Setup-Anleitung zum Aufbau eines Live-Streaming-Servers

Diese Anleitung ist Schritt für Schritt auszuführen. Für die Installation sind folgende Voraussetzungen erforderlich:

- Grundkenntnisse von dem Betriebssystem Ubuntu
- Umgang mit einem einfachen Text-Editor (HTML)

```
// ********************************************************************************
Autor: Gerhard Peilstoecker
Copyright (c) 2019 - 2024 [https://it-service-peilstoecker.de]
Aenderungen:
05.01.2021 "mein.streaming-key" nach "mein.stream-key" geaendert
06.09.2021 nginx Version von 1.15.1. nach 1.18.0 geaendert
10.08.2023 Edition v.1.2 als PDF-Download bereitgestellt
// ********************************************************************************
```

Voraussetzung für die Installation auf Ubuntu, vorher installierte Programme:

a) VLC-Media-Player
b) OBS Open Broadcast Software
c) Zip-Entpacker
d) hier genutztes Linux-Betriebssystem Ubuntu 18.03.

Tipp: Die hier angegebenen Befehlszeilen können einfach mit **[Strg]**+**[c]** kopiert werden und mit **[Strg]**+**[Shift]** +**[v]** in das Konsolen-Fenster eingefügt werden.

1. Vor der Installation solltest du dein System auf den aktuellen Stand bringen.

Mit der Tastenkombination **[Strg]**+**[Alt]**+**[T]** öffnest Du die Konsole und gibst dazu folgende Befehlszeile ein.

```
$ sudo apt-get update && sudo apt-get dist-upgrade
```

Gib Dein Passwort ein und bestätige mit **[Enter]**. Mit der Eingabe **[J]** startet das Update und Dein System wird auf den neusten Stand gebracht.

2. Die folgenden notwendigen Tools sind als nächstes so zu installieren:

```
$ sudo apt-get install build-essential libpcre3 libpcre3-dev libssl-dev
```

3. Den Webserver enginx (gesprochen “engine-X”) herunterladen. Er ist derzeit einer der leistungsfähigen Webserver.

```
$ wget http://nginx.org/download/nginx-1.18.0.tar.gz
```

4. Quellen für das RTMP-Modul herunterladen

```
$ wget https://github.com/sergey-dryabzhinsky/nginx-rtmp-module/archive/dev.zip
```

5. Entpacken und in das Server-Verzeichnis wechseln

```
$ tar -zxvf nginx-1.18.0.tar.gz
$ unzip dev.zip
$ cd /nginx-1.18.0
```

6. Erstellen und Installieren des nginx-Server mit dem eingebundenem RTMP-Modul.

```
$ ./configure --with-http_ssl_module --add-module=../nginx-rtmp-module-dev

$ make

$ sudo make install
```

7. Notwendige Verzeichnisstruktur anlegen für die Fragmente des Video-Streams.

```
$ sudo mkdir /HLS

$ sudo mkdir /HLS/live

$ sudo mkdir /HLS/mobile

$ sudo mkdir /video_recordings

$ sudo chmod -R 777 /video_recordings
```

Wird ubuntu mit Firewall betrieben (Standard ist aus), dann muss folgende Regel eingefügt werden.

```
$ sudo ufw allow 80 $ sudo ufw allow 1935 sudo $ ufw enable
```

8. Der Server ist installiert im Standardverzeichnis: /usr/local/nginx und wird nun gestartet. Den Browser öffnen und http:/// oder “localhost” eingeben. Es erscheint die Startseite des Servers. (Die IP-Adresse des Servers wird über das Komando $ ifconfig ausgegeben.)

Server **starten** mit:

```
$ sudo /usr/local/nginx/sbin/nginx
```

Server **stoppen** mit:

```
$ sudo /usr/local/nginx/sbin/nginx -s stop
```

9. Installieren von FFmpeg zum Konvertieren und Packen des Streams.

```
$sudo apt-get install software-properties-common

$sudo add-apt-repository ppa:kirillshkrogalev/ffmpeg-next
```

10. Paketliste aktualisieren und installieren

```
$sudo apt-get update

$sudo apt-get install ffmpeg
```

11. Der Server konfigurieren vorher die original Konfigurationsdatei “nginx.conf “sichern.

```
$ sudo cp /usr/local/nginx/conf/nginx.conf
/usr/local/nginx/conf/nginx.conf.original

$ sudo nano /usr/local/nginx/conf/nginx.conf
```

12. Den gesamten Inhalt in die vorhandene Datei “nginx.conf” kopieren (überschreiben) und speichern. Die

Platzhalter “meine-ip sind mit der IP-Adresse deines Server zu ersetzten. Beispiel: 192.168.254.178

