

# Streaming [WIP]

How To für ein Streaming-Setup.

Dieses How To ist noch im aufbau.

- [Einführung](#)
- [OBS Basics](#)
- [FFmpeg Basics](#)
- [Streaming Konzepte](#)
  - [Einfaches Setup](#)
  - [Event: Talk Konzept](#)
  - [Team: Studio, Regie, Technik](#)
  - [ffmpeg](#)
- [Streaming Endpoints](#)
  - [Twitch](#)
  - [media.ccc.de](#)
  - [Google](#)
  - [Owncast](#)
  - [Nginx](#)
- [OBS: Troubleshooting](#)
- [FAQ](#)
- [Einkaufsliste](#)

# Einführung

## Einführung

In diesem How To sollen die wichtigsten Komponenten und der Aufbau eines Streaming Setups beschrieben werden. Wichtig ist die Größe und Komplexität zu beachten. Zu einem kann man ein kleines Setup benutzen, um nur den Bildschirm vom lokalen Computer zu einem Streamingdienstanbieter zu senden, bis hin eine Konferenzschaltung mit mehreren Teilnehmenden, sowie mit Einblendungen, Bauchbinden und Übergängen zwischen Kameras oder Szenen.

## Die Basics

Ein Stream verhält sich ähnlich wie eine Videokonferenz im **Big Blue Button** oder **Jitsi**. Während die Videokonferenz rudimentäre Werkzeuge zur Verfügung stellt, kann man bei einem Stream das ausgehende Video manipuliert werden. Da der Videofluss bearbeitet wird bevor dieser rausgeht, ist ein leistungsstärker Computer zu empfehlen. Zwar kann dies auch mit einem Consumer Notebook ohne dedizierter Grafikkarte erfolgen, aber dieses Notebook wird wegen der schwachen Grafikkarte und CPU viel kühlen müssen und je nach Anwendung, kann dies zu einem Leistungsverlust führen.

Als Beispiel soll eine Anwendung mit 3D Effekten (zum Beispiel ein Spiel), welches in FullHD läuft, auf Twitch gestreamt werden. Dies wird mit einem Gerät ohne stärkerer GPU und CPU zu Bildverlusten und öfteren Hängern des Videos oder Frame-Drops führen. Da nicht nur die Anwendung rechenintensive Aufgabe ausführt, sondern auch das Video codiert werden muss, kommt es zu einer doppelten Belastung der Hardware. Anders verhält es sich, wenn nur der Desktop mit einer Präsentation und einer Webcam gestreamt werden soll. Hierfür kann bereits ein schwacher Office-PC ausreichen.

Für einen Stream wird ein Computer benötigt (gegebenenfalls mit einer Webcam, wenn man sich selber filmen möchte), eines den aufgelisteten Betriebssystemen, eine Streaming-Software wie OBS und ein Ziel wohin gestreamt werden soll.

- **Computer:**
  - Dedizierte NVIDIA oder AMD Grafikkarte
    - Alternativ eine interne Grafikkarte (sofern nichts Rechenintensives gestreamt werden soll)
  - Eine Mittelklassen CPU
    - Intel i5 oder besser
    - AMD Ryzen 3 oder besser
  - evtl. Webcam
- **Betriebssystem:**

- Windows
- OS X
- Linux Derviate
- FreeBSD
- OpenBSD
- **Streaming Software:**
  - Grafische Tools
    - OBS
  - Command Line Interface
    - ffmpeg
- **Streaming Dienst:**
  - Twitch
  - YouTube
  - Owncast (Selfhosted)
  - Lokale Aufnahme
- Internet
  - Breitband Internet mit durchgehende und stabile 7000 Kbit/s im Upload.

# OBS Basics

Open Broadcast Software (kurz OBS) ist ein Open-Source-Programm, welches eine einfache und freie Möglichkeit gibt, vom lokalen Computer zu unterschiedlichen Anbietern zu streamen. OBS kann für verschiedene Betriebssysteme heruntergeladen und erweitert werden. Die Beliebtheit ist der einfachen Handhabung, sowie der vielseitigen Einsatzmöglichkeit zu verdanken, weshalb die Gaming- und Twitch Community großflächig dieses Tool einsetzt.

---

## Download und Installation

OBS steht für verschiedene Betriebssystemen zur Verfügung und kann von der offiziellen Seite, sowie aus den Paketquellen des jeweiligen Paketmanagers heruntergeladen werden.

### Windows

Installer herunterladen und ausführen:

**OBS Studio Website:** <https://obsproject.com/download>

**GitHub Releases:** <https://github.com/obsproject/obs-studio/releases>

Installation via den Microsoft Store:

**Microsoft Store:** [\[click\]](#)

Installation via Steam:

**Steam:** [\[click\]](#)

Aus dem Quellcode kompilieren

**Dokumentation:** <https://obsproject.com/wiki/build-instructions-for-windows>

### macOS

#### details

Disk Image (.dmg) herunterladen und die OBS.app in den Programme Ordner schieben

**OBS Studio Website:** <https://obsproject.com/download>

**GitHub Releases:** <https://github.com/obsproject/obs-studio/releases>

Installation via Steam:

**Steam:**[\[click\]](#)

Aus dem Quellcode kompilieren

**Dokumentation:** <https://obsproject.com/wiki/build-instructions-for-mac>

## GNU/Linux

### details

Die Installation für Linux kann auf verschiedenen Wegen erfolgen.

**Flatpak:** Installation via [Flathub](#)

### Ubuntu:

```
sudo add-apt-repository ppa:obsproject/obs-studio
sudo apt update
sudo apt install obs-studio
```

## Arch Linux und Manjaro:

### details

```
sudo pacman -S obs-studio
```

Im [AUR](#) befinden sich verschiedene PKBUILDS um OBS mit unterschiedlichen Modulen zu bauen:

- obs-studio-tytan652
  - OBS mit VST 2 Plugin, RIST Protokoll und AJA Device Support.
  - Integrierten Browser Modul
  - Kleine Anpassungen um das Leben leichter zu machen
- obs-studio-browser
  - OBS mit Integrierten Browser Modul und VST

- obs-studio-rc
  - Release Candidate und Beta Version

## Debian:

### details

```
sudo apt update
sudo apt install ffmpeg obs-studio
```

Aus dem Quellcode kompilieren

**Dokumentation:** <https://obsproject.com/wiki/build-instructions-for-linux>

## Fedora:

### details

```
# RPM Fusion hinzufügen, falls noch nicht geschen
sudo dnf install https://download1.rpmfusion.org/free/fedora/rpmfusion-free-release-$(rpm -E
%fedora).noarch.rpm https://download1.rpmfusion.org/nonfree/fedora/rpmfusion-nonfree-release-$(rpm
-E %fedora).noarch.rpm
# Installation OBS
sudo dnf install obs-studio
```

## Gentoo:

### details

```
sudo emerge media-video/obs-studio
```

## NixOS:

### details

```
nix-env -i obs-studio
```

## OpenMandriva Lx4

### details

```
dnf install obs-studio
```

## Solus

### details

```
eopkg install obs-studio
```

## Void

### details

```
sudo xbps-install obs
```

## FreeBSD

### details

Die Installation für FreeBSD erfolgt via `pkg`.

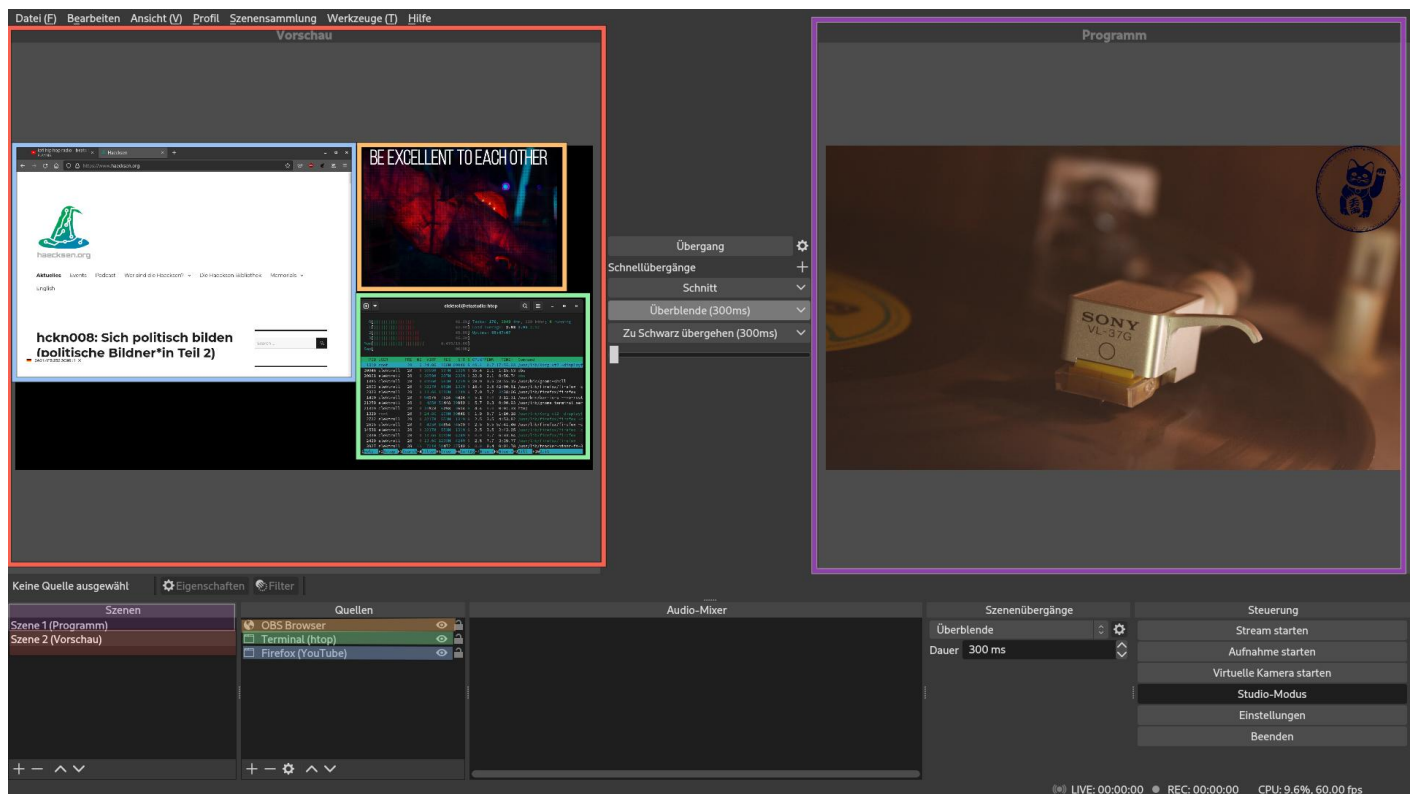
```
pkg install obs-studio
```

## OpenBSD

### details

Für OpenBSD besteht ein Work In Progress Port auf GitHub unter:  
<https://github.com/jasperla/openbsd-wip>

# Einführung



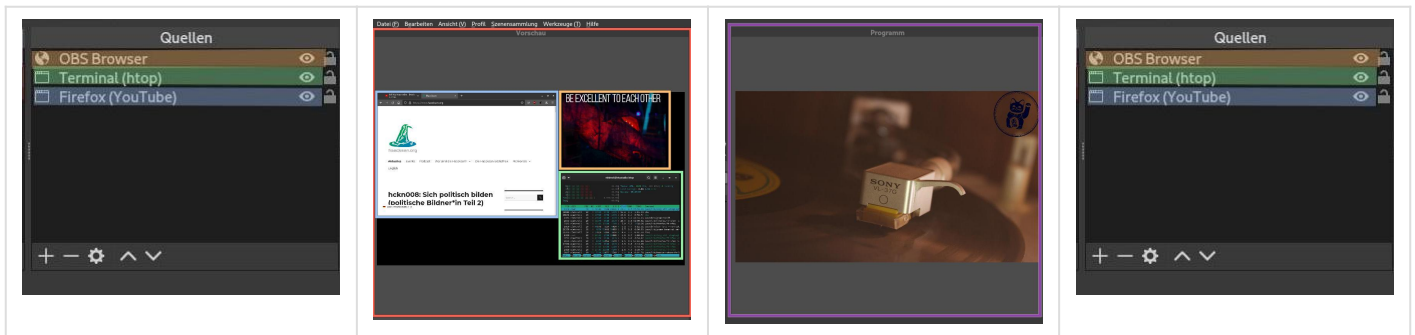
*Screenshot OBS Version 27.1.3 unter Arch Linux, kompiliert mit Browser Modul.*

In dieser Dokumentation kann das Äußere/Aussehen von OBS je nach Betriebssystem und Version leicht variieren. In der Standardansicht von OBS, wird nur die aktuell laufende Szene angezeigt. Es ist zu empfehlen, unten rechts auf den **Studio-Modus** zu wechseln, in dem man die gleichnamige Schaltfläche dafür anklickt.

Im Studio-Modus bietet OBS eine getrennte Ansicht, zwischen dem was aktuell gesendet wird (rechts) und dem was in Vorbereitung (links) ist.

