EFG Bad Lausick - Dokumentationen

Liveübertragung von Gottesdiensten

Bernd Reuther

Bernd Reuther

Inhaltsverzeichnis

1. Liveübertragung von Gottesdiensten

letzte Änderung: 14.05.2020

In dieser Dokumentation beschreiben wir, wie die Gottesdienste in unserer Gemeinde live über das Internet angesehen werden können. Die Übertragung erfolgt bei uns nicht öffentlich, sondern ist für Gemeindemitglieder und Freunde gedacht, die aus verschiedenen Gründen den Gottesdienst nicht besuchen können, aber mit unserer Gemeinde verbunden sind. Wir streamen daher nicht auf bekannte Plattformen wie YouTube, Facebook, o.ä., sondern verwenden dafür einen gemieteter Internet Server. Die grundsätzliche Vorgehensweise lässt sich aber auch an andere Anwendungsbereiche oder Streaming-Ziele anpassen.

Info

Während der Corona-Krise 2020 erfolgte die Übertragung der Gottesdienste ausschließlich online und wir bekamen sehr positives Feedback, dass dadurch die Verbundenheit miteinander erhalten blieb.

1.1 Ausstattung

1.1.1 Hardware

- Kamera
- HDMI Audio Inserter
- Encoder
- Audio-Mischpult
- PC
- Server

1.1.2 Software

- MistServer
- OBS-Studio

1.2 Kamera

Als Kamera nutzen wir eine Canon LEGRIA HF R77. Das ist ein "normaler" Camcorder aus dem Heimvideo-Bereich. Wichtig ist hier ein HDMI-Ausgang, der das Live-Bild ausgibt und die Möglichkeit, den Camcorder auch ohne Aufnahme im Dauerbetrieb laufen zu lassen. Einige Camcorder schalten nach einer gewissen Zeit im Leerlauf ab. In diesem Fall könnte man aber auch einfach die Aufnahme aktivieren, sofern das Live-Bild trotzdem über den HDMI-Ausgang gesendet wird.

^ *zum Anfang*

1.3 HDMI-Audio-Inserter

Der HDMI-Ausgang der Kamera ist zunächst mit einem sogenannten "HDMI-Audio-Inserter" verbunden. Wir verwenden das Modell HM-CV032K von SPEAKA. Wie der Name schon sagt, ist der Zweck dieses kleinen Kastens, ein externes Audiosignal in den HDMI-Datenstrom der Kamera einzufügen bzw. den Originalton der Kamera zu ersetzen. Natürlich könnte man auch den Ton der Kamera verwenden, die Qualität wäre dann aber deutlich geringer, als bei einem direkt vom Audiomischpult abgegriffenen Signal.

Wenn man den Ton der Kamera nicht verwenden möchte, ist auf jeden Fall das Einfügen des Audiosignals **vor** dem Encoder notwendig, um die Synchronität von Bild und Ton beizubehalten. Software-seitig (im OBS-Studio) könnte man auch noch Tonspuren einfügen, die würden dann aber im fertigen Stream zeitversetzt vor dem Bild laufen, weil der Encoder den HDMI-Datenstrom leicht verzögert ausgibt.

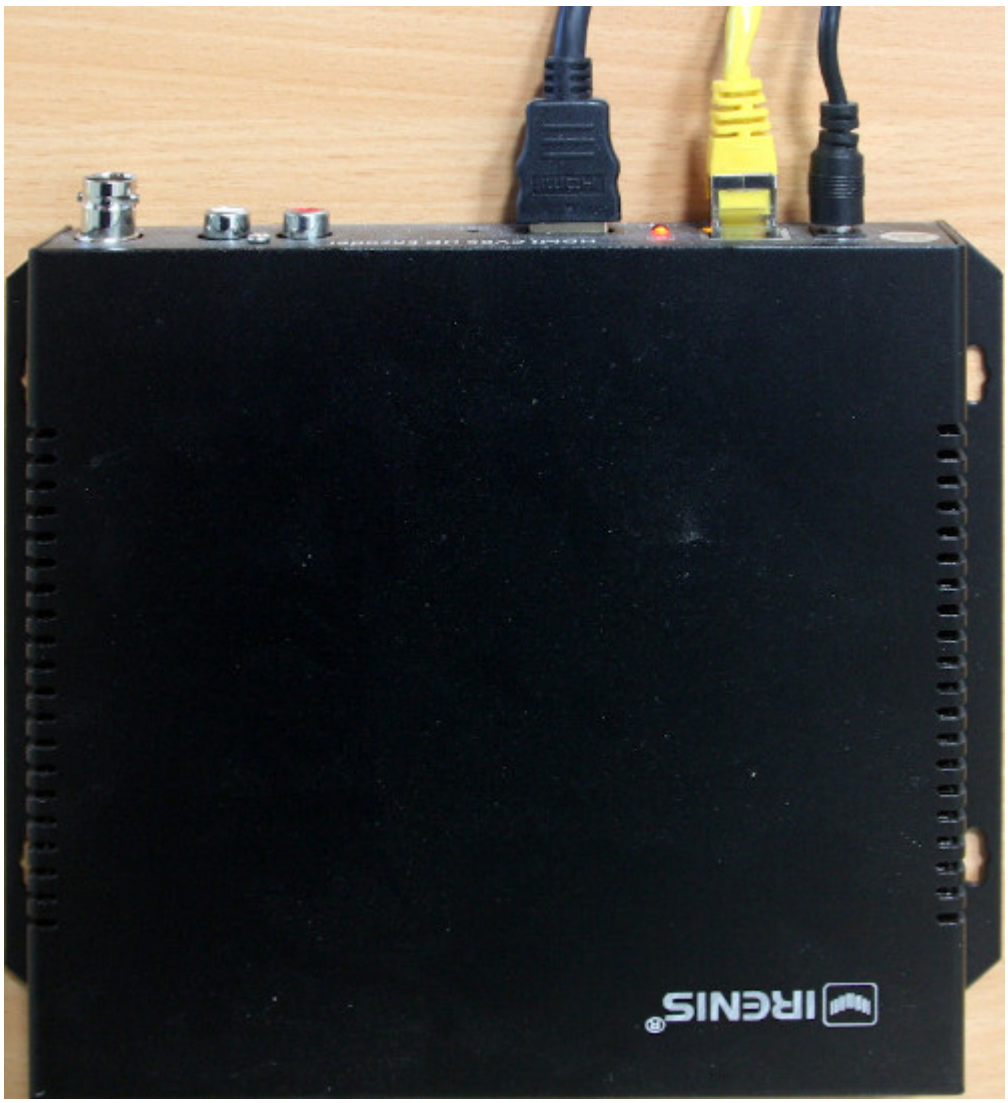
> **Info**
>
> Neuere Encoder haben die Audio-Insert-Funktion bereits integriert und man kann das Audiosignal direkt in den Encoder einspeisen. Das von uns verwendete Modell ist eine Vorgängerversion der aktuellen Serie und besitzt keinen Audio-Eingang.

^ *zum Anfang*

1.4 Encoder

Obwohl man auch mit Hilfe eines HDMI-Grabbers (HDMI-Capture Karte) den HDMI-Datenstrom direkt in den Computer übertragen könnte, haben wir uns aus verschiedenen Gründen für eine externe Lösung in Form eines IPTV-Encoders entschieden:

- Ein externes Gerät entlastet den Computer. In unserem Fall läuft darauf neben dem OBS-Studio auch noch OpenLP damit das Beamer-Bild in den Video-Stream integriert werden kann
- Der von uns verwendete Encoder IRENIS ADE-264 kann 2 Streams mit unterschiedlichen Auflösungen und Kompressionsraten erzeugen. Wir nutzen das, um einen Stream mit höherer Auflösung in unserem Eltern-Kind-Raum an einem großen Monitor anzuzeigen. Der zweite Stream wird über das OBS-Studio an unseren Internet-Server geschickt.



^ *zum Anfang*

1.5 Audio-Mischpult

Die Einstellungen des Audio-Mixers, der für die Beschallung unserer Gottesdienste verwendet wird, entsprechen in den meisten Fällen nicht denen, die für eine Aufnahme bzw. Übertragung notwendig sind. Das reicht von unterschiedlichen Lautstärkeverhältnissen zwischen Musik und Sprache bis zu den Klang- und Dynamikeinstellungen der einzelnen Kanäle. Neuere und größere Digital-Mischpulte, wie das von uns verwendete Soundcraft Si Impact bieten die Möglichkeit, einen Sub-Mix zu erstellen und an separate Ausgänge zu legen. Weil das ziemlich komplexe Einstellungen sind und unerfahrene Techniker schnell an ihre Grenzen bringt, verwenden wir ein zusätzliches kleines Digital-Mischpult für den Video-Ton.