Für “mein-stream-key”, dein frei erfundener key z.B. tuy31nevyqz9 einsetzen.

```
worker_processes  1;

error_log  logs/error.log debug;

events {

worker_connections  1024;

}

rtmp {

server {

listen 1935;

allow play all;

#creates our "live" full-resolution HLS videostream from our incoming encoder
stream and tells where to put the HLS video manifest and video fragments

application live {

allow play all;

live on;

record all;

record_path /video_recordings;

record_unique on;

hls on;

hls_nested on;
```

```
hls_path /HLS/live;

hls_fragment 10s;

#creates the downsampled or "trans-rated" mobile video stream as a 400kbps,
480x360 sized video

exec ffmpeg -i rtmp://meine-ip:1935/$app/$name -acodec copy -c:v libx264 -preset
veryfast -profile:v baseline -vsync cfr -s 480x360 -b:v 400k maxrate 400k -
bufsize 400k -threads 0 -r 30 -f flv rtmp://meine-ip:1935/mobile/$;

}

#creates our "mobile" lower-resolution HLS videostream from the ffmpeg-created
stream and tells where to put the HLS video manifest and video fragments

application mobile {

allow play all;

live on;

hls on;

hls_nested on;

hls_path /HLS/mobile;

hls_fragment 10s;

}

#allows you to play your recordings of your live streams using a URL like
"rtmp://meine-ip:1935/vod/filename.flv"

application vod {

play /video_recordings;

}

}
```

```
}

http {

include       mime.types;

default_type  application/octet-stream;

server {

listen 80;

server_name meine-ip;

#creates the http-location for our full-resolution (desktop) HLS stream -
"http://meine-ip/live/mein-stream-key/index.m3u8"

location /live {

types {

application/vnd.apple.mpegurl m3u8;

}

alias /HLS/live;

add_header Cache-Control no-cache;

}

#creates the http-location for our mobile-device HLS stream -
"http://meine-ip/mobile/mein-stream-key/index.m3u8"

location /mobile {

types {

application/vnd.apple.mpegurl m3u8;

}
```

```
alias /HLS/mobile;

add_header Cache-Control no-cache;

}

#allows us to see how stats on viewers on our Nginx site using a URL like:
"http://meine-ip/stats"

location /stats {

stub_status;

}

#allows us to host some webpages which can show our videos: "http://meine-ip/my-
page.html"

location / {

root   html;

index  index.html index.htm;

}

}

}
```

13. Das Program OBS (Open Broadcast Studio) starten und die Standard-Einstellungen belassen, bis auf den Eintrag “URL” und “Stream Key”

```
Encoder-x264

Variable bitrate (not CBR or Constant Bit Rate), Quality highest

Max bitrate-600kbps
```

```
Audio-Codec-AAC

Audio-Format-44.1khz

Audio-bitrate-64kbps

URL "rtmp://deine-ip:1935/live"

Stream Key-"mein-stream-key"

Resolution-640x480

FPS (frames per second)-30

CFR (Constant Frame Rate) - Yes

Keyframe interval-2 seconds (one keyframe every 2 seconds)

x264 Encoding Profile-baseline (may work with main depends on player used)

x264 CPU Present-veryfast
```

a) In OBS eine Videoquelle auswählen, z.B. Webcam
b) die Schaltfläche “Streaming starten” betätigen

Ergebnis: Die Beschriftung der Schaltfläche wechselt in “Streaming stoppen” und es wird in der unteren Statuszeile die Übertragungsrate angezeigt.

14. Nun folgt ein Test mit dem VLC-Media-Player. Den Menüpunkt des VLC, “Medien/Netzwerkstream öffnen” und die folgende Adresse eingeben:

```
http://meine-ip/live/mein-stream-key/index.m3u8
```

Wenn alles geklappt hat, zeigt der VLC-Player deinen Stream. Gratulation!

15. Stream in einer Website mit dem Media-Player “Video.js” einbinden.

a) Den video.js-Player herunter laden und im Verzeichnis /usr/local/nginx/html entpacken.

b) Folgende HTML-Seite mit dem Namen z.B. “meine-index.html” in dem o.g. Verzeichnis erstellen.

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 />
<title>Meine Live-Streaming-Server</title>
<link rel="stylesheet" type="text/css" href="videojs-matrix.css">
<link href="/video-js.css" rel="stylesheet">
<script src="/video.js"></script>
<script src="/videojs-contrib-hls.js"></script>
<script>
var player = videojs('example-video');
player.play();
</script>
</head>
<body>
<h1></h1>
<body>
<video poster="/logo.jpg" width="640px" height="267px" id="video" class="vjs-
matrix video-js  vjs-big-play-centered " controls autoplay preload="auto" data-
```

```
setup='{ "aspectRatio":"640:267"}'>

<source src="http://<meine-ip>/live/123/index.m3u8" type="application/x-
mpegURL">

<p class="vjs-no-js">To view this video please enable JavaScript, and consider
upgrading to a web browser that

<a href="http://videojs.com/html5-video-support/" target="_blank">supports HTML5
video</a></p>

</video>

</body>

</html>
```

16. Die erstellte Website im Browser aufrufen: **http://meine-ip/meine-index.html**

Die Website erscheint mit dem eingefügtem Live-Player.

Glückwunsch!

Du hast konzentriert und exakt gearbeitet und besitzt nun einen eigenen Live-Streaming-Server!

Weiter hilfereiche Seiten für das von mir beschriebene Setup:

https://obsproject.com/forum/resources/how-to-set-up-your-own-private-rtmp-server-using-nginx.50/
https://www.vultr.com/docs/setup-nginx-on-ubuntu-to-stream-live-hls-video
https://www.vultr.com/docs/setup-nginx-rtmp-on-ubuntu-14-04
https://www.selimatmaca.com/category/live-streaming/