Dem Screenshot kann man verschiedene farbliche Markierungen entnehmen. Das linke Fenster in **Orange** entspricht der Szene mit dem Namen "**Szene 2 (Vorschau)**". Das rechte Fenster in **Lila** entspricht somit der Szene "**Szene 1 (Programm)**". Die Benennung der Szenen kann frei gesetzt werden. Das ist möglich über einen Rechtsklick auf die Szene und dann die Schaltfläche "Umbenennen"/"Rename".



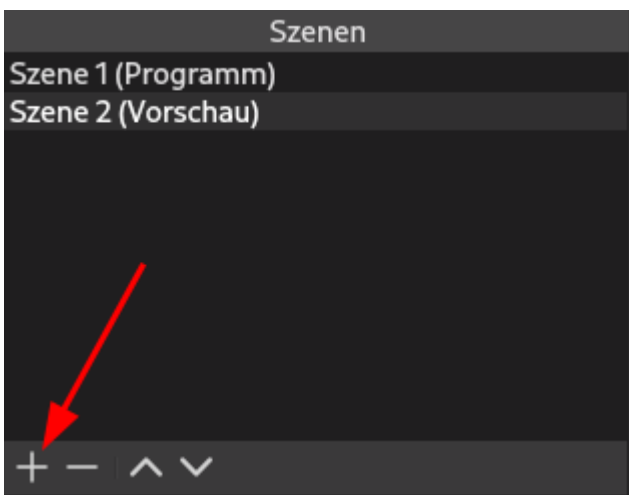


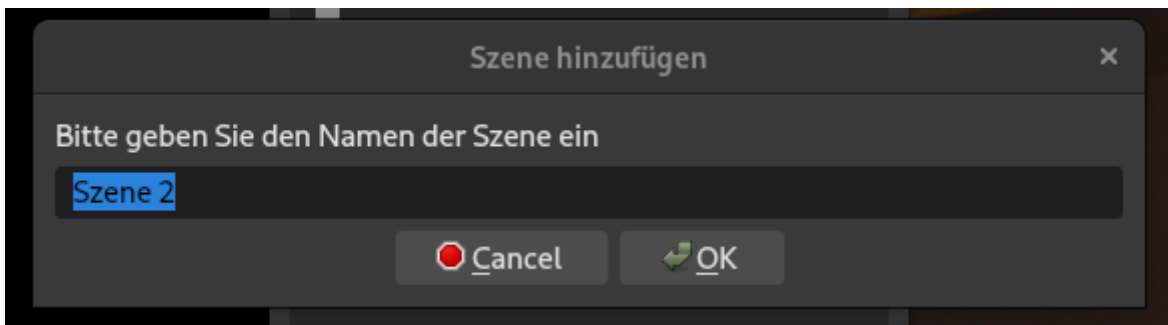
In einer Szene können mehrere Quellen hinterlegt werden. Im linken Vorschaufenster (Orange) sind drei unterschiedliche Quellen zusehen, welche gerade vom Bildschirm abgegriffen werden und frei positioniert wurden. Diese Quellen sind als Ebenen übereinander gelegt. Dies kann man sich wie Klarsichtfolien vorstellen, welche jeweils ein unterschiedliches Motiv zeigen und übereinander gelegt sind. Zu einem haben wir ganz oben das gelbliche Fenster namens OBS Browser. Hierbei handelt es sich um einen integrierten Browser im OBS, welcher eine URL aufruft und die Seite rendert. Darunter in Grün wird ein Terminal mit dem Taskmanager htop aufgenommen und zum Schluss wurde in diesem Beispiel noch ein Firefox Fenster hinzugefügt. Die Szenen können über einen Klick gewechselt werden, sodass Szene 2 Live geht.

## Neue Szene hinzufügen

Um eine eigene neue **Szene** hinzuzufügen, muss lediglich im Szenen Fenster auf das **Plus** geklickt und der **Szene** ein Name gegeben werden. Hierbei wird eine neue und leere Szene erstellt, welche wir befüllen können. Zum Beispiel mit dem Abgreifen eines Fensters. Für dieses Beispiel benennen wir die neu erstellte Szene "**Szene 3**" und fügen im benachbarten Fenster eine neue **Quelle hinzu**.

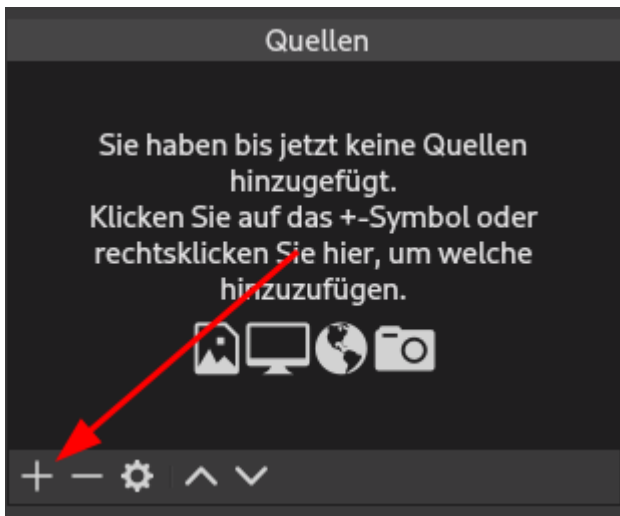
Beachtet das sich die Quellen im Namen unterscheiden können, je nach dem welches Betriebssystem benutzt wird und welcher grafischer Server im Einsatz ist. In diesem Beispiel wird unter Arch Linux mit X11 gearbeitet, weshalb die Fensteraufnahme den Beitel Xcomposite trägt.