Das Soundcraft Ui12 ist sehr kompakt und wird über eine Netzwerkverbindung mit einem Browser bedient. Das hat natürlich auch den Vorteil, dass man die Einstellungen z.B. vom Eltern-Kind Übertragungsraum aus vornehmen kann und so eine direkte Kontrolle des Ergebnisses hat. In der aktuellen Einstellung wird das Summen-Signal vom "großen Bruder" und zusätzlich noch einmal der Mikrofon-Kanal des Rednerpults eingespeist. Die Lautstärke des Mikrofon-Kanals wird mit Hilfe der Dynamik-Effekte (Kompressor/Limiter/Gate) angepasst. Als weitere "Ausbaustufe" wären z.B. zusätzliche Raum-Mikrofone denkbar, die das Live-Empfinden der Zuschauer verbessern, da die ganze Raumakustik (Gemeindegesang,...) mit eingefangen wird.

^ *zum Anfang*

1.6 PC

Als PC verwenden wir einen handelsüblichen Büro-Computer mit Ubuntu als Betriebssystem. Wichtig ist hier, dass der Computer über ausreichend Leistung verfügt, um Lieder und Multimedia-Inhalte mit OpenLP am Beamer anzuzeigen und gleichzeitig den Video-Stream mit OBS-Studio an den Server zu senden. Das Video-

Signal wird hierbei teilweise mit dem Beamer-Bild überlagert und muss demzufolge noch einmal neu encodiert werden (ein Intel i3 mit 8GB RAM ist hier bereits ausreichend).

^ *zum Anfang*

1.7 Server

Für die Bereitstellung des Video-Live-Streams über das Internet verwenden wir einen gemieteten root-Server. Die detaillierte Beschreibung der Einrichtung des Servers würde hier zu weit führen. Wer einen eigenen Server betreibt, sollte auf jeden Fall über ausreichende Kenntnisse in diesem Bereich verfügen. Die erforderliche Leistung des Servers hängt von der Anzahl der gleichzeitigen Zuschauer des Live-Streams ab. Neben der reinen Prozessorleistung bzw. Arbeitsspeichergröße, ist dabei auch die Netzwerk-Last zu beachten. Für unseren Live-Stream verwenden wir einen root-Server mit 4 Kernen und 16GB Ram. Bei 48 Clients liegt die CPU-Last bei etwa 30%. Der Downstream (Server zu Clients) erzeugt ca. 80 MBit/s.

^ *zum Anfang*

1.8 MistServer

1.8.1 Installieren

Der MistServer ist eine OpenSource Software, mit deren Hilfe man sehr einfach und komfortabel einen Video-Stream im Internet zur Verfügung stellen kann. Die Installation der OpenSource-Version ist unter https://mistserver.org/download beschrieben und mit einer Zeile auf der Linux-Konsole erledigt:

> **Info**
>
> Die Versionsnummer in dem Befehl kann zu einem anderen Zeitpunkt natürlich anders sein

```
curl -o - https://releases.mistserver.org/is/mistserver_64V2.17.tar.gz 2>/dev/null | sh
```

> **Achtung**
>
> Nach dem Ausführen des Befehls startet der MistServer automatisch und ist ungesichert über den Port 4242 erreichbar (wenn keine Firewall aktiv ist)!

Der MistServer kommuniziert standardmäßig über folgende Ports, die gegebenenfalls in der Firewall zugelassen werden müssen.

- TCP 1935 - RTMP-Port für Upload von OBS-Studio bzw. Download über entsprechende Programme
- TCP 4242 - Management-Interface
- TCP 8080 - HTTP-Port für Streaming-Clients
- TCP 4433 - HTTPS-Port für Streaming-Clients

Die Ports 4242 und 8080 müssen nicht freigegeben werden, wenn vor dem MistServer noch ein anderer Webserver (wie z.B. Apache) als ReverseProxy eingerichtet ist.

Wir nutzen den Apache-Webserver mit folgenden Einstellungen (Servernamen als Beispiel):

```
<VirtualHost *:80>

    ServerName video.domain.tld

    DocumentRoot /var/www/html/

    SSLEngine off

    RewriteEngine on
    RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}

</VirtualHost>

<VirtualHost *:443>

    ServerName video.domain.tld

    DocumentRoot /var/www/html/

    SSLEngine on
    SSLCertificateFile /etc/letsencrypt/live/domain.tld/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/domain.tld/privkey.pem

    ProxyRequests off
    ProxyPreserveHost On
    SetOutputFilter proxy-html

    <Location "/">

        RewriteEngine on
        RewriteCond %{HTTP:UPGRADE} ^WebSocket$ [NC]
        RewriteCond %{HTTP:CONNECTION} Upgrade$ [NC]
        RewriteRule .* ws://localhost:8080%{REQUEST_URI} [P]

        ProxyPass http://localhost:8080/
        ProxyPassReverse http://localhost:8080/
        ProxyHTMLURLMap http://localhost:8080 /

    </Location>

    <Location "/admin">

        AuthType Basic
        AuthName "Restricted Content"
        AuthUserFile /etc/apache2/.video-admin
        Require valid-user

        RewriteEngine on
        RewriteCond %{HTTP:UPGRADE} ^WebSocket$ [NC]
        RewriteCond %{HTTP:CONNECTION} Upgrade$ [NC]
        RewriteRule .* ws://localhost:4242%{REQUEST_URI} [P]

        ProxyPass http://localhost:4242/
        ProxyPassReverse http://localhost:4242/
```

```
        ProxyHTMLURLMap http://localhost:4242/

    </Location>

    # wenn das LetsEncrypt Zertifikat per webroot Plugin erneuert wird,
    # muss der entsprechende Ort vom Proxy ausgenommen werden
    <Location "/.well-known/acme-challenge">
        ProxyPass "!"
    </Location>

    CustomLog /var/log/apache2/video.domain.tld.log combined
    ErrorLog /var/log/apache2/video.domain.tld.error.log

</VirtualHost>
```

Hinweis

Leider ist das Management-Interface in der OpenSource-Version nur unverschlüsselt über `http` erreichbar. Wenn man Port 4242 in der Firewall sperrt und einen den Webserver als ReverseProxy konfiguriert, funktioniert auch `https`. Wir haben aber festgestellt, dass dann das Management-Interface ohne Anmeldung auch von anderen Clients aus erreichbar war. Laut MistServer-Support funktioniert der Zugriff von `localhost` aus immer ohne Anmeldung. Und genau das macht ja der Proxy: er übersetzt die externe Anfrage nach `localhost:4242`. Aus diesem Grund haben wir in der Webserver-Konfiguration eine zusätzliche Authentifizierung eingerichtet.

Die Datei `/etc/apache2/.video-admin` kann wie folgt erstellt werden:

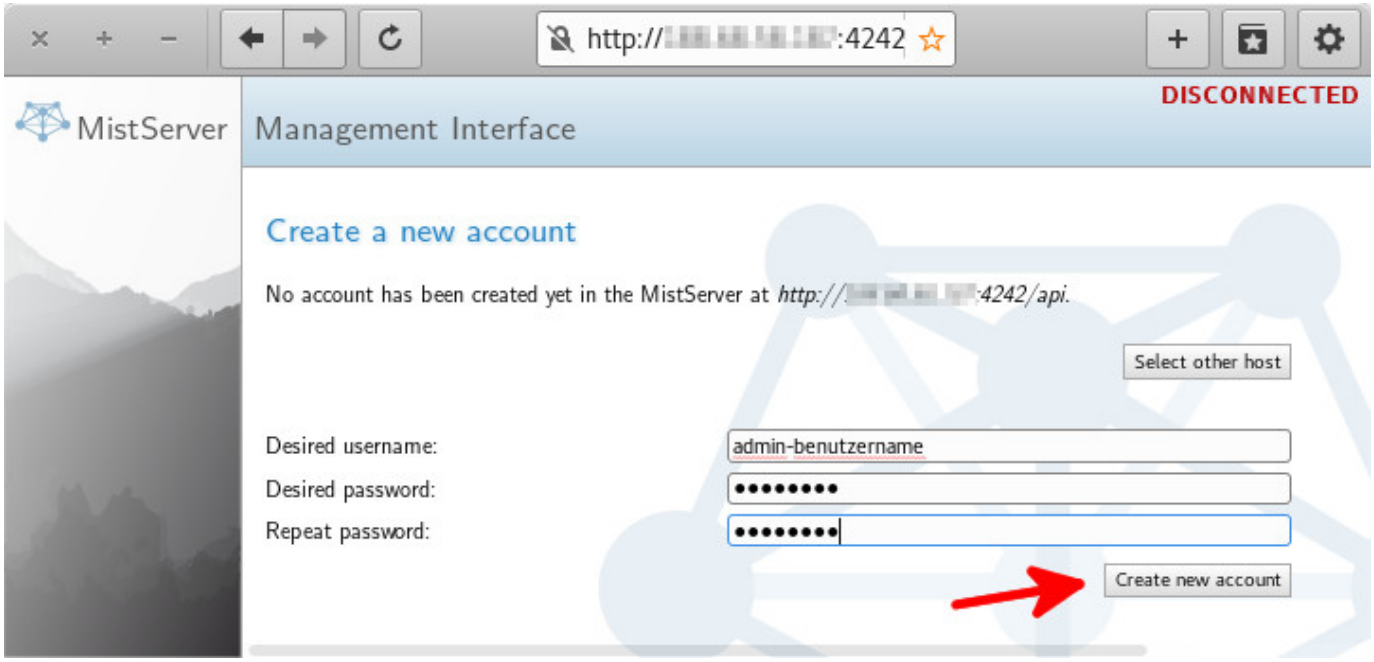
```
apt install apache2-utils
htpasswd -c /etc/apache2/.video-admin <admin-benutzername>
New password: <admin-passwort>
Re-type new password: <admin-passwort>
```

Mit diesen Einstellungen ist das Management-Interface vom MistServer nur noch über `https://video.domain.tld/admin` erreichbar und die Video-Streams unter `https://video.domain.tld/<Streamnummer>.html`. RTMP läuft weiterhin über Port 1935 und mit einem entsprechenden Programm (wie z.B. VLC-Player) können die Streams auch unter `rtmp://video.domain.tld:1935/play/<Streamnummer>` angesehen werden.

