096;
        notify_method get;

        publish_time_fix off;

        application live {
            live on;
            record off;
            hls on;
            hls_path /var/www/html/hls;
            hls_fragment 3;    # i use 2
            hls_playlist_length 60;   # i use 2
            deny play all;
        }

    }
}
```

Place the following into **/etc/nginx/sites-enabled/default**

```
# /etc/nginx/sites-enabled/default

server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/html;

    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
            try_files $uri $uri/ =404;
    }

    location /hls {
        # Disable cache
        add_header Cache-Control no-cache;

        # CORS setup
        add_header 'Access-Control-Allow-Origin' '*' always;
        add_header 'Access-Control-Expose-Headers' 'Content-Length';

        # allow CORS preflight requests
        if ($request_method = 'OPTIONS') {
            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Max-Age' 1728000;
            add_header 'Content-Type' 'text/plain charset=UTF-8';
            add_header 'Content-Length' 0;
            return 204;
        }

        types {
            application/vnd.apple.mpegurl m3u8;
            video/mp2t ts;
        }


    }
}
```

Create the hls directory:

```
sudo mkdir -p /var/www/html/hls
sudo chown pi:pi /var/www/html/hls
```

Restart NGINX:

```
sudo systemctl restart nginx
```

Set up the GStreamer Pipeline

GStreamer is the swiss-army-knife of streaming media. While there is a gstreamer element for the Raspberry Pi Camera, it does not come included in the Raspbian distribution and would require compilation. It is much easier to just get started by using the **raspivid** utility included in Raspian and pipe the output to GStreamer.

We will capture 720p 30fps video with a 2Mb bitrate. raspivid outputs h264 encoded frames, which we will then pipe into gstreamer to wrap into a Flash Video container before uploading to our RTMP server.

Place the following into **/usr/local/bin/gst-rpi-stream**. Make it executable.

```
#!/bin/bash
# /usr/local/bin/gst-rpi-stream

raspivid -t 0 -b 2097152 -w 1280 -h 720 -fps 30 -n -o - | gst-launch-1.0 fdsrc ! vide
o/x-h264,width=1280,height=720,framerate=30/1,profile=high,stream-format=byte-stream
 ! h264parse ! queue ! flvmux streamable=true ! rtmpsink location=\"rtmp://localhost:1
935/live/picam\"
```

Make it executable:

```
sudo chmod +x /usr/local/bin/gst-rpi-stream
```

Create SystemD Service for the GStreamer Pipeline

We will create a systemd unit file to manage the gstreamer process. We'll configure it to automatically restart the stream if it goes down.

Place the following into ***/lib/systemd/system/gst-rpi-stream.service**

```
[Unit]
Description=RPI GStreamer RTMP Source

[Service]
Type=simple
ExecStart=/usr/local/bin/gst-rpi-stream
Restart=on-failure
RestartSec=5s

[Install]
WantedBy=multi-user.target
```

Enable and start the service:

```
sudo systemctl daemon-reload
sudo systemctl enable gst-rpi-stream
sudo systemctl start gst-rpi-stream
```

Make the Website

Let's be lazy and make our site be a fullscreen video that autoplays when you load it.

Place the following into **/var/www/html/index.html**

```
<!DOCTYPE html>
<meta charset="UTF-8">
<html lang="en">
    <head>
        <title>Pi Cam</title>
        <link href="https://unpkg.com/video.js/dist/video-js.css" rel="stylesheet">
    </head>
    <body>
        <div width="100%" height="100%">
        <video id="videojs-player" class="video-js vjs-default-skin vjs-layout-huge" preload="auto" data-setup="{}" muted fluid="true" controls="true" autoplay="autoplay">
            <source src="/hls/picam.m3u8" type="application/x-mpegURL">Your browser does not support the video tag.
        </video>
        </div>
        <script src="https://unpkg.com/video.js/dist/video.min.js"></script>
        <script src="https://unpkg.com/videojs-flash/dist/videojs-flash.min.js"></script>
        <script src="https://unpkg.com/videojs-flvjs/dist/videojs-flvjs.min.js"></script>
    </body>
</html>
```

And remove the default index:

```
sudo rm /var/www/html/index.nginx-debian.html
```

Access the Stream

Head to **http://{YOUR_PI_IP_ADDRESS}** and you should see your live stream!