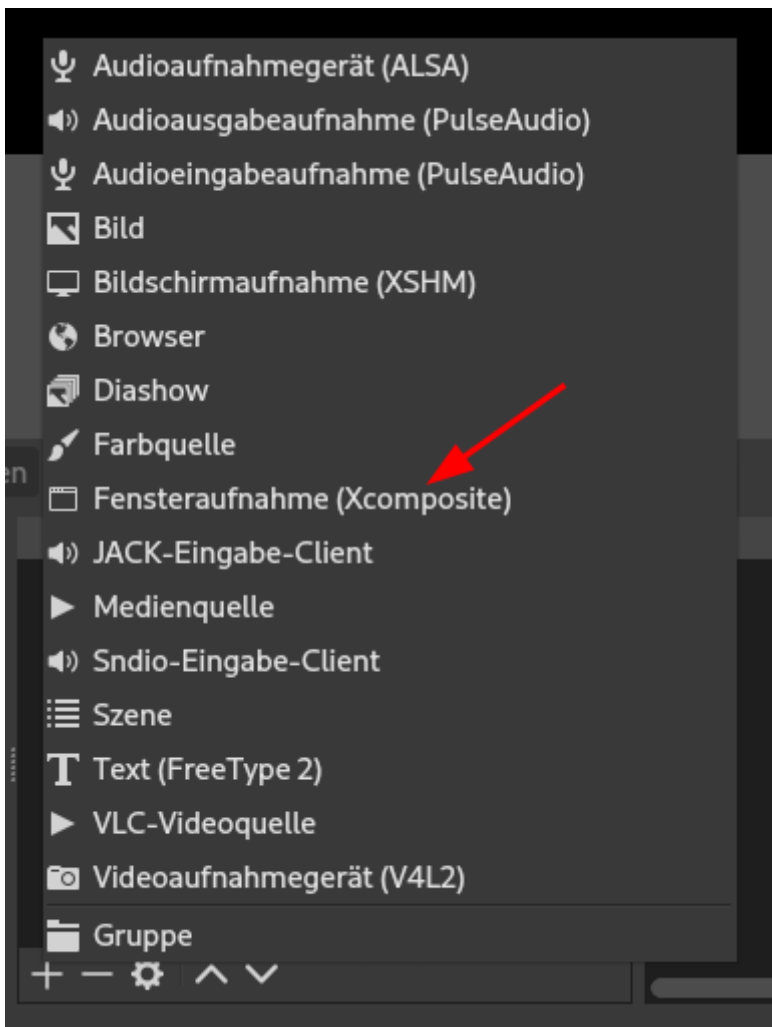




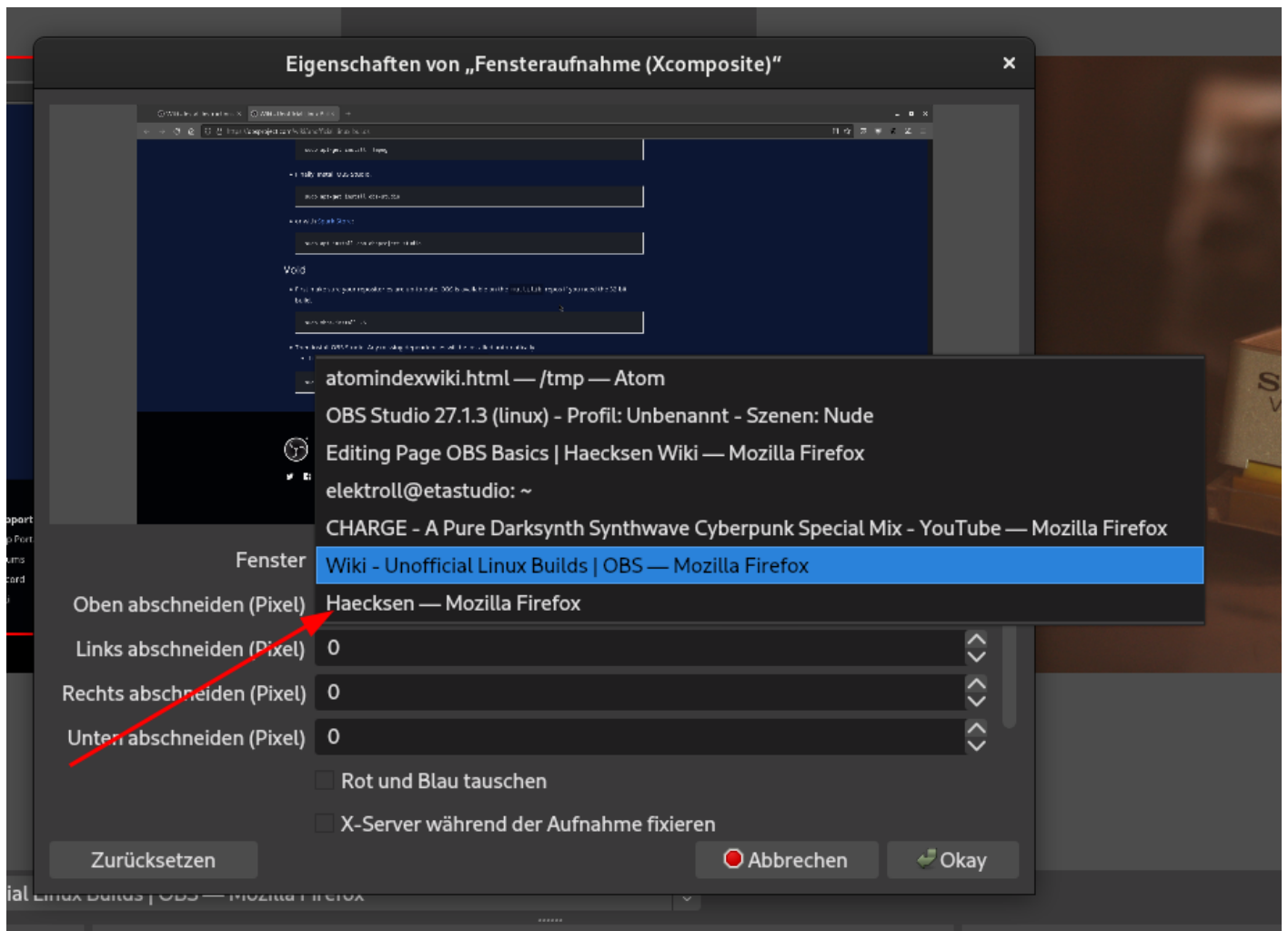
## Bild Quellen hinzufügen

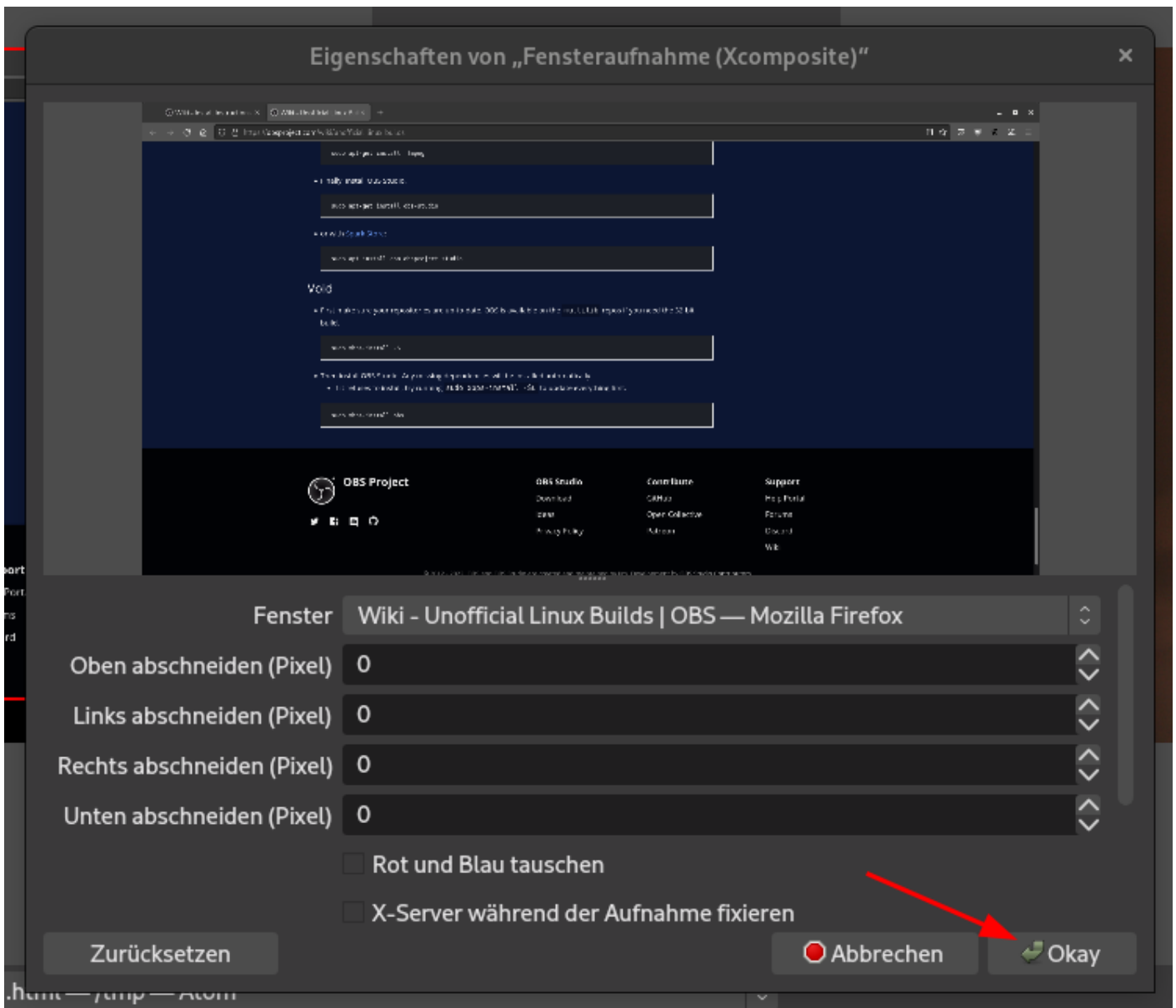
Da wir nun eine neue **Szene** haben, klicken wir ebenfalls auf das **Plus** Symbol bei den **Quellen** und fügen eine neue **Quelle** hinzu. Diese nennt sich **Fensteraufnahme**.



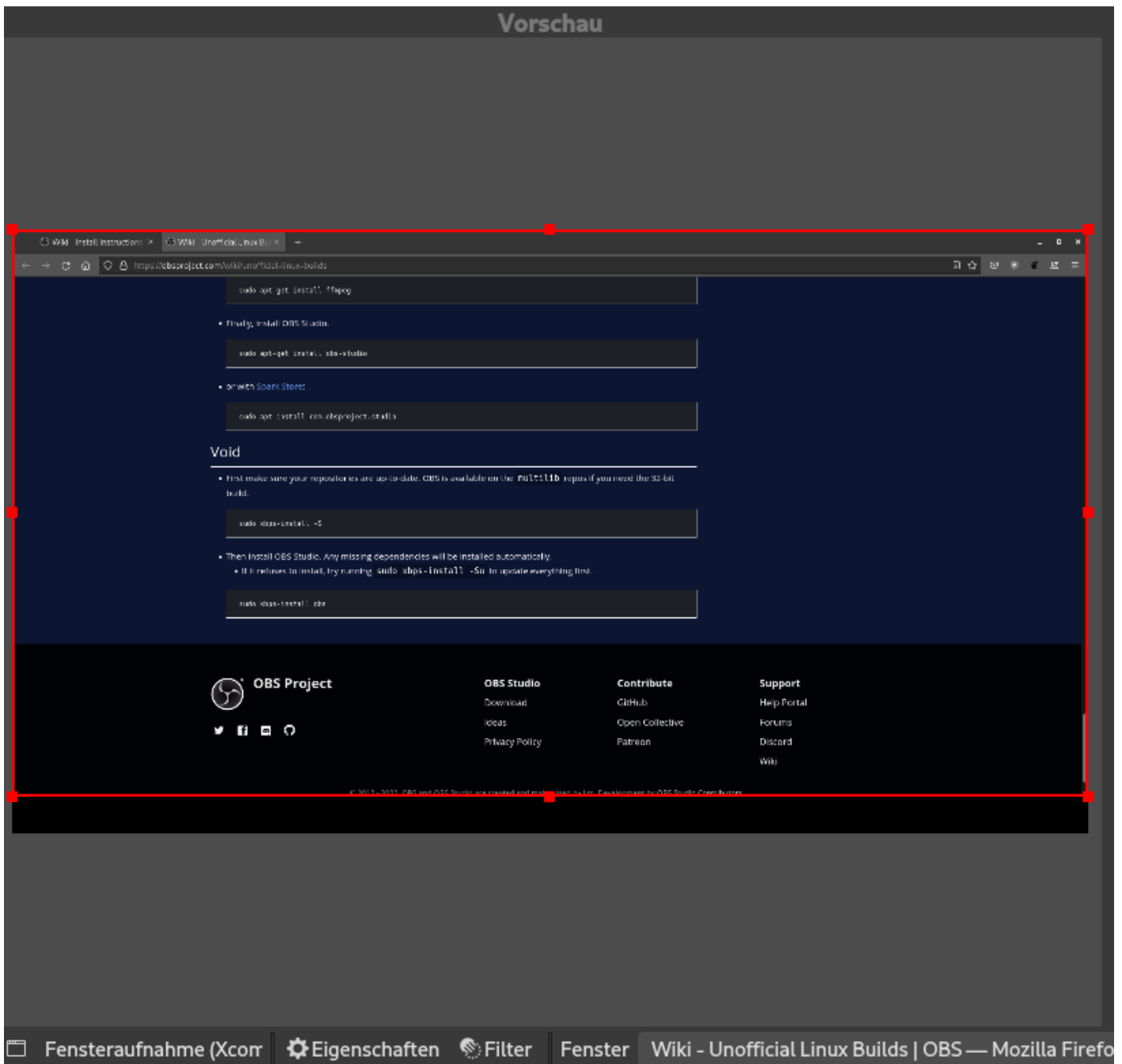


In dem darauf folgenden Fenster können wir das Fenster definieren, welches aufgezeichnet werden soll. Hierzu muss auf die Schaltfläche Fenster geklickt werden und das jeweilige Fenster ausgewählt werden. Daraufhin können noch vereinzelt Optionen angepasst werden. Bestätigt wird dies mit Okay.

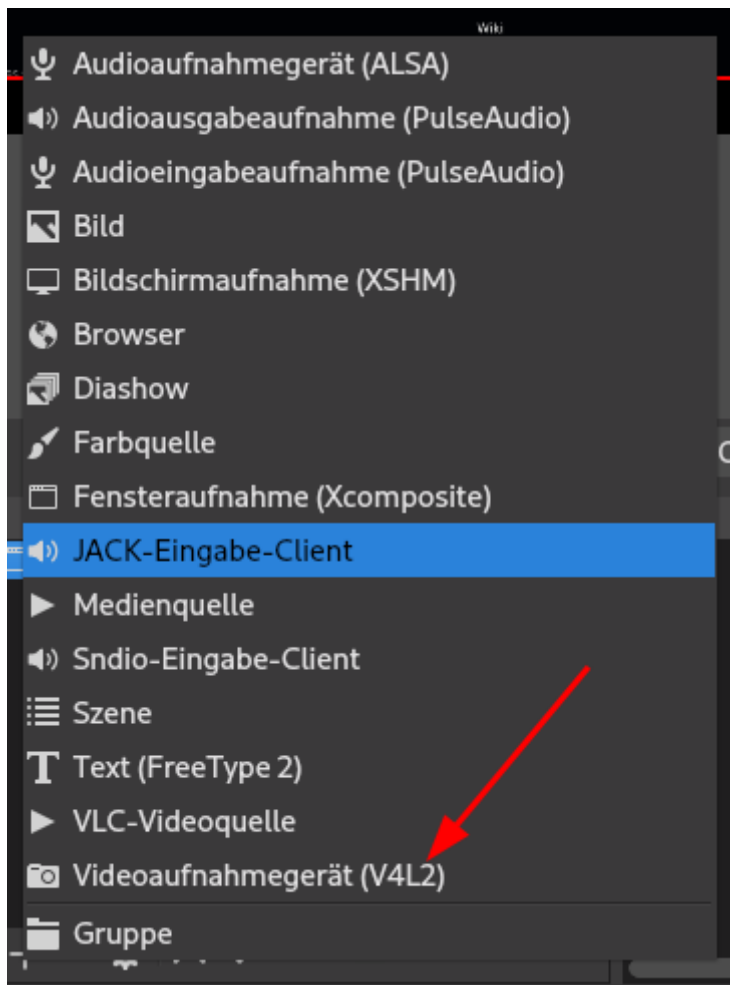


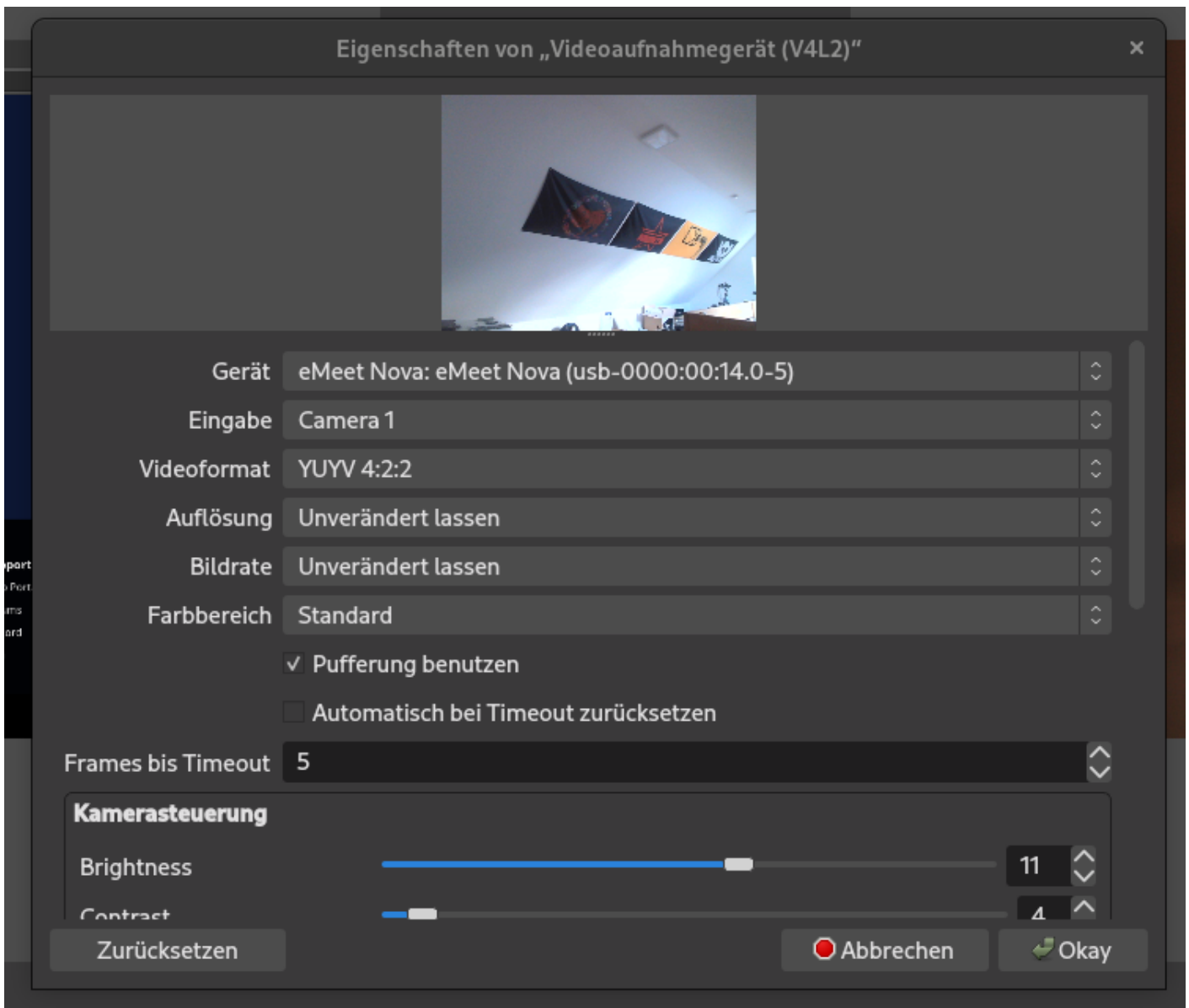


Im Vorschaufenster sehen wir nun den Firefox Browser, welchen ich für diese Einführung ausgewählt habe. Die rote Markierung um das Fenster bietet mit die Möglichkeit das Fenster zu skalieren und somit zu verkleinern oder zu positionieren.



Zu der bestehenden Quelle fügen wir nun unsere Webcam als eine eigene Quelle hinzu. Dafür klicken wir erneut auf das Plussymbol und wählen **Videoaufnahmegerät**. (Das V4L2 steht für Video For Linux 2 und ist eine Linux spezifische Bezeichnung). In dem folgenden Fenster können noch Einstellungen durchgeführt und mit einem Klick auf "okay" bestätigt werden.