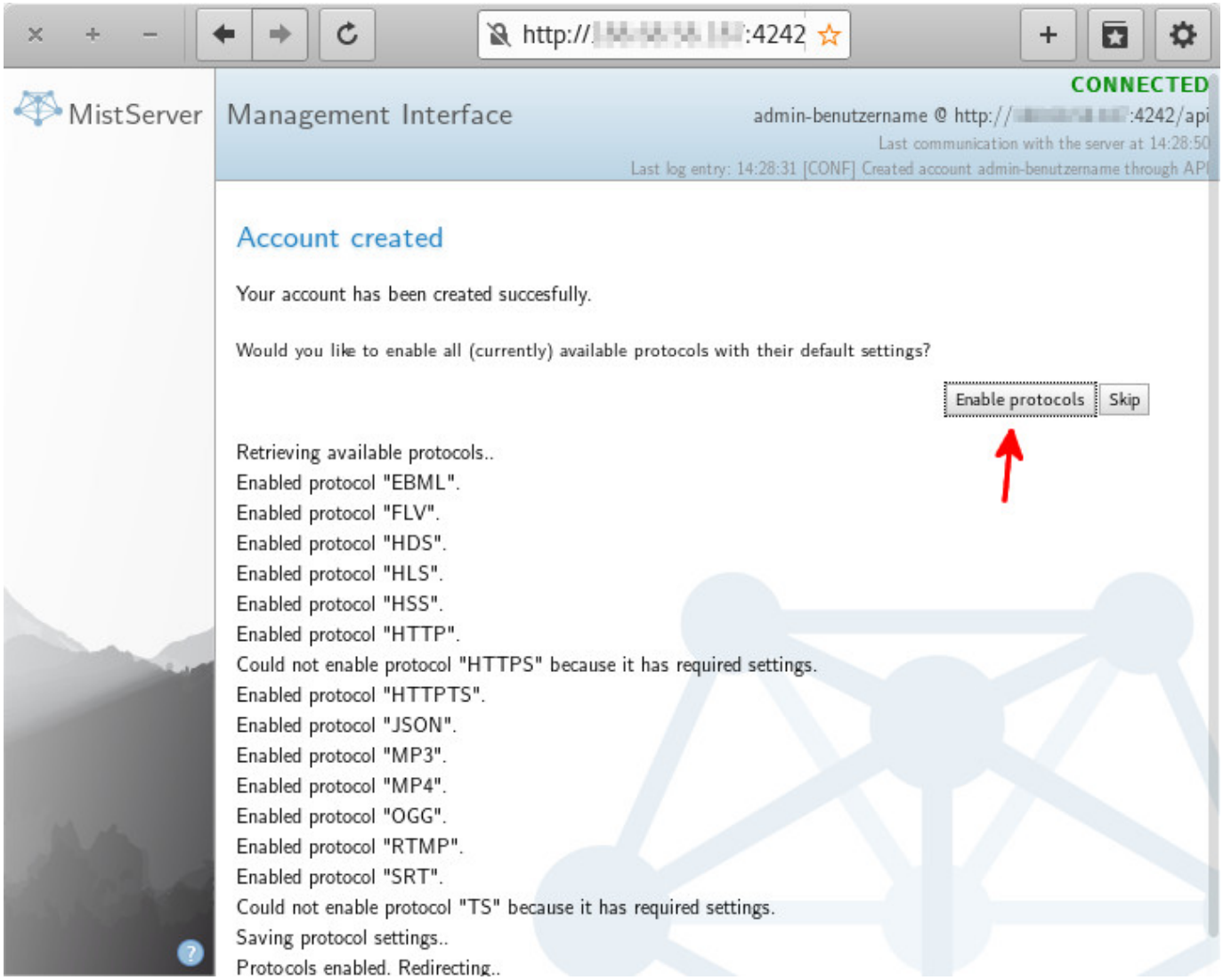
^ *zum Anfang*

1.8.2 MistServer konfigurieren

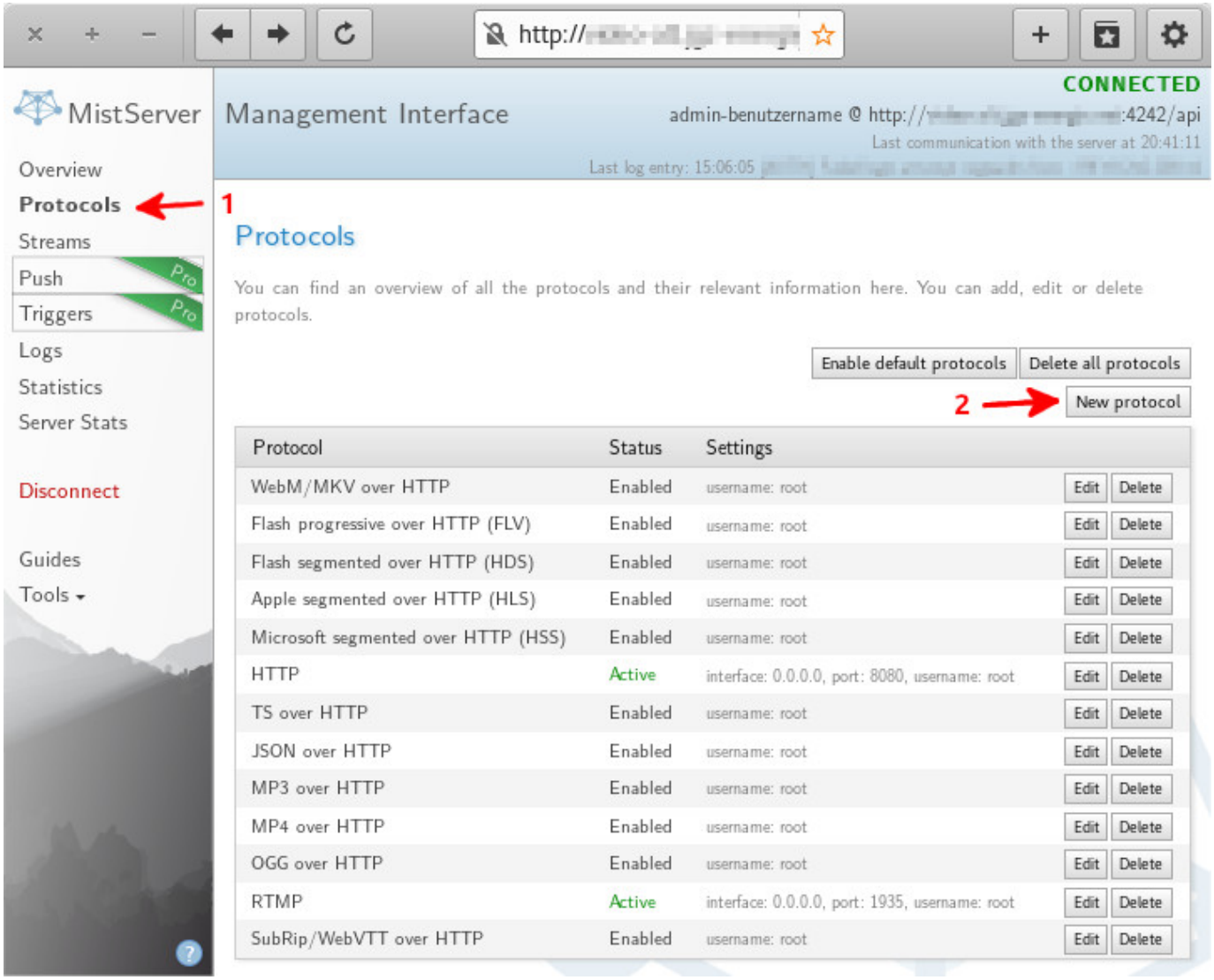
Wenn man nach der Installation des MistServers das erste mal die Webseite des Management-Interfaces startet, wird man aufgefordert, einen Benutzernamen und ein Passwort zu vergeben.