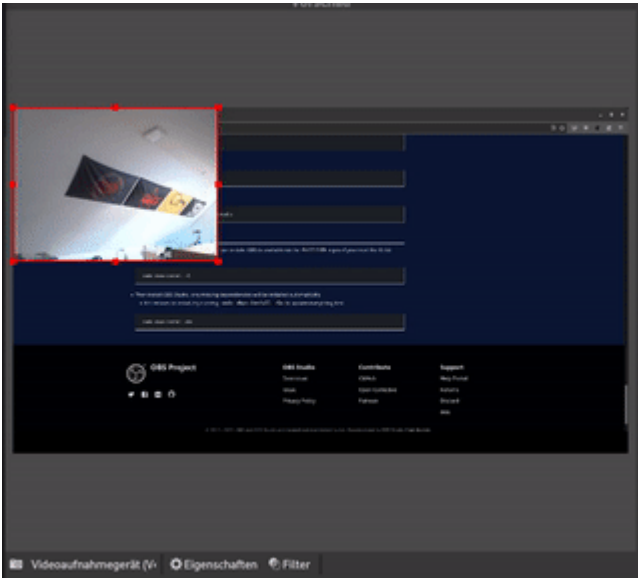




## Bild Quellen anpassen

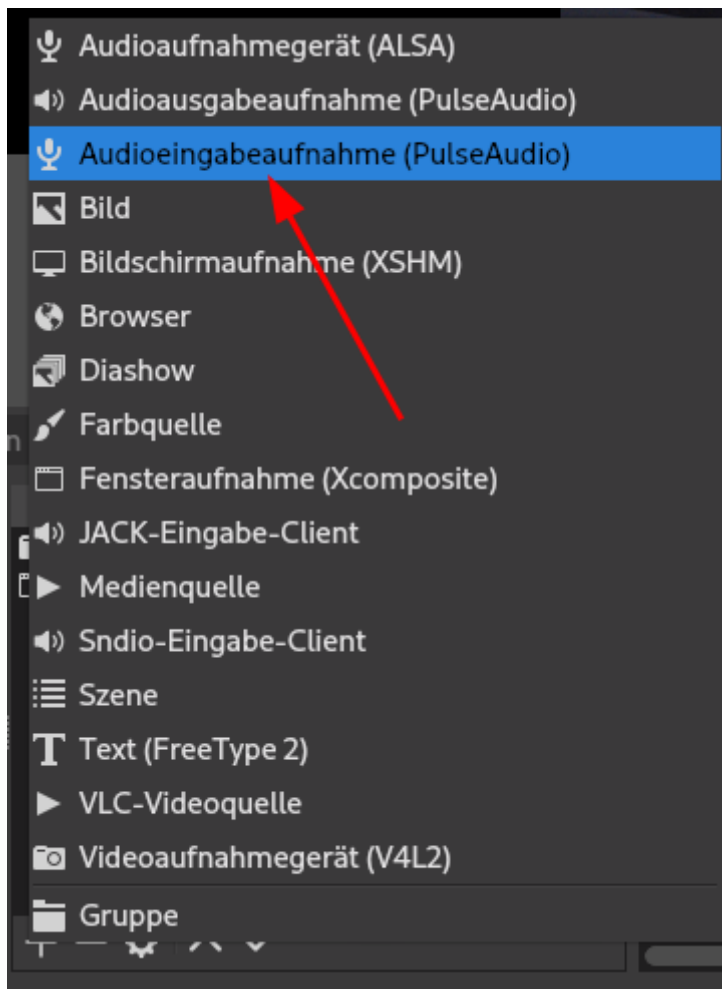
Es ist nun möglich beide Quellen im Vorschaufenster, via Drag-and-drop zu verschieben, zu skalieren und zuzuschneiden, indem wir die Objekte einfach anklicken und schieben. Für die Skalierung der Fenstergröße müssen die Ecken ausgewählt werden. **Wenn man dabei die ALT Taste festhält, wird das Fenster zugeschnitten.**

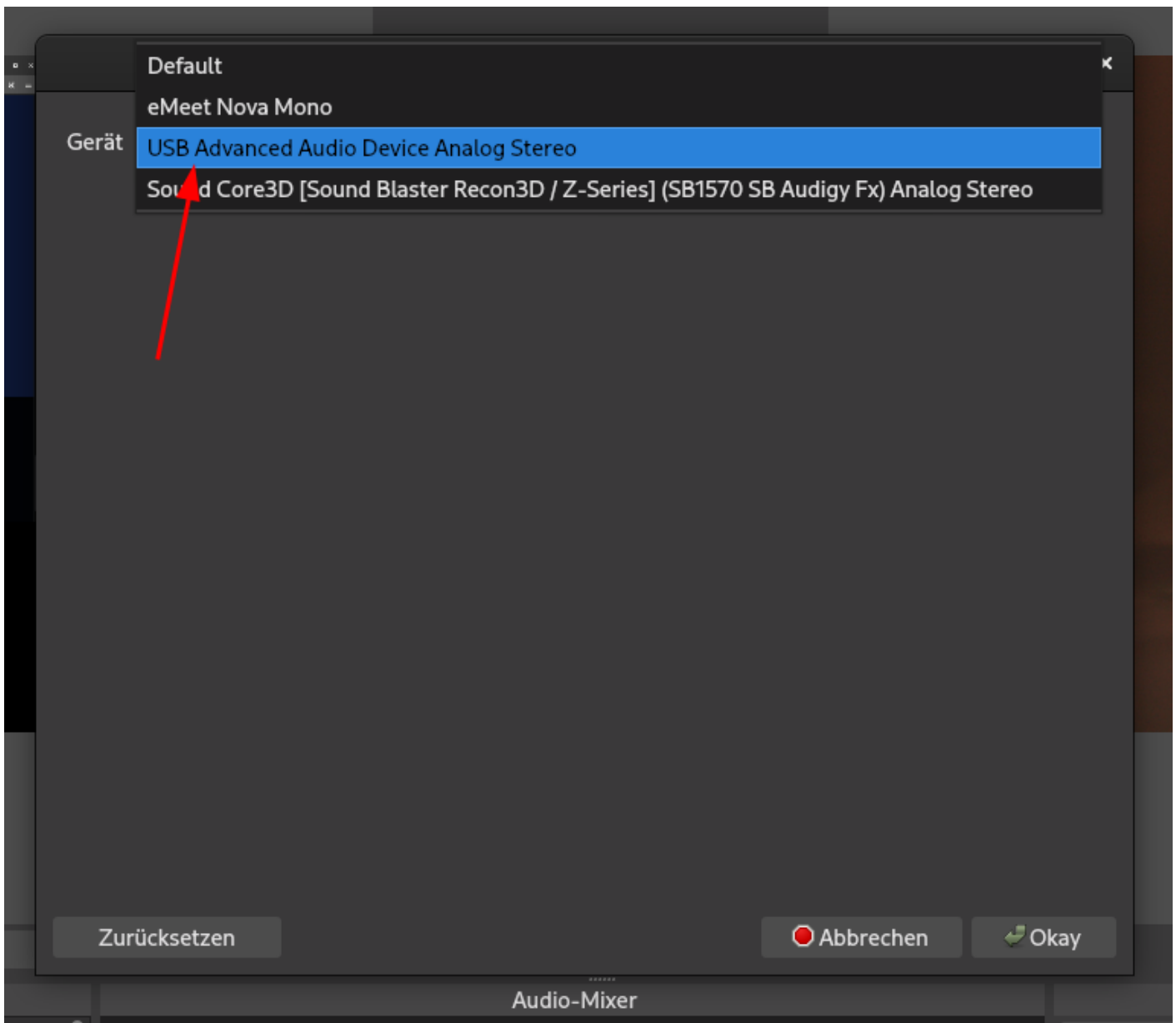




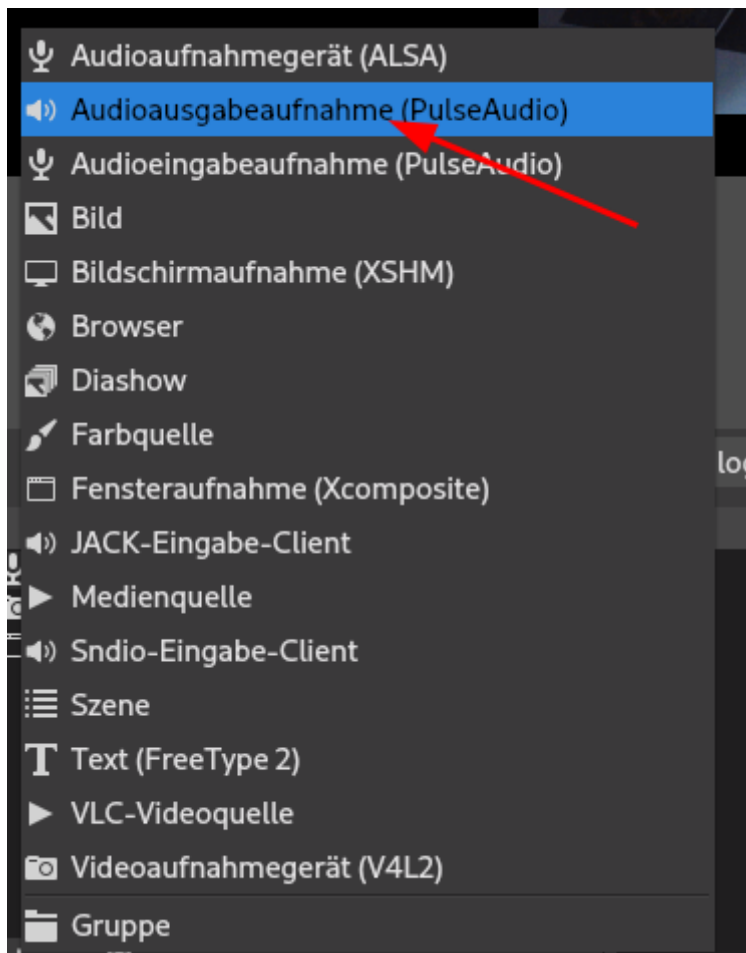
## Audio Quellen

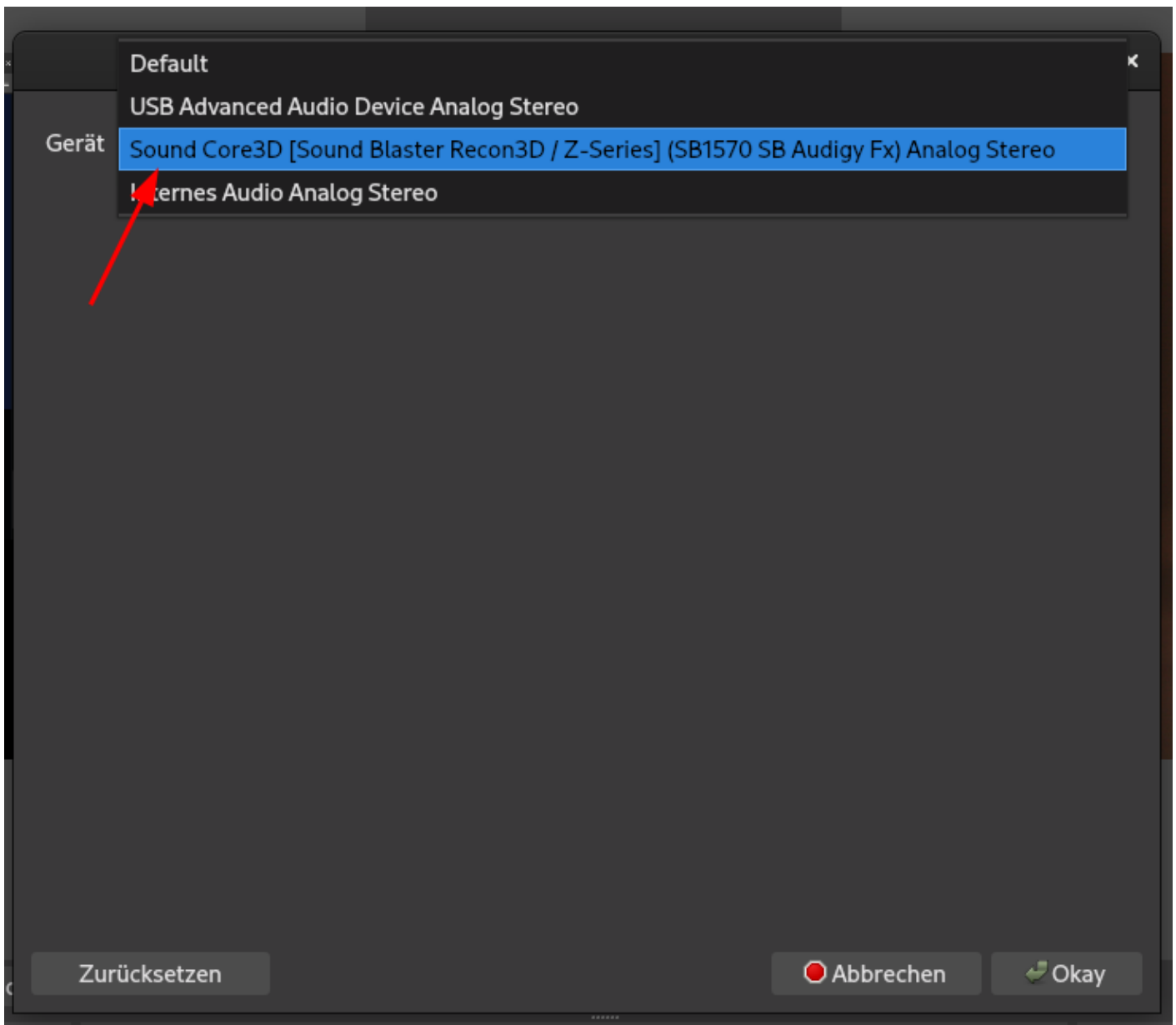
Wir haben nun die Bild-Quellen, welche wir aufnehmen wollen. Jetzt fehlen nur noch die Audio-Quellen wie **Desktop Audio** und ein **Mikrofon**. Diese werden wie die Bild-Quellen innerhalb der Szene als **Quelle** hinzugefügt. Um ein lokal angeschlossenes **Mikrofon** aufzunehmen, wählt man die **Audioeingabeaufnahme** (das PulseAudio steht für ein Linux spezifischen Audio-Server). In dem darauf folgenden Fenster kann das Mikrofon ausgewählt werden, welches als Audioeingabegerät genutzt werden soll. Bestätigt mit Okay.



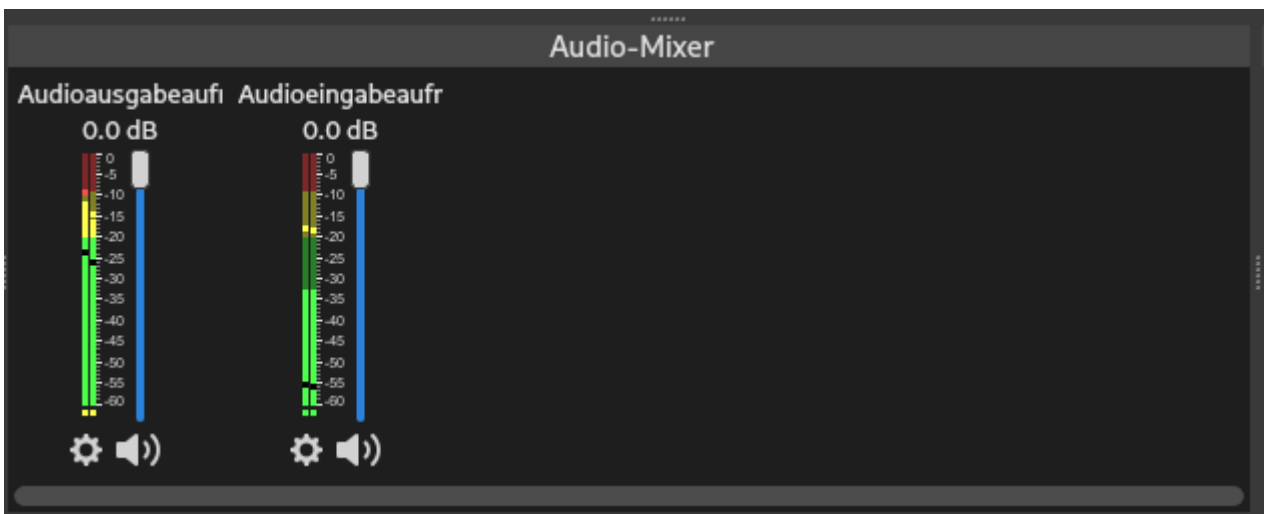
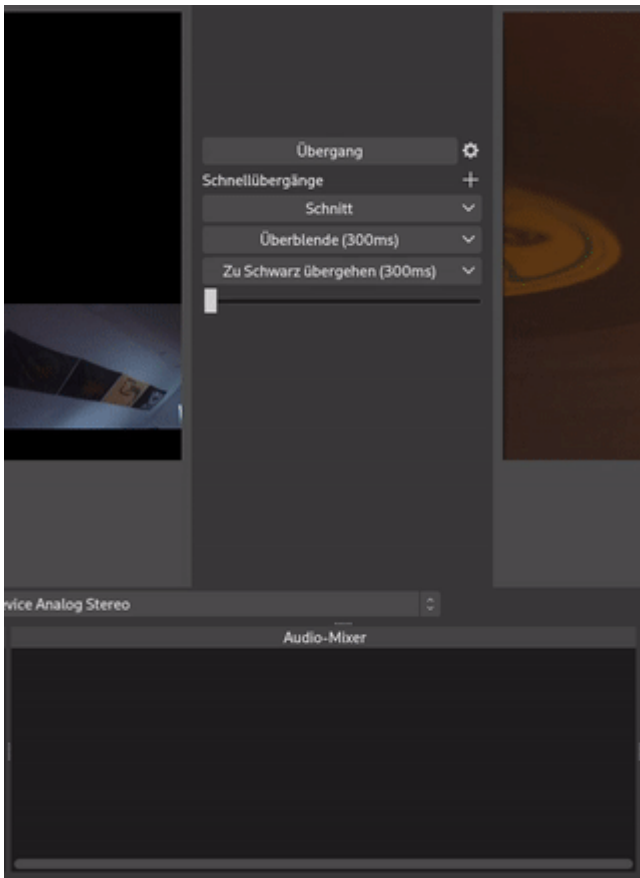


Wir wollen noch jenes Audio mitsenden, welches auf dem PC abgespielt wird. Sei es ein Spiel, Hintergrundmusik von einem Musikplayer oder ein Soundboard. Hierzu gehen wir genau so vor, wie oben beschrieben und wählen anstelle der Audioeingabeaufnahme nun **Audioausgabeaufnahme**. Wählt die Soundkarte aus, von welcher ihr das Audio abgreifen wollt. Die Gerätebezeichnungen werden mit sehr hoher Wahrscheinlichkeit andere sein, da in diesem Fall eine interne Soundkarte, eine zusätzliche Soundkarte und das Mikrofon als Soundkarte zur Auswahl stehen. Solltet ihr euch nicht sicher sein, ist die Default-Soundkarte meistens die richtige Wahl.





Neben der Liste an Quellen befindet sich ein Fenster mit den Audiogeräten, welche zu Beginn nicht angezeigt werden. OBS listet nur die Audio-Quellen auf, welche Live sind. Es gibt einen kleinen Trick, um die Anzeige der Audiogeräte im Mixer zu erzwingen. Zwischen dem Vorschaufenster und der Live Ansicht befindet sich ein kleiner Regler, welcher minimal nach rechts geschoben werden kann. Dieser Regler dient für den manuellen Übergang zwischen der Vorschau und dem, was gestreamt wird. Hierzu später mehr.



Im Audio-Mixer Fenster kann die Lautstärke des Audios angepasst werden, sollte zum Beispiel das Desktop-Audio das Mikrofon übertönen. Ein Audio Setup kann ziemlich komplex werden. Zum Beispiel möchte man sein Desktop Audio aufnehmen, jedoch nicht, was in einer Audiokonferenz mit mehreren teilnehmenden Personen besprochen wird. Hierzu werden virtuelle Soundkarten, Senken und des gleichen benötigt und ist von Betriebssystem zu Betriebssystem unterschiedlich zu konfigurieren.

## Stream Einstellungen

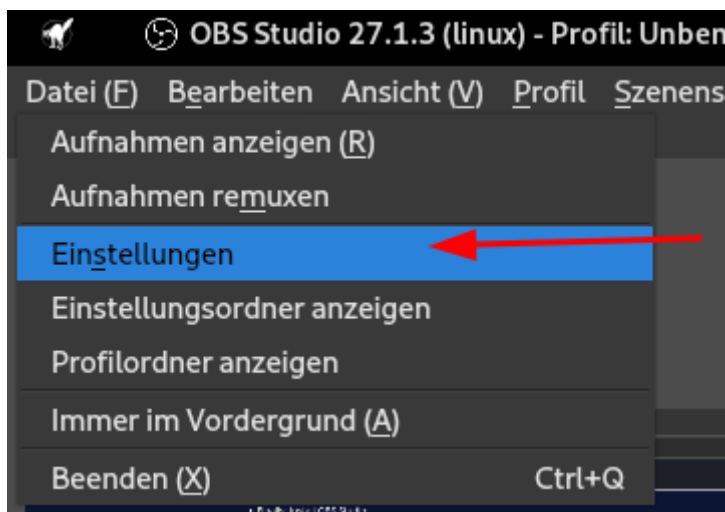
Da wir nun die **Bild-Quellen** und die **Audio-Quellen** definiert haben, muss nun das Ziel für den Stream definiert werden. Es gibt bereits voreingestellte Optionen für **Twitch**, **YouTube**, **Facebook**

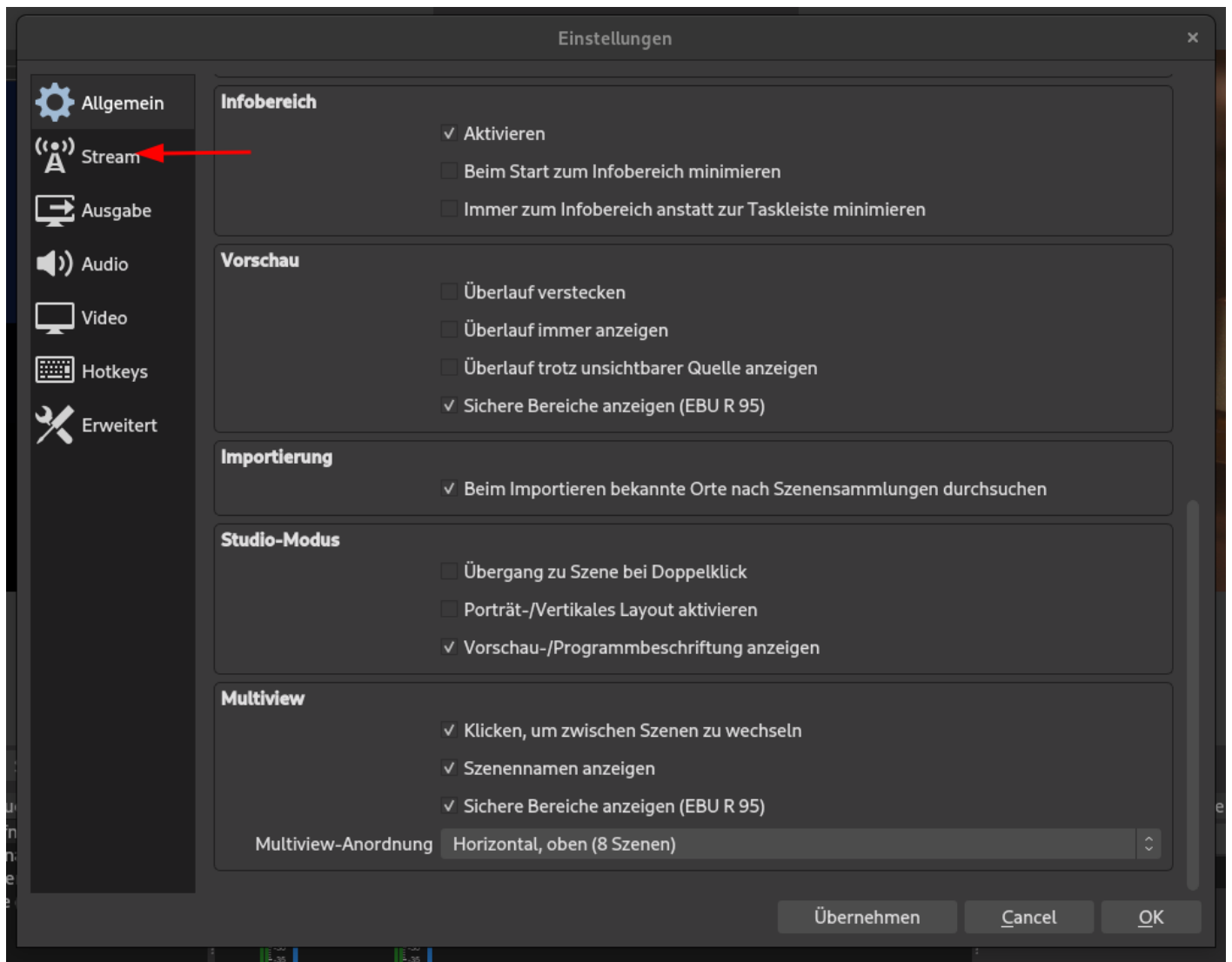
, **Twitter**, **Restream.io** und viele andere Anbieter. Es lassen sich aber auch nicht mit aufgelistete Dienste wie **media.ccc.de** einstellen. Die Konfiguration hierzu wird dann vom VOC mitgeteilt. Für dieses How To wählen wir Twitch.