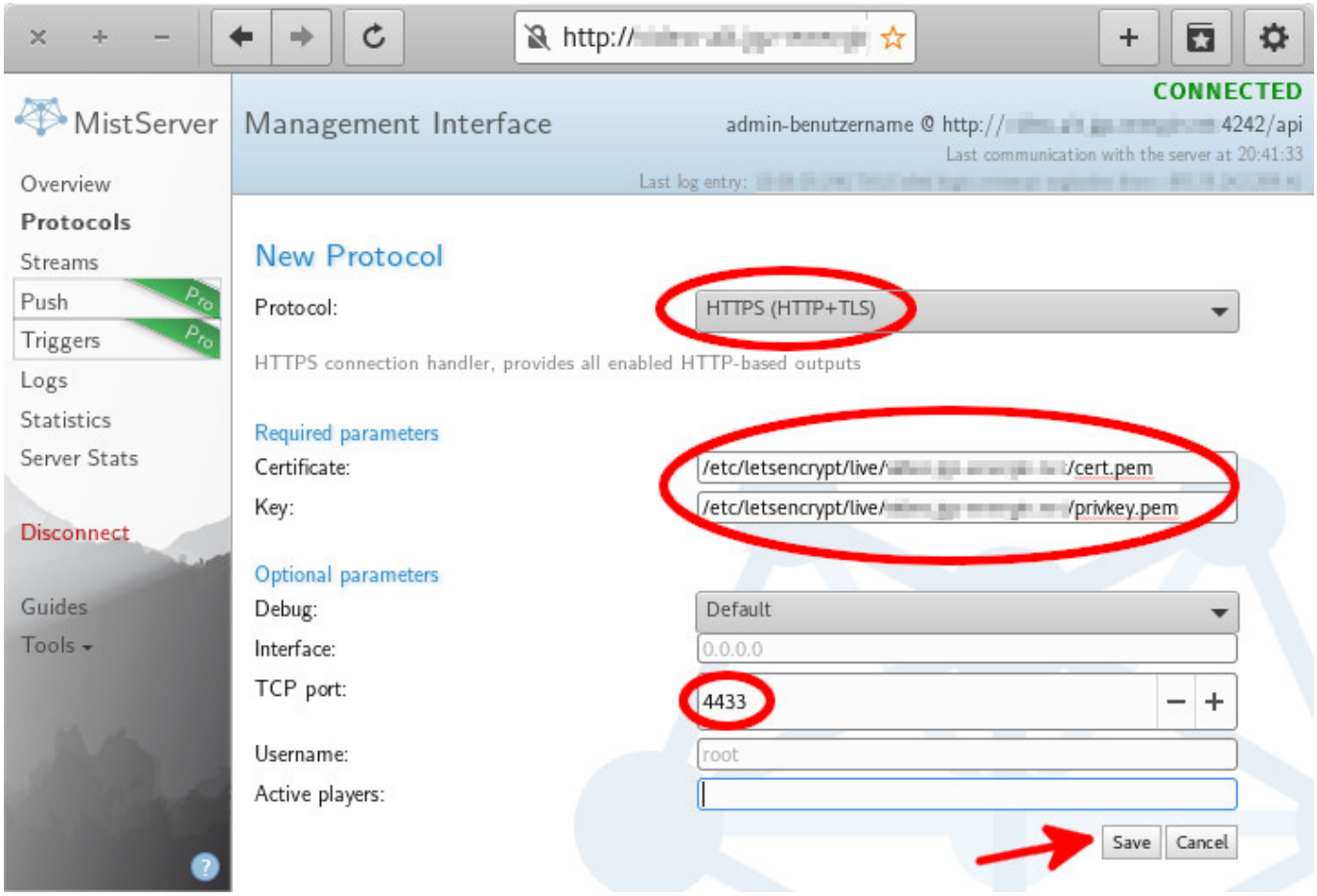


Im nächsten Schritt werden alle Standard-Protokolle aktiviert.

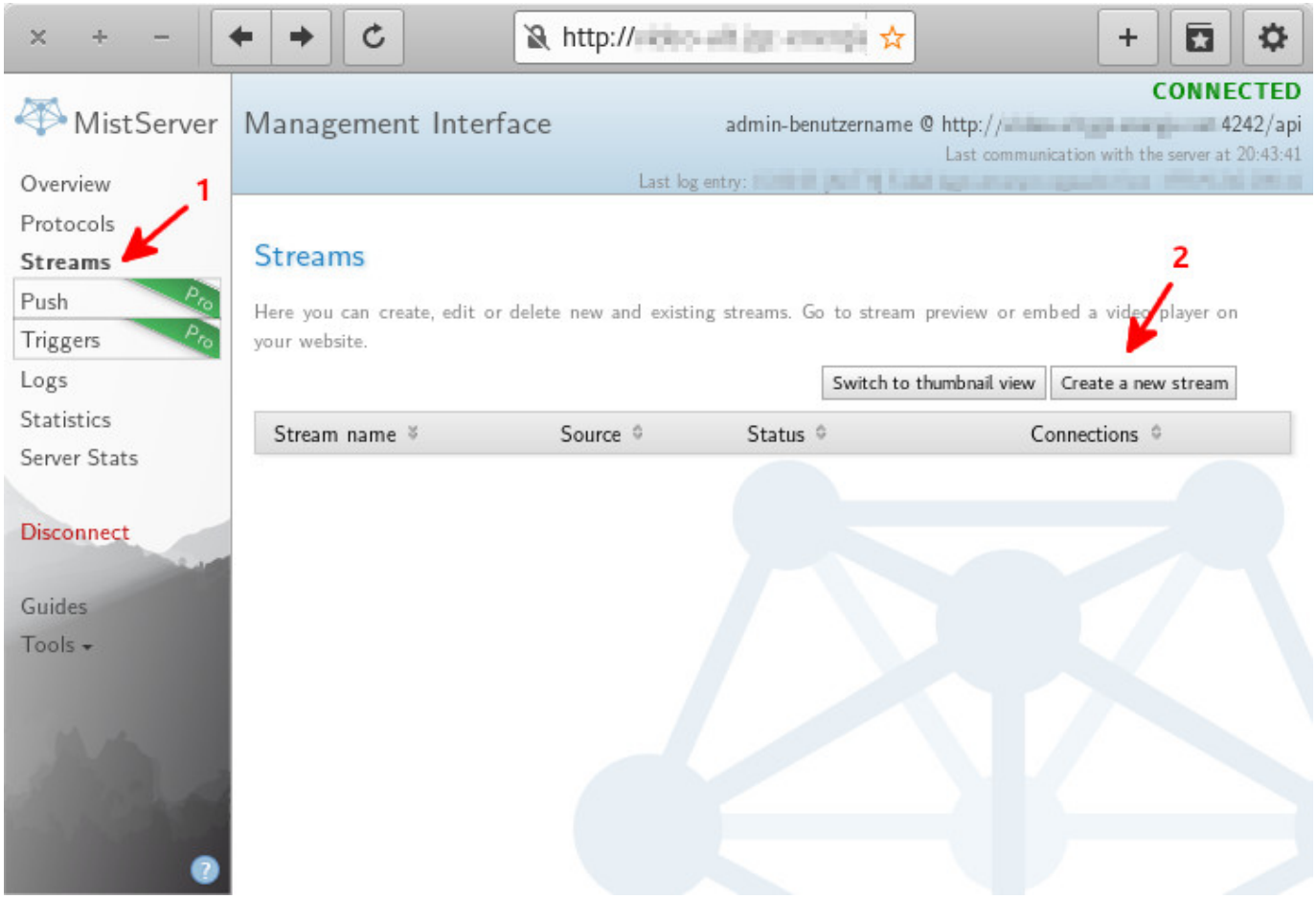


Soll der Stream über eine verschlüsselte Verbindung abgerufen werden, muss noch das https-Protokoll manuell aktiviert werden.


http://
MistServer
Management Interface
CONNECTED
admin-benutzername @ http://:4242/api
Last communication with the server at 20:41:11
Last log entry: 15:06:05
Overview
Protocols
Streams
Push
Triggers
Logs
Statistics
Server Stats
Disconnect
Guides
Tools
1
Protocols
You can find an overview of all the protocols and their relevant information here. You can add, edit or delete protocols.
Enable default protocols
Delete all protocols
2
New protocol
Protocol
Status
Settings
WebM/MKV over HTTP Enabled username: root Edit Delete
Flash progressive over HTTP (FLV) Enabled username: root Edit Delete
Flash segmented over HTTP (HDS) Enabled username: root Edit Delete
Apple segmented over HTTP (HLS) Enabled username: root Edit Delete
Microsoft segmented over HTTP (HSS) Enabled username: root Edit Delete
HTTP Active interface: 0.0.0.0, port: 8080, username: root Edit Delete
TS over HTTP Enabled username: root Edit Delete
JSON over HTTP Enabled username: root Edit Delete
MP3 over HTTP Enabled username: root Edit Delete
MP4 over HTTP Enabled username: root Edit Delete
OGG over HTTP Enabled username: root Edit Delete
RTMP Active interface: 0.0.0.0, port: 1935, username: root Edit Delete
SubRip/WebVTT over HTTP Enabled username: root Edit Delete



Jetzt kann der Stream angelegt werden.



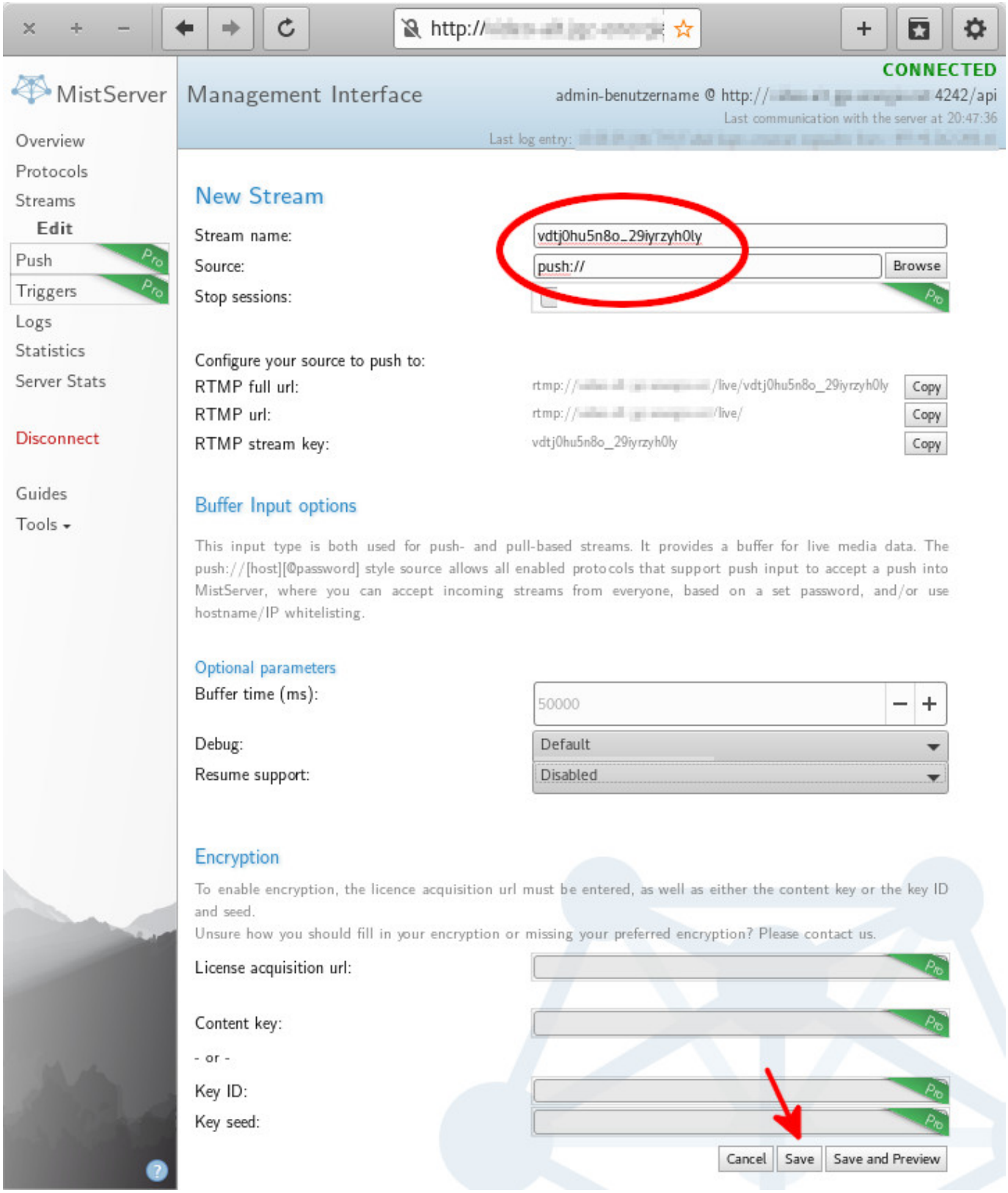
Für die Einrichtung des Live-Streams sind im MistServer nur zwei Einträge notwendig: der Stream-Name und die Quelle. Da es in der OpenSource-Version keine Möglichkeit gibt, den Stream mit einem Passwort zu schützen, hat man nur die Möglichkeit, den Stream-Namen so zu wählen, dass er nicht zu erraten ist. Mit dem folgenden Befehl kann man auf der Linux-Konsole eine zufällige Zeichenfolge generieren, die dann als Stream-Name verwendet wird:

```
< /dev/urandom tr -dc _a-z-0-9 | head -c${1:-23};echo;
```

Achtung

Im Stream-Namen dürfen keine Großbuchstaben vorkommen.

In diesem Fall wird eine 23-stellige Zeichenfolge gebildet. Die Adresse für den Live-Stream wäre dann `https://video.domain.tld/vdtj0hu5n8o_29iyrzyh0ly.html` und damit nahezu unmöglich für nicht autorisierte Zuschauer erreichbar. Natürlich muss diese Adresse dann per E-Mail, Messenger,... verschickt werden.



In einigen Fällen wurde uns von Unterbrechungen im Live-Stream berichtet, wenn über die gesicherte https-Verbindung zugeschaut wurde. Wir haben daher vorerst Port 8080 in der Firewall offen gelassen, damit auch noch über den unverschlüsselten Link `http://video.domain.tld:8080/vdtj0hu5n8o_29iyrzyh0ly.html` zugeschaut werden kann.

^ *zum Anfang*

1.9 OBS-Studio

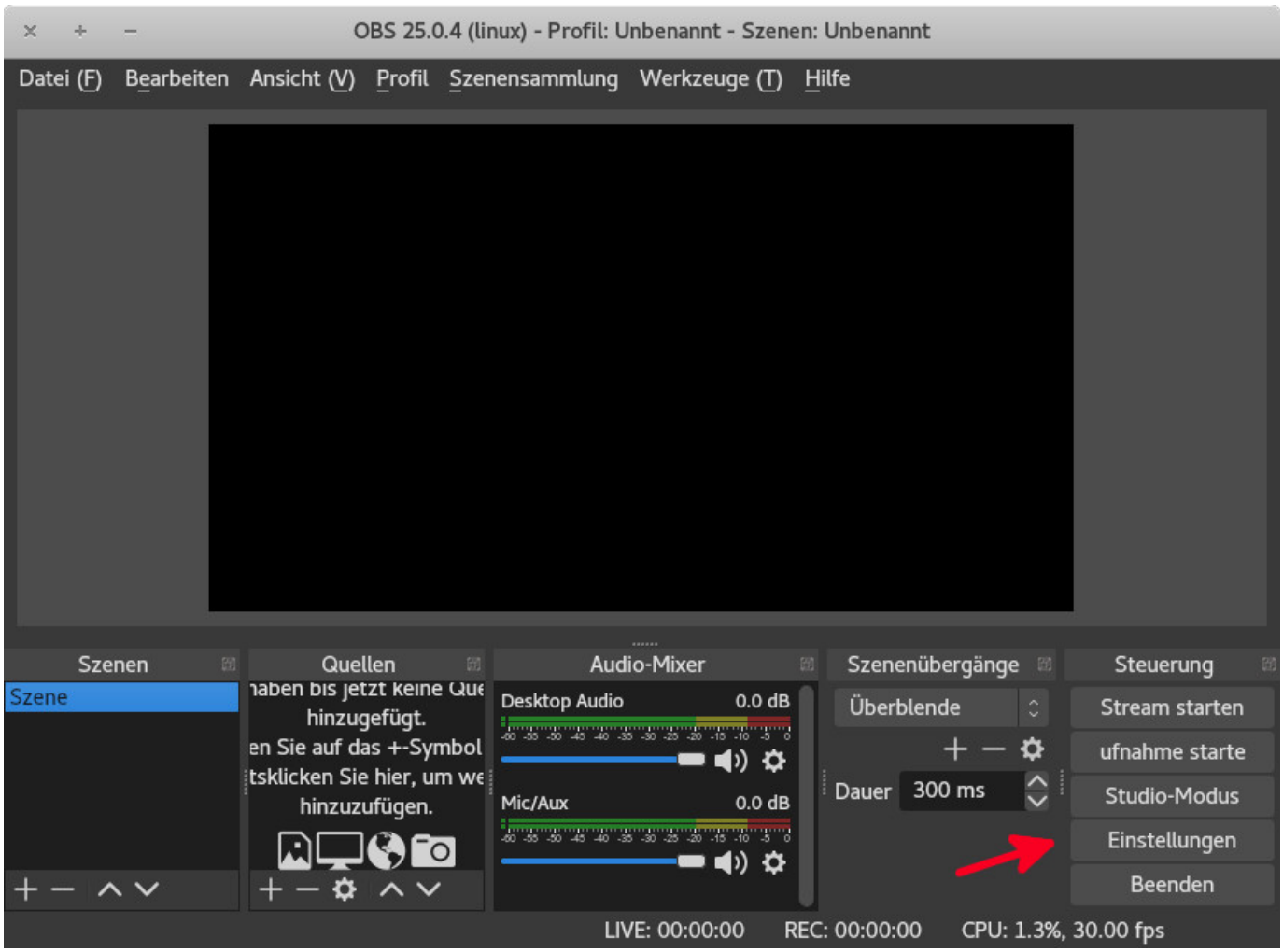
1.9.1 Installieren und einrichten

Das OBS-Studio ist eine ziemlich geniale OpenSource Software, mit der man sehr leicht Multimedia-Inhalte über das Internet streamen kann. Hier wird nur ein kleiner Ausschnitt der Funktionalität beschrieben. Unter Ubuntu kann man das Programm mit folgenden Befehlen installieren (für andere Betriebssysteme ist das unter https://obsproject.com/wiki/install-instructions beschrieben):