Es wird ein Twitch Account benötigt

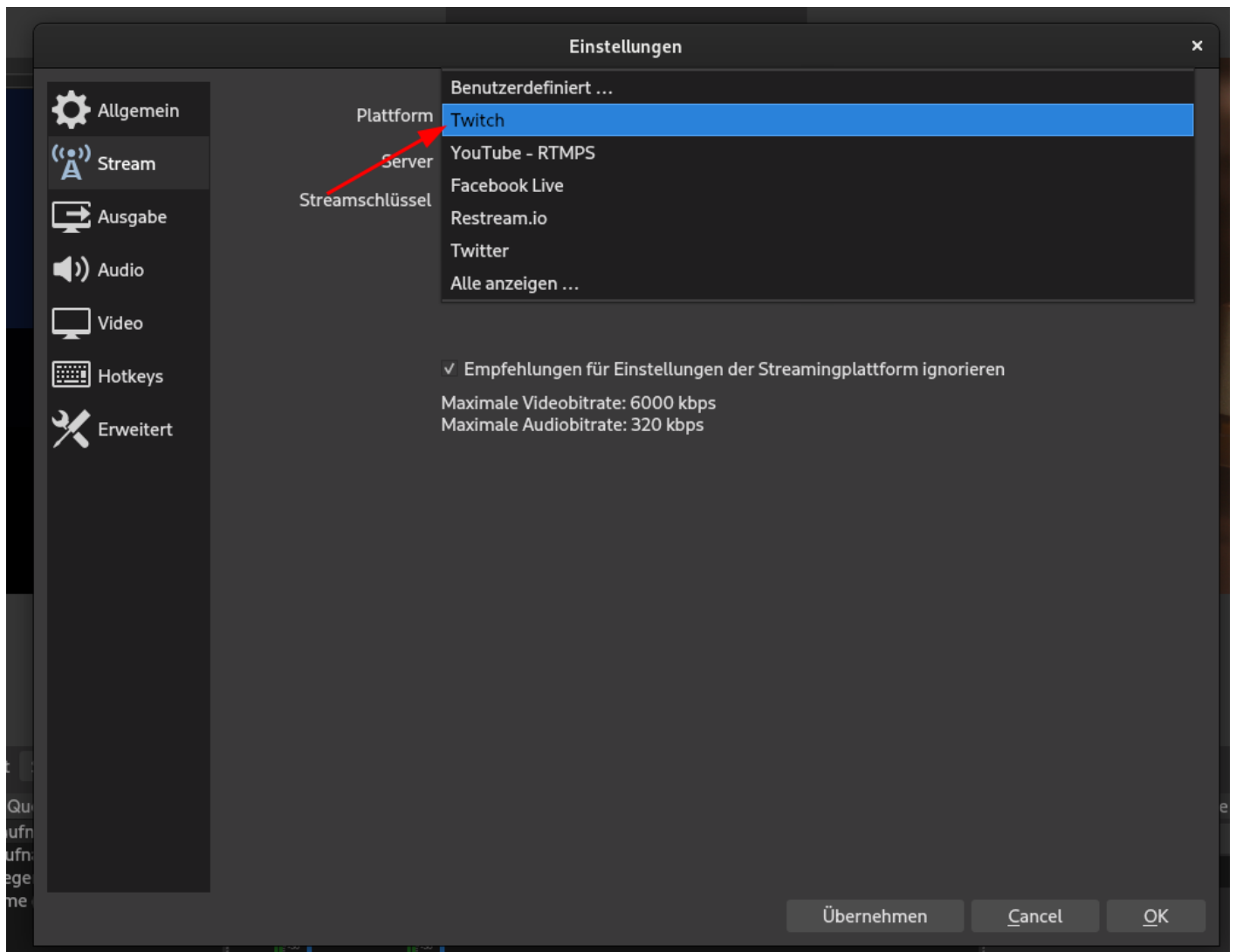
Die Anmeldung von OBS zu Twitch erfolgt nicht über die Zugangsdaten Usernamen/Email und Password, sondern über einen Schlüssel, welchen wir aus der Profilseite auf twitch.tv entnehmen können.

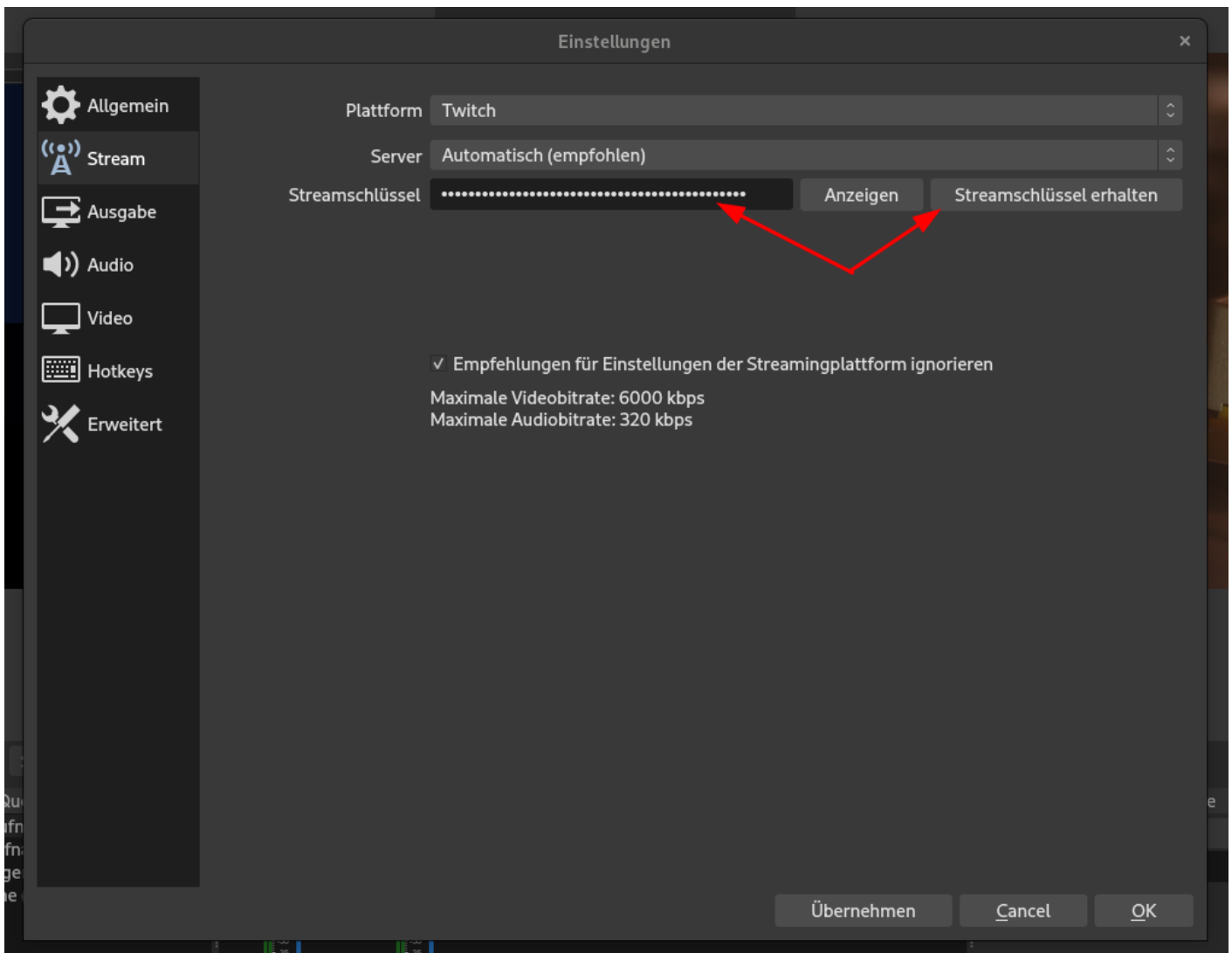
Twitch als Zielserver wird in den Einstellungen definiert. Geht hierbei über **Datei/Einstellungen** und klickt im neuen Fenster auf **Stream**. Über die Auswahl Plattform kann nun Twitch ausgewählt werden. Wenn ihr ein Browser offen habt und dabei in Twitch angemeldet seid, könnt ihr in OBS "**Streamschlüssel erhalten**" klicken und gelangt somit auf die korrekte Seite, um den Schlüssel zu kopieren, welcher in die Eingabemaske eingetragen werden muss. Damit weiß nun OBS wohin die Aufnahme gesendet werden soll, sobald man Live geht.











## Video Einstellungen

Neben der Möglichkeit zu einem Anbieter zu streamen, besteht auch die Möglichkeit den Videostream lokal zu speichern. Dies ist ratsam, wenn man eine Sicherheitskopie des Rohmaterials haben will oder wenn ein Stream nicht gewünscht ist, sondern nur eine Aufnahme stattfinden soll. Dabei wird die Ausgabe, welche zum Streaminganbieter geschickt wird, lokal als Datei hinterlegt. Eine weitere Option ist, wenn der Anbieter wie Twitch eine maximale Videobitrate von 6000 kbps unterstützt, aber das lokal gesicherte Videomaterial höherwertiger sein soll. Somit kann via Twitch ein Stream mit 6000 kbps gesendet werden, während lokal das Videomaterial eine bessere Qualität hat. Die Einstellungen für den Stream und die Aufnahme wird in den Einstellungen unter Ausgabe definiert.

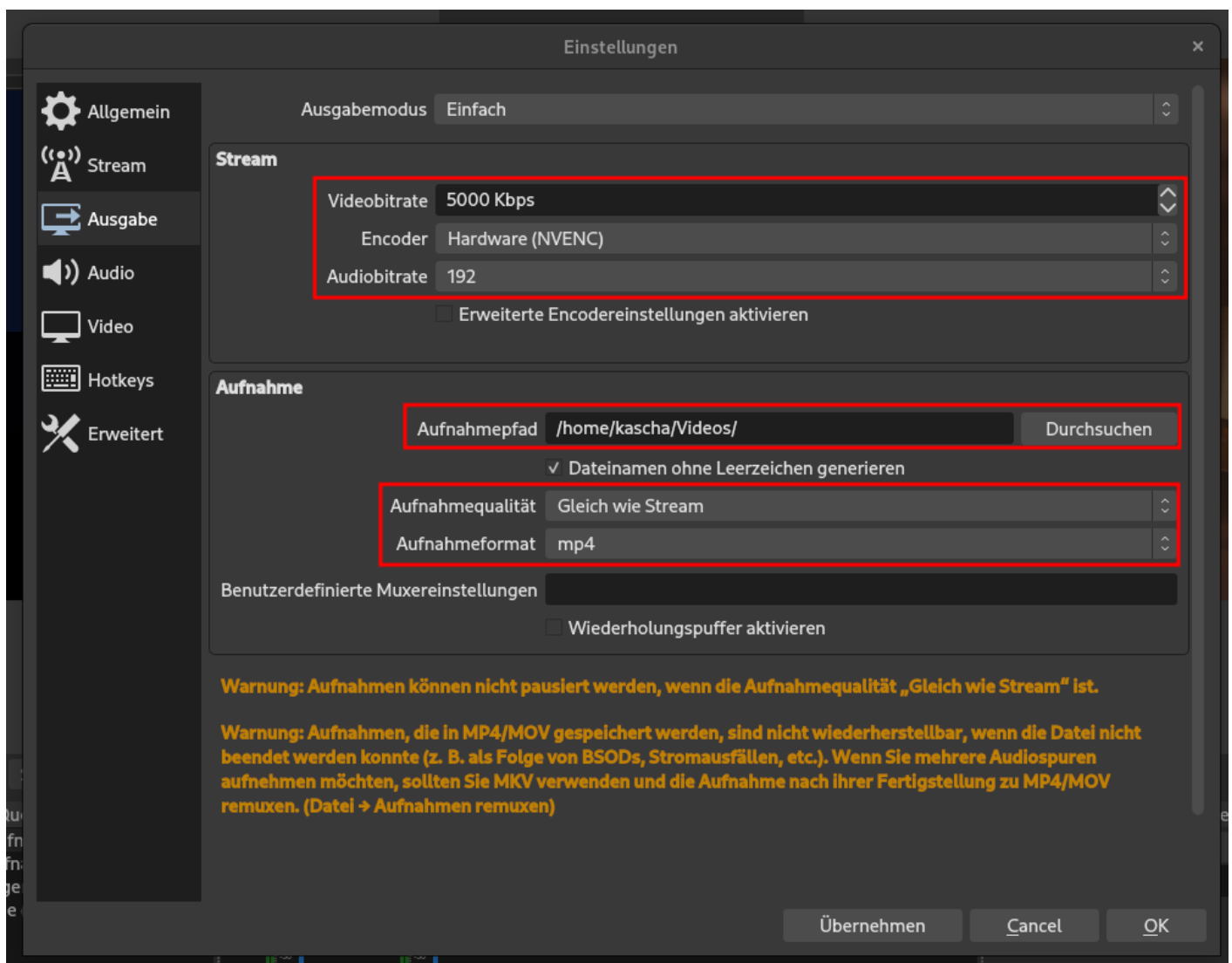
## Stream

Unter dem Bereich Stream im Reiter Ausgaben, können verschiedene Einstellungen durchgeführt werden. Zum Beispiel welcher Encoder benutzt werden soll. Die Videobitrate für den Stream ist unter anderem von eurer Internetanbindung abhängig. Bei einer geringen upload Geschwindigkeit, muss das Video bevor es gesendet wird herunter gestampft werden. Dies bedeutet, dass es zu Bildartefakten kommt oder der Stream matschig wird. Es ist ratsam immer den Stream über eine

stabile Breitbandanbindung durchzuführen.

Für einen Stream mit einer Ausgabe von 1920x1080, einer Bitrate von 5000 Kbps, Audiobitrate von 192 und eventuell noch Gespräche via Mumble, Discord, Big Blue Button usw. sollten 7500 Kbps im Upload reichen. Als Faustregel für die maximale Bitrate kann man  $\text{Upload}/1,5 = \text{Bitrate} + \text{Audiobitrate}$  nehmen.