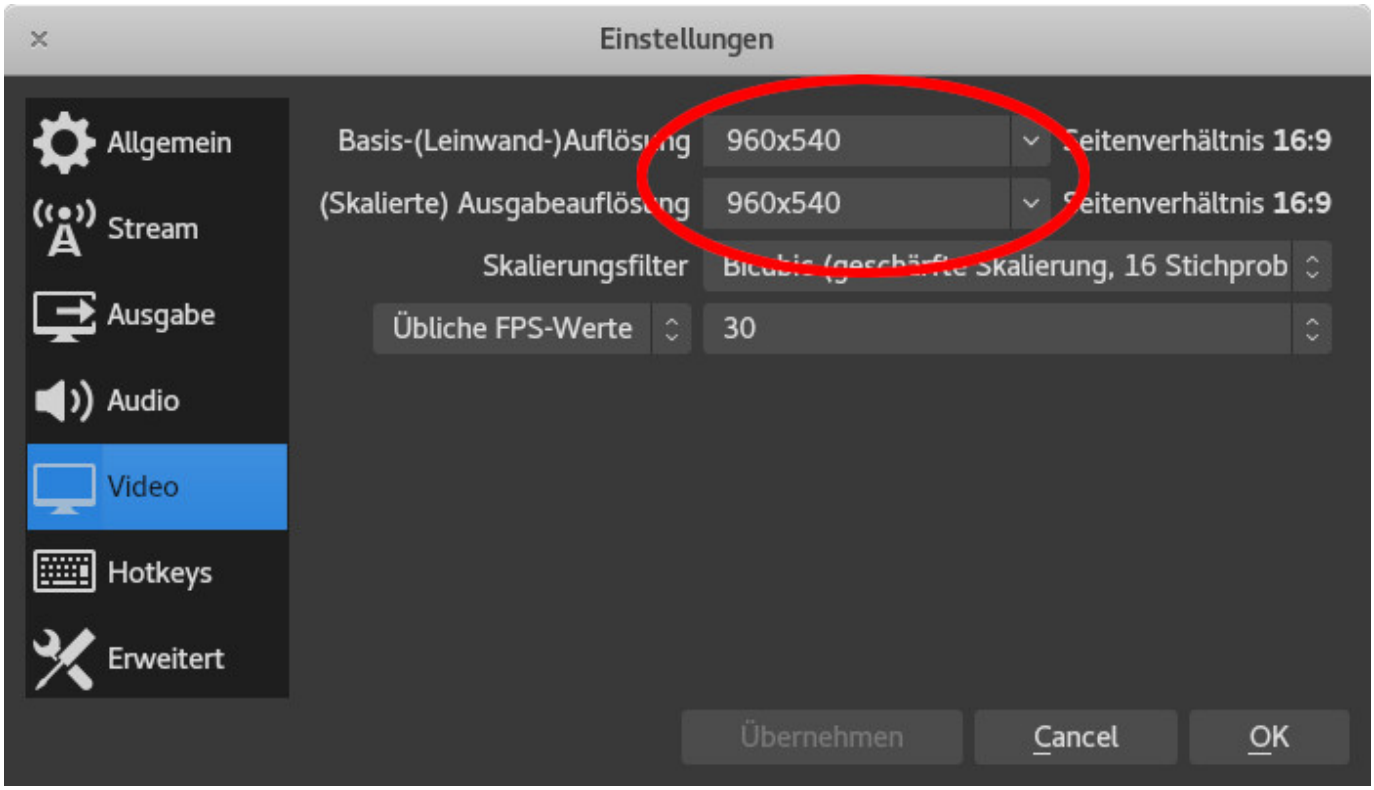
```
sudo apt-get install ffmpeg

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

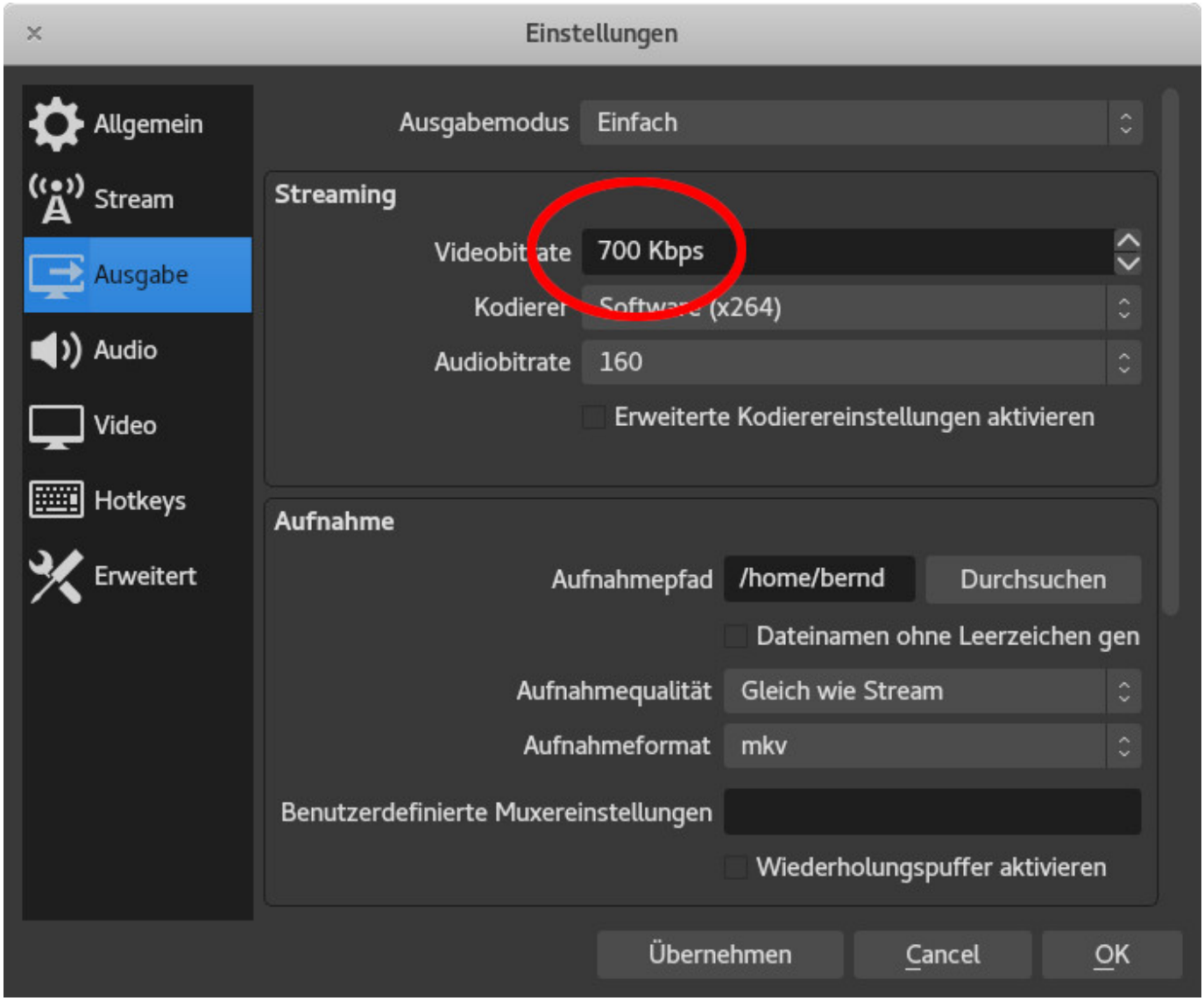
Nach erfolgreicher Installation sind in den Einstellungen des Programms zunächst verschiedene Werte anzupassen.



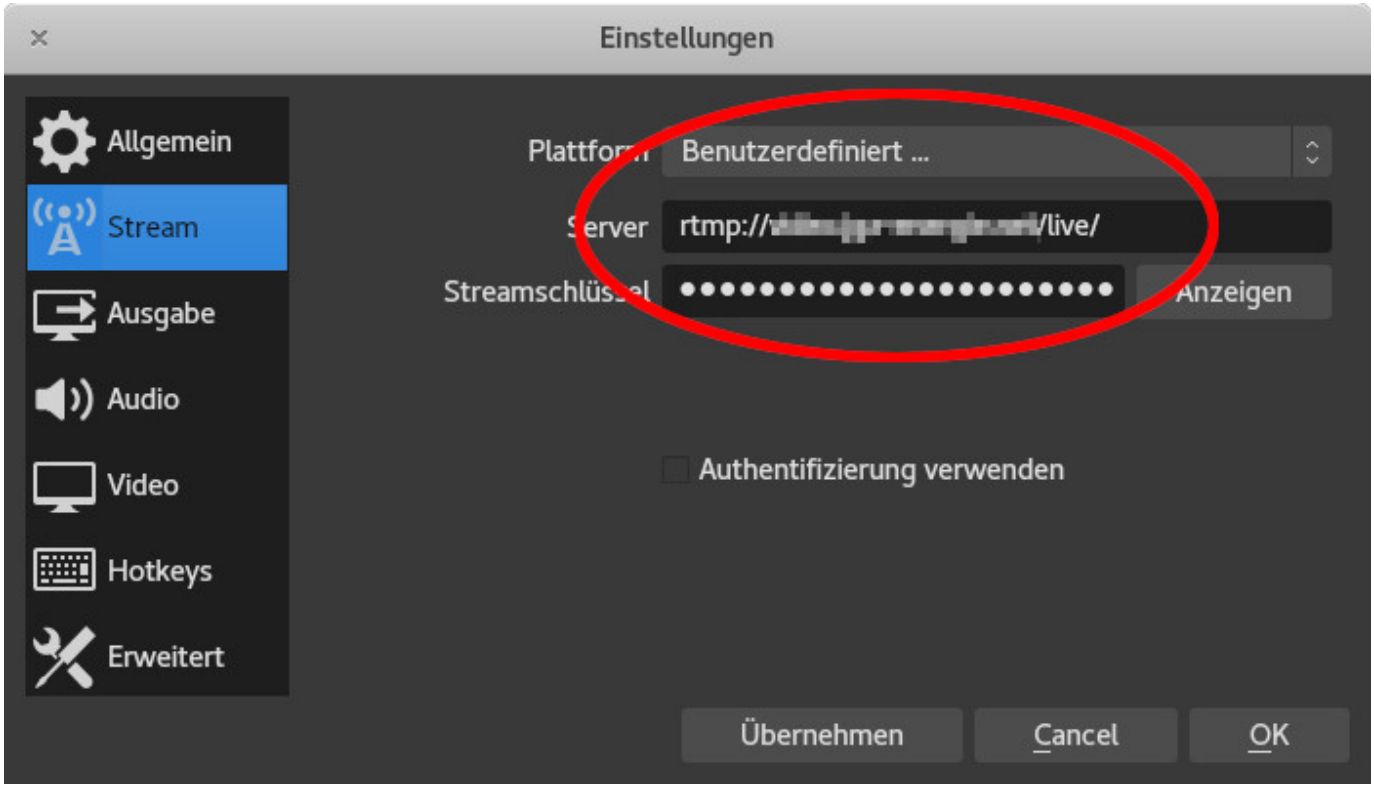
Im Bereich "Video" wird die Ausgabeauflösung eingestellt. Die Basisauflösung im OBS-Studio kann eine andere Auflösung haben, dann wird die Ausgabe skaliert. Wir haben in beiden Fällen die Auflösung des Eingangs-Signals (960x540) gewählt. Das ist für uns ein guter Kompromiss zwischen Bildqualität und Datenmenge.



Bei der "Ausgabe" haben wir nur die Bitrate angepasst. Hier ist zu beachten, dass neben der Upload-Kapazität des Internetanschlusses, von dem aus gestreamt wird, auch die Server-Kapazität eine Rolle spielt. In dieser MistServer-Dokumetation gibt es zwei Formularen, mit deren Hilfe man entweder die nötige Server-Kapazität anhand der gewünschten Streaming-Parameter (Anzahl Streams, Bitrate, Anzahl Zuschauer) oder die mögliche Zuschauer-Anzahl anhand der Server-Parameter (CPU, RAM, HDD, Bandbreite) ermitteln kann. Viele Provider drosseln die Verbindungsgeschwindigkeit, wenn über einen gewissen Zeitraum zu viele Daten übertragen werden. Daher sollte man bei der Bitrate nicht zu hoch gehen. Aus unserer Erfahrung sind 700KBit/s ein guter Wert für einen günstigen root-Server mit bis zu 50 Zuschauern (bzw. angeschlossenen Geräten).

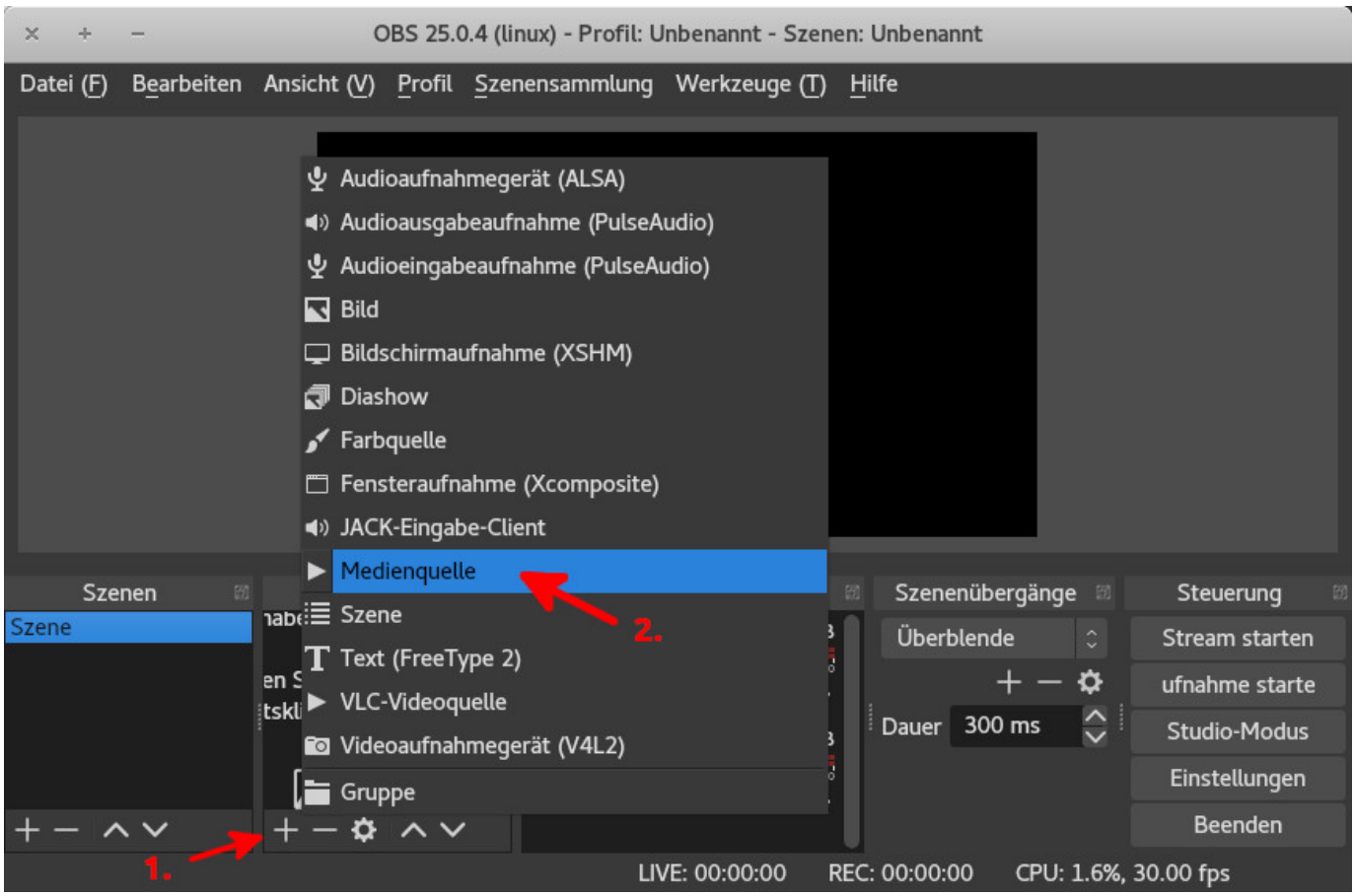


Bei den "Stream" Einstellungen wird zunächst als Plattform "Benutzerdefiniert..." ausgewählt. Danach müssen die Daten aus den MistServer Stream-Einstellungen eingefügt werden. Server wäre in dem Fall `rtmp://video.domain.tld/live` und als Key wird der Stream-Name eingetragen `vdtj0hu5n8o_29iyrzyh0ly`.

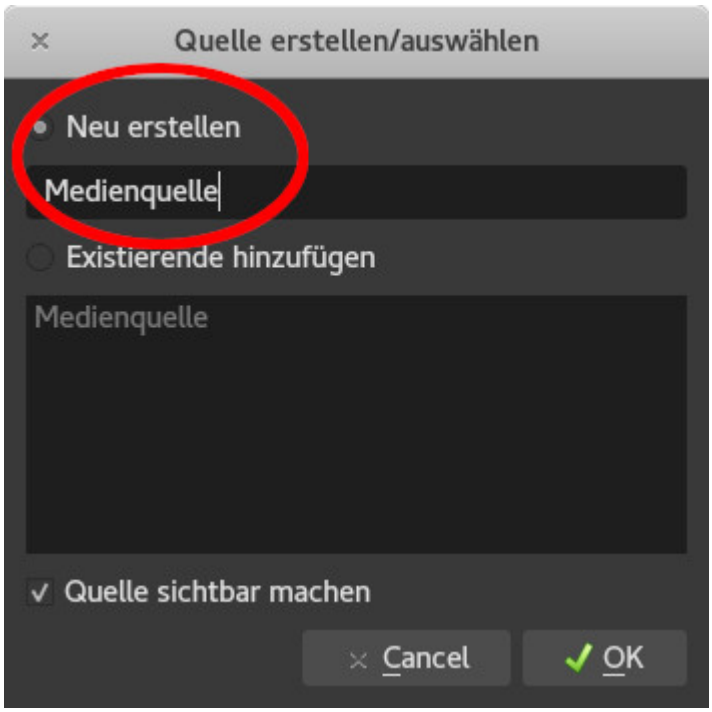


1.9.2 Szenen und Quellen festlegen

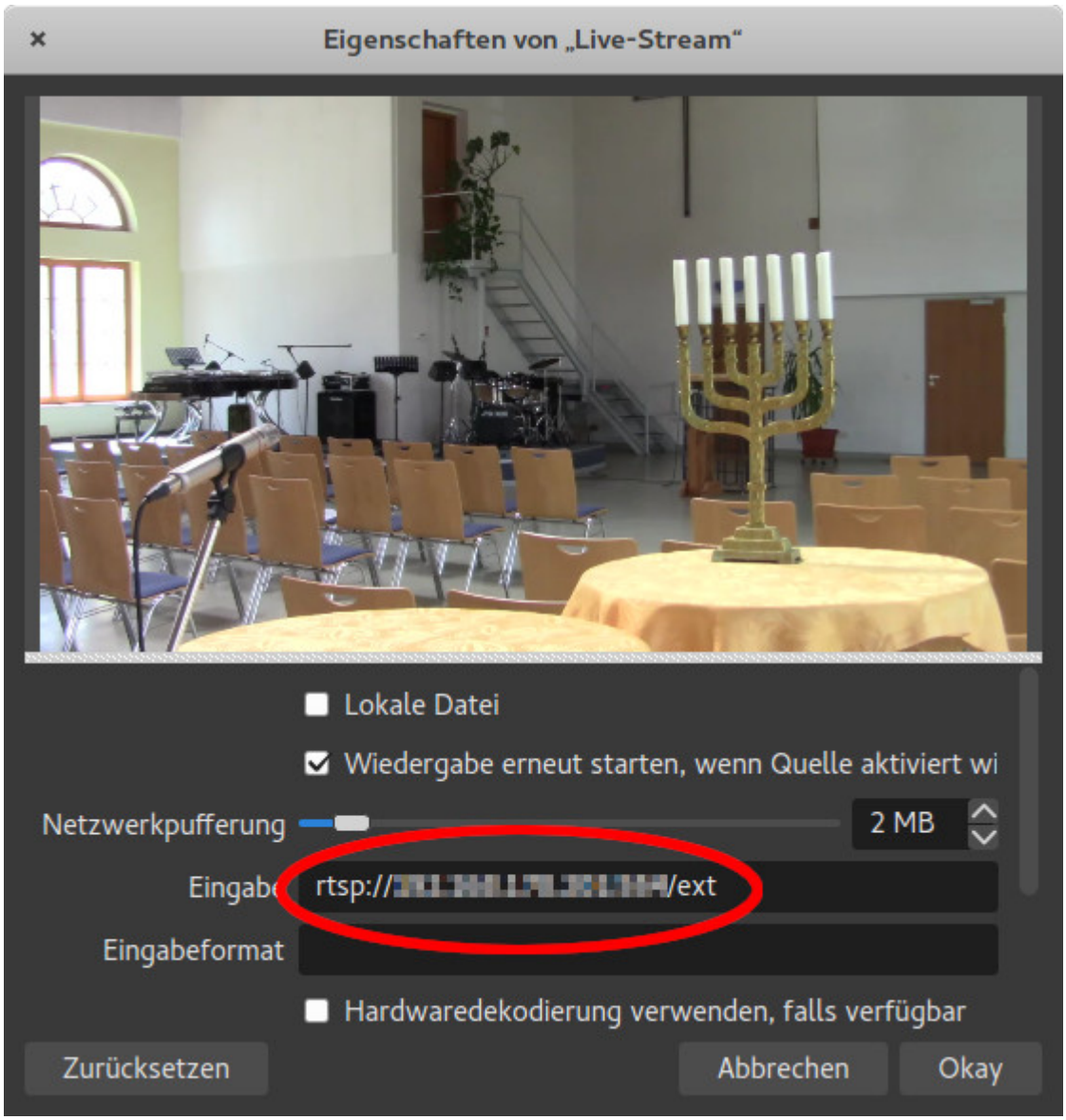
Im Hauptfenster des OBS-Studios wird jetzt eine neue Medienquelle hinzugefügt:



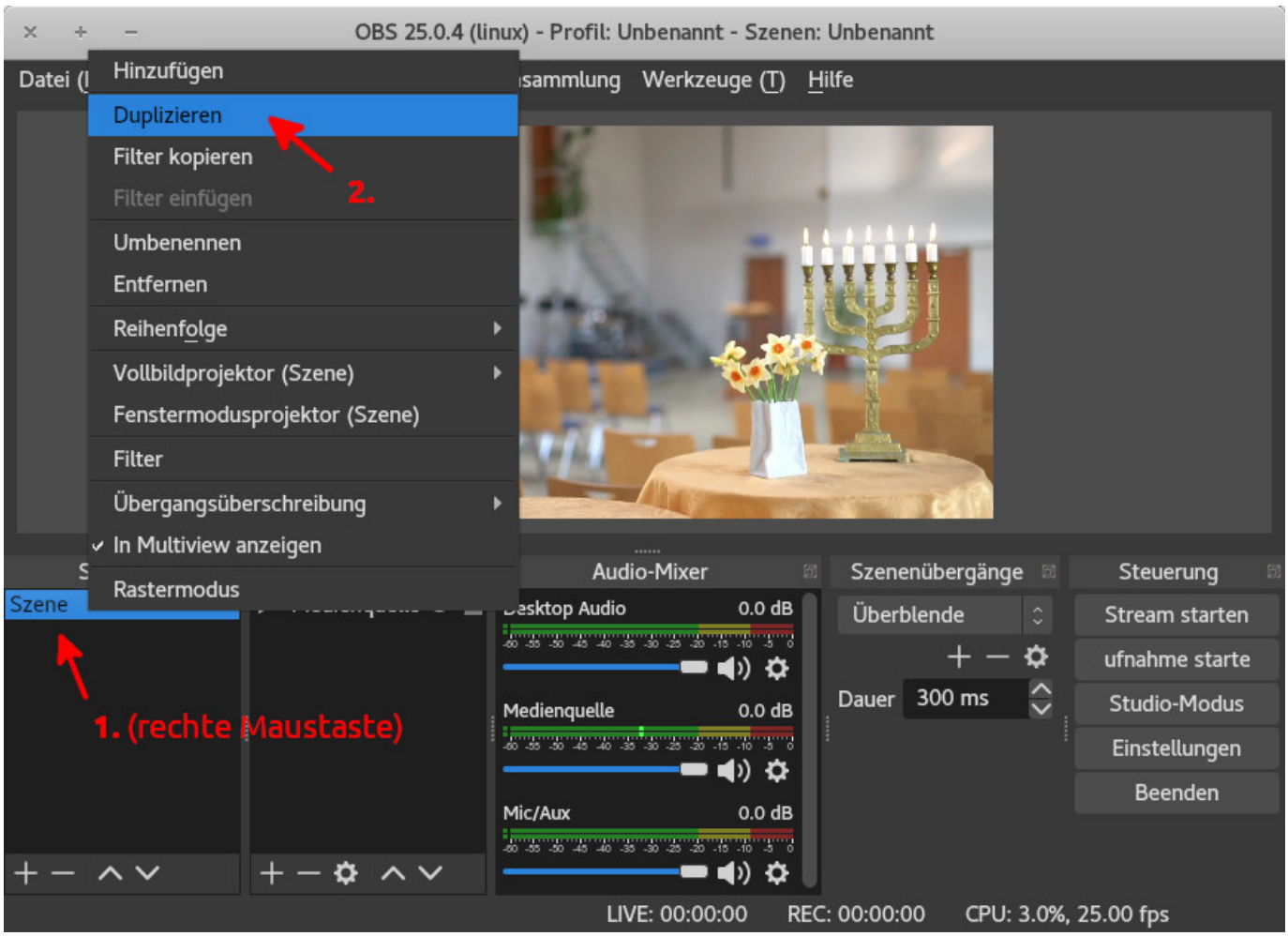
Anschließend kann man auswählen, ob man eine neue Medienquelle hinzufügen möchte, oder ob man eine vorhandene Quelle (aus einer anderen Szene) einfügen möchte.