Der Encoder welcher ausgewählt werden kann, ist abhängig von der Grafikkarte des Computers. Auf dem Screenshot kann man einen Hardware Encoder von Nvidia (NVENC) vernehmen. Dies kann bei euch abweichen, wenn ihr zum Beispiel eine Grafikkarte von AMD habt oder keine dedizierte Grafikkarte. Das Videoencoding kann auch Softwareseitig erfolgen, damit wird jedoch die CPU belastet.

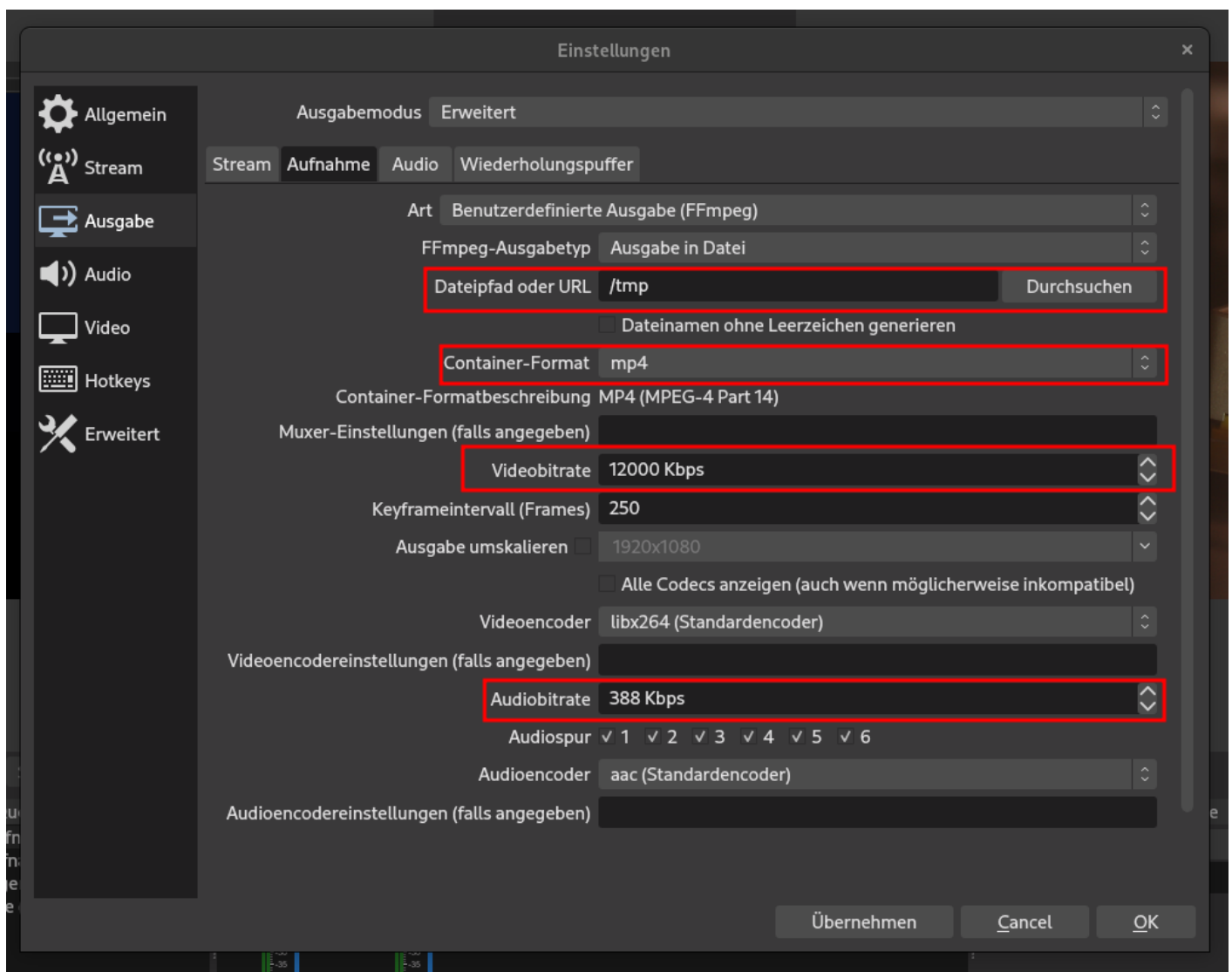


## Aufnahme

Dem Screenshot oben, können die Aufnahme Einstellungen entnommen werden. Zu einem wohin die Videodateien gespeichert werden sollen, aber auch in welcher Qualität die Dateien gespeichert werden sollen. In der vereinfachten Ansicht, bietet OBS vorgefertigte Profile, welche von der Bezeichnung her selbsterklärend sind. Es besteht auch die Möglichkeit die Aufnahme frei zu konfigurieren, in dem man auf die Erweiterte Ansicht umstellt. Somit lassen sich auch der lokalen

Aufnahme verschiedene Tonspuren zuweisen. Ich rate für den Anfang davon ab, die Erweiterte Ansicht zu benutzen, da man bei OBS mit den einfachen Einstellungen auf der sichereren Seite ist.

Wie dem Screenshot oben bereits entnommen werden kann, gibt es eine Warnung bei der Nutzung von MP4 als Videoformat. Die Technische Erklärung für die Warnung ist sehr simpel. Eine MP4 Datei kann im Gegensatz zu anderen Videoformaten keine Fehler beinhalten (zum Beispiel, fehlen plötzlich einige Sekunden in der Mitte des Videos). Sobald das encodieren des Videos nicht ordentlich beendet wurde, ist die gesamte Datei unbrauchbar. Ein MKV Container hingegen kann in mitte der Aufnahme Fehler haben oder kurzweilig die Aufnahme stoppen. Der Nachteil einer MKV ist, dass dieses Format nicht überall unterstützt wird. Der Einfachheit halber und der besseren Unterstützung hat sich MP4 durchgesetzt. Somit ist es ratsam MKV-Datei aufzunehmen und nach der Aufnahme in MP4 umzuwandeln.

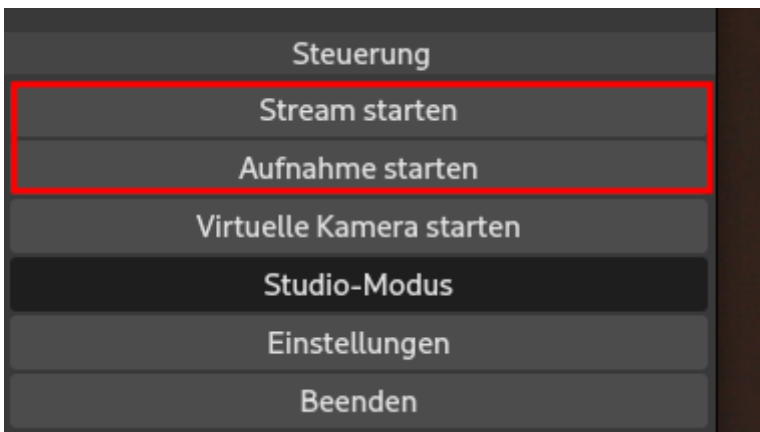
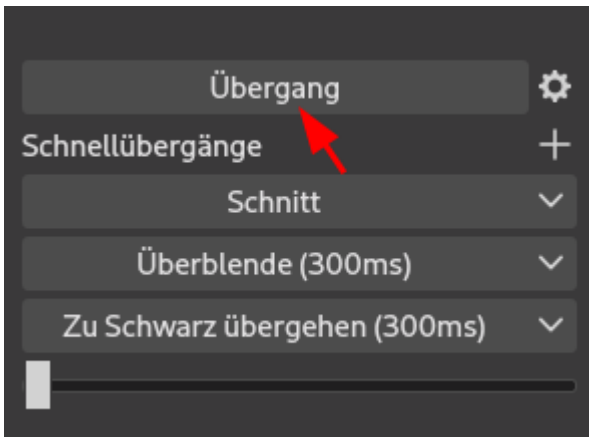


## Live gehen und Aufnahme

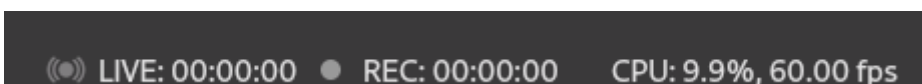
Da wir alle Bild und Audio-Quellen hinzugefügt haben, welche wir streamen möchten und OBS die Video und Stream Einstellungen konfiguriert haben, können wir mit dem Streamen beginnen. Wählt die Szene aus, welche ihr Streamen möchtet und klickt zwischen dem Vorschau und Programm

Fenster auf Übergang um aus der Vorschau das Livebild zu machen und klickt dann unten Rechts auf einer oder mehrere der Start Funktionen.

- Stream starten
  - Startet den Stream zum eingestellten Streaminganbieter (in diesem Beispiel Twitch).
- Aufnahme starten
  - Die Szene wird lokal gespeichert.
- Virtuelle Kamera starten
  - Die Szene wird als Virtuelle Kamera ausgegeben und kann dann zum Beispiel im Big Blue Button als Webcam benutzt werden.



Sofern keine Fehlermeldung kommt, ist der Stream Live. Um zu sehen ob der Stream oder die Aufnahme (noch) läuft, zeigt OBS die dauer der Aufnahme und Streams im Footer an. Sollte der Stream unterbrochen sein, wird dies in dieser Anzeige dargestellt. Als weitere Sicherheit, kann die Seite, auf der der Stream läuft aufgerufen und gecheckt werden (benötigt weitere Bandbreitenkapazität). Zur Beendigung von der Aufnahme oder dem Stream klickt man auf die Schlatfläche, wo zuvor noch starten stand: Aufnahme oder Stream stoppen.



# FFmpeg Basics

Bei **FFmpeg** handelt es sich um eine mächtige Sammlung an freien Computerprogrammen und Programmbibliotheken, die digitales Video- und Audiomaterial aufnehmen, konvertieren, senden, filtern und in verschiedene Containerformate verpacken können. FFmpeg wird zu meist über die Kommandozeile bedient, es bestehen aber auch unterschiedliche grafische Oberflächen für die unterschiedlichsten Einsatzzwecken. Zum Beispiel Videoschnitt Programme, welche auf FFmpeg aufbauen oder Programme um Videos von zum Beispiel DVDs herunter zu bekommen. Auf dieser Seite beschäftigen wir uns mit dem streamen eines Desktops mithilfe FFmpeg auf der Kommandozeile.

---

## Download und Installation

FFmpeg steht für verschiedene Betriebssystemen zur Verfügung und kann von der offiziellen Seite, sowie aus den Paketquellen des jeweiligen Paketmanagers heruntergeladen werden.

### Windows

Installer herunterladen und ausführen:

**FFmpeg:** <https://ffmpeg.org/download.html#build-windows>

**GitHub Source:** <https://github.com/obsproject/obs-studio/releases>

Aus dem Quellcode kompilieren

**Dokumentation:** <https://trac.ffmpeg.org/wiki/CompilationGuide>

### macOS

Installer herunterladen und ausführen:

**FFmpeg:** <https://ffmpeg.org/download.html#build-mac>

**GitHub Source:** <https://github.com/obsproject/obs-studio/releases>

Aus dem Quellcode kompilieren

**Dokumentation:** <https://trac.ffmpeg.org/wiki/CompilationGuide>

Mittels `brew` installieren:

```
brew install ffmpeg
```

## macOS

Installer herunterladen und ausführen:

**FFmpeg:** <https://ffmpeg.org/download.html#build-mac>

**GitHub Source:** <https://github.com/obsproject/obs-studio/releases>

Aus dem Quellcode kompilieren

**Dokumentation:** <https://trac.ffmpeg.org/wiki/CompilationGuide>

Mittels `brew` installieren:

```
brew install ffmpeg
```

## GNU/Linux

### details

Die Installation für Linux kann auf verschiedenen Wegen erfolgen.

### Ubuntu:

```
sudo apt update
sudo apt install ffmpeg
```

### Arch Linux und Manjaro:

#### details

```
sudo pacman -S ffmpeg
```

Im AUR befinden sich verschiedene PKBUILDS um FFmpeg in unterschiedlichen Ausführungen zu bauen:

- ffmpeg-full
  - Die aktuelle stabile Version von FFmpeg wird komplett aus dem Quellcode gebaut.
- ffmpeg-full-git
  - Der aktuelle Release Candidate von FFmpeg wird komplett aus dem Quellcode gebaut.
- ffmpeg-amd-full
  - FFmpeg wird in der aktuellsten stabilen Version für AMD Hardware gebaut.
- ffmpeg-amd-full-git
  - Der aktuelle Release Candidate von FFmpeg wird für AMD Hardware gebaut.
- ffmpeg-cuda
  - FFmpeg wird in der aktuellen stabilen Version mit CUDA (Nvidia) Unterstützung gebaut.

## Debian:

### details

```
sudo apt update  
sudo apt install ffmpeg
```

## Fedora:

### details

```
sudo dnf -y install ffmpeg
```

## Gentoo:

### details

```
emerge --ask media-video/ffmpeg
```



## NixOS:

### details

<https://github.com/NixOS/nixpkgs/blob/master/pkgsets/development/libraries/ffmpeg-full/default.nix>

## OpenMandriva Lx4

### details

```
dnf install ffmpeg
```

## Solus

### details

```
eopkg install ffmpeg
```

## Void

### details

```
sudo xbps-install ffmpeg
```

# Einführung

Es bestehen unterschiedliche Möglichkeiten mittels FFmpeg zu streamen. Der Einsatzzweck kann hierbei sehr unterschiedlich sein. Zum Beispiel soll ein bestehender RTMP Stream abgegriffen und umgeleitet werden, oder es befinden sich Videos auf einem Server, welche gestreamt werden sollen, damit der eigene PC nicht dauerhaft laufen muss.

Für diesen Wikiartikel wurde `FFmpeg` in Version **n5.0.1** unter Arch Linux verwendet. Die Syntax, Optionen oder Argumente können je nach Version und Betriebssystem abweichen.

Als Beispiel Zielsever wird eine eigene Owncast Instanz gewählt.

## Lokales Video streamen

Um mit FFmpeg ein lokales Video an ein Zielsever zu streamen muss an FFmpeg die Quelle mittels `-i` angegeben werden, sowie das Ziel zum Schluss angehangen werden. Dazwischen können Optionen gesetzt werden. Zum Beispiel, ob das Video vor dem streamen konvertiert werden soll.

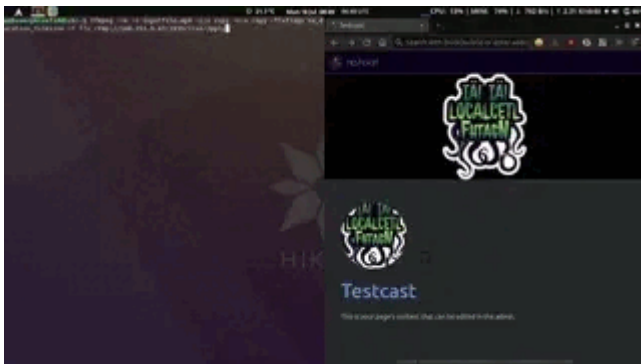
```
ffmpeg -re -i inputfile.mp4 -c:v copy -c:a copy -f flv rtmp://127.0.0.1:1935/live/<streamkey>
```

Mit dem Befehl von oben liest ffmpeg die datei inputfile.mp4 ein und sendet das Video als flv zum Zielsever, welcher via RTMP auf Port 1935 erreichbar ist. Folgende Optionen wurden an FFmpeg mit angeheftet:

- `-re` Der Input wird mit nativer Bildwiederholungsrate eingelesen.
- `-i` Mittels dieser Option wird der Pfad zur Datei gesetzt.

Wichtig. Der Dateiname darf keine Leerzeichen haben. FFmpeg hat hiermit Probleme, auch wenn man diese escaped.

- `-c:v copy` Gibt an, dass der Video Codec beibehalten werden soll und FFmpeg nichts en/decodieren muss.
- `-a:v copy` Wie beim Video Codec wird auch der Audio Codec beibehalten.
- `-f flv` Mit dieser Option wird das Format FLV erzwungen, welches zwingend beim RTMP Streams ist.



Die Optionen `-c:v copy` und `-a:v copy` sind wichtig zu setzen, sollte der jeweilige Computer langsamer en/decodieren, als mit 1x Geschwindigkeit. Im Fall einer langsamen berechnung des Videos, wird FFmpeg den Stream abbrechen, da das Video langsamer berechnet wird als FFmpeg senden kann und somit Frames erwartet, welche FFmpeg noch nicht kennt. Es kommt zu einem `Broken Pipe` und `connection reset by peer`.

## Weitere Optionen

Sollte eine Grafikkarte vorhanden sein, mit welcher ein reencoding schnell genug durchgeführt werden kann, kann das Video während des streamens entsprechend auf Größe und Format gebracht werden. Dies ist besonders interessant, falls man das Video nicht im Vorfeld skaliert hat und die Bandbreite im Upload begrenzt ist.

`-video_size 1280x720` Gibt die Ausgabeauflösung an  
`-c:v libx264` mit der Option wird der H264 Codec benutzt  
`-preset veryfast` Das vorgefertigte Profil für eine schnelle Berechnung, anstelle von Qualität wird benutzt  
`-b:v 3000k` Die Bitrate des Videos wird auf 3000 KB/s eingestellt  
`-maxrate 3000k` Die maximale Bitrate des Videos entspricht 3000 KB/s  
`-bufsize 6000k` Die Buffer größe muss das doppelte der maximalen Bitrate entsprechen.  
`-g 60` Die g Option stellt die Group Of Picture länge an. Hohe werte verursachen eine größere Kompression.  
`-c:a aac` Als Audio Ausgabe Codec soll aac benutzt werden  
`-b:a 128k` Dies stellt ein mit welcher Bitrate das Audio im Video gesendet werden soll. 192KB/s entsprich CD Qualität  
`-ar 44100` Dies stellt die Samplerate Hz ein.

## Streamen von Webcam und Mikrofon (Linux)

Falls eine USB Webcam gestreamt werden muss die Audioquelle, sowie die Video-Hardware angegeben werden. Um die Audio Hardware zu bestimmen, nutzen wir `$ aplay --list-devices` und listen jegliche Hardware auf.

```
$ aplay -l
**** Liste der Hardware-Geräte (PLAYBACK) ****
Karte 0: PCH [HDA Intel PCH], Gerät 0: ALC892 Analog [ALC892 Analog]
  Sub-Geräte: 0/1
  Sub-Gerät #0: subdevice #0
Karte 1: NVidia [HDA NVidia], Gerät 3: HDMI 0 [HDMI 0]
  Sub-Geräte: 1/1
  Sub-Gerät #0: subdevice #0
Karte 1: NVidia [HDA NVidia], Gerät 7: HDMI 1 [HDMI 1]
  Sub-Geräte: 1/1
  Sub-Gerät #0: subdevice #0
Karte 1: NVidia [HDA NVidia], Gerät 8: HDMI 2 [HDMI 2]
  Sub-Geräte: 1/1
  Sub-Gerät #0: subdevice #0
Karte 1: NVidia [HDA NVidia], Gerät 9: HDMI 3 [HDMI 3]
  Sub-Geräte: 1/1
  Sub-Gerät #0: subdevice #0
Karte 1: NVidia [HDA NVidia], Gerät 10: HDMI 4 [HDMI 4]
```

Sub-Geräte: 1/1

Sub-Gerät #0: subdevice #0

Karte 1: NVidia [HDA NVidia], Gerät 11: HDMI 5 [HDMI 5]

Sub-Geräte: 1/1

Sub-Gerät #0: subdevice #0

Karte 1: NVidia [HDA NVidia], Gerät 12: HDMI 6 [HDMI 6]

Sub-Geräte: 1/1

Sub-Gerät #0: subdevice #0

Karte 2: Creative [HDA Creative], Gerät 0: ALC898 Analog [ALC898 Analog]

Sub-Geräte: 0/1

Sub-Gerät #0: subdevice #0

Karte 3: Device [USB Advanced Audio Device], Gerät 0: USB Audio [USB Audio]

Sub-Geräte: 0/1

Sub-Gerät #0: subdevice #0

In der Ausgabe können wir sehen, welche Audio-Hardware vom Betriebssystem erkannt wurde und zur Verfügung steht. Nun müssen wir die passende `Karte` und `Subdevice` wählen. In diesem Beispiel will ich die Karte 3, das USB Mikrofon benutzen. Sprich, meine Hardware ist die `3,0`.

Alternative Möglichkeiten die Hardware zu definieren kann hier nachgelesen werden:

[https://en.wikibooks.org/wiki/Configuring\\_Sound\\_on\\_Linux/HW\\_Address](https://en.wikibooks.org/wiki/Configuring_Sound_on_Linux/HW_Address)

Nun muss ich die Video-Hardware Bestimmen, welche genutzt werden soll. Diese befindet sich unter `/dev/video*`. Diese wird als Input mit `-i` definiert. In meinen Fall ist es die Webcam unter dem Pfad `/dev/video2`. Nun muss zusätzlich das Input Format, sowie das ausgehende Videoformat gewählt werden. Der Input wird mit `-input_format yuyv422` eingestellt. Dieses Format kommt von der Webcam und sollte für den ausgehenden Stream in `yuv420p` mit `-vf "format=yuv420p"` umgewandelt werden. Mit der Kombination aus den vorherigen Befehlen und zusätzlichen Optionen sieht der gesamte Befehl wie folgt aus:

```
ffmpeg -f alsa -ac 2 -i hw:3,0 \
-f v4l2 -framerate 60 -video_size 1280x720 -input_format yuyv422 -i /dev/video2 \
-c:v libx264 -preset veryfast -b:v 1984k -maxrate 1984k -bufsize 3968k \
-vf "format=yuv420p" -g 60 -c:a aac -b:a 128k -ar 44100 \
-f flv rtmp://127.0.0.1:1935/live/<streamkey>
```

# Streaming Konzepte

Welche Software kann ich benutzen, um von meinem Computer ins Internet zu streamen?

# Einfaches Setup

# Event: Talk Konzept

Auf dieser Seite soll erklärt werden, welche Anforderungen vom Stream Team und den Vortragenden benötigt werden. Des Weiteren wird hier auch die Personale Planung beschrieben, sowie zum Beispiel auf die [Checkliste](#) verwiesen, bevor ein Talk statt findet.

# Team: Studio, Regie, Technik



Streaming Konzepte

# ffmpeg

# Streaming Endpoints

Wohin kann ich streamen?

Wie kann ich einen eigenen Server bereitstellen?

Streaming Endpoints

# Twitch

Streaming Endpoints

# media.ccc.de

Streaming Endpoints

Google

# Owncast

Owncast Endpoint

Owncast selber hosten

Streaming Endpoints

# Nginx

# OBS: Troubleshooting



# FAQ

# Einkaufsliste

Die Remote Rhein Ruhr Stage hat uns deren Einkaufsliste geteilt, mit simplen Peripherien welche getestet wurden und funktionieren. Diese Liste dient für die Vortragenden, die sonst keine oder schlechte Gerätschaften haben. Diese Liste hat keinen Mehrwert für Personen mit einer professionellen Ausstattung und dient nicht als Empfehlung für ein professionelles Setup.

Die Einkaufsliste ist eine Empfehlung der Remote Rhein Ruhr Stage und wurde von mir (Kascha) noch nicht verifiziert.

## Video

### Jiga echte 1080p-USB-Webcam



1080P Webcam mit Mikrofon, Full HD Facecam Streaming Webcams, USB Kamera mit Ringlicht, Stativ, 360° Schwenken, JIGA Web Camera für PC, Videochat, Laptop, Zoom, Skype (Weiß, Warmes, Natürliches Licht)

Marke: JIGA

★★★★☆ 538 Sternebewertungen

16<sup>99</sup> €

✓prime

& KOSTENFREIE Retouren

Preisangaben inkl. USt. Abhängig von der Lieferadresse kann die USt. an der Kasse variieren. [Weitere Informationen.](#)

Marke

JIGA

- [Ebay](#)

## Ton

### MPOW 3,5mm + USB-Adapter Headset



## Mpow USB Kopfhörer PC Laptop Headset 3,5mm Kabelgebunden Stereo Mikrofon Telefon

Artikelzustand: **Neu**

Multi-Rabatt:

1 Kaufen  
EUR 15,03/Stk.

2 kaufen  
EUR 13,83/Stk.

EU

Stückzahl:  4 oder mehr für EUR 12,78/Stk.

Mehr als 10 verfügbar / [27 verkauft](#)

Preis: **EUR 15,03/Stk.**  
(inkl. MwSt.)  
~~EUR 15,99~~ Sparen Sie 6%

Sofort

In den W

Auf di  
Beobachtu

- [Ebay](#)
- [Amazon](#)

# Licht

LED-Strip, flackerfrei mit vielen LEDs, an der Wand hinterm Monitor “wild” hinhängen für blendarme Ausleuchtung (15-20€)



## Govee LED Strip 5m Warmweiss 3000K LED Lichtband LED Streifen 300 LEDs warmweiß dimmbar für Spiegel Deko Party Küche Warmweiß [Energieklasse A]

Besuche den Govee-Store



2.202 Sternebewertungen

Amazons Tipp für "led strip warmweiss"

18<sup>99</sup> €

✓prime

& KOSTENFREIE Retouren

Preisangaben inkl. USt. Abhängig von der Lieferadresse kann die USt. an der Kasse variieren. Weitere Informationen.

A

Art der Lichtquelle	LED
Zur Verwendung im Innen- und Außenbereich	Innenbereich
Farbe	Warmweiß
Besonderes	Dimmbar

- [Amazon](#)
- [Ebay](#) (5m, warm, indoor, 3A-Netzteil)

## Sonstiges

Klemmstativ für Webcam



## Webcam Halterung-64cm Klemme Tisch Halter, Webcam ständer für Live Stream für Logitech Webcam c925e, C922 X, C930e, C922, C930, C920, C615, für GoPro Hero 8/7/6/5, Arlo Ultra/Pro/Pro 2/Pro 3/Brio 4 K

Besuche den AnkePower-Store



182 Sternebewertungen

Amazons Tipp für "webcam halterung"

23<sup>99</sup> €

✓prime

& KOSTENFREIE Retouren

- [Ebay](#)
- [Amazon](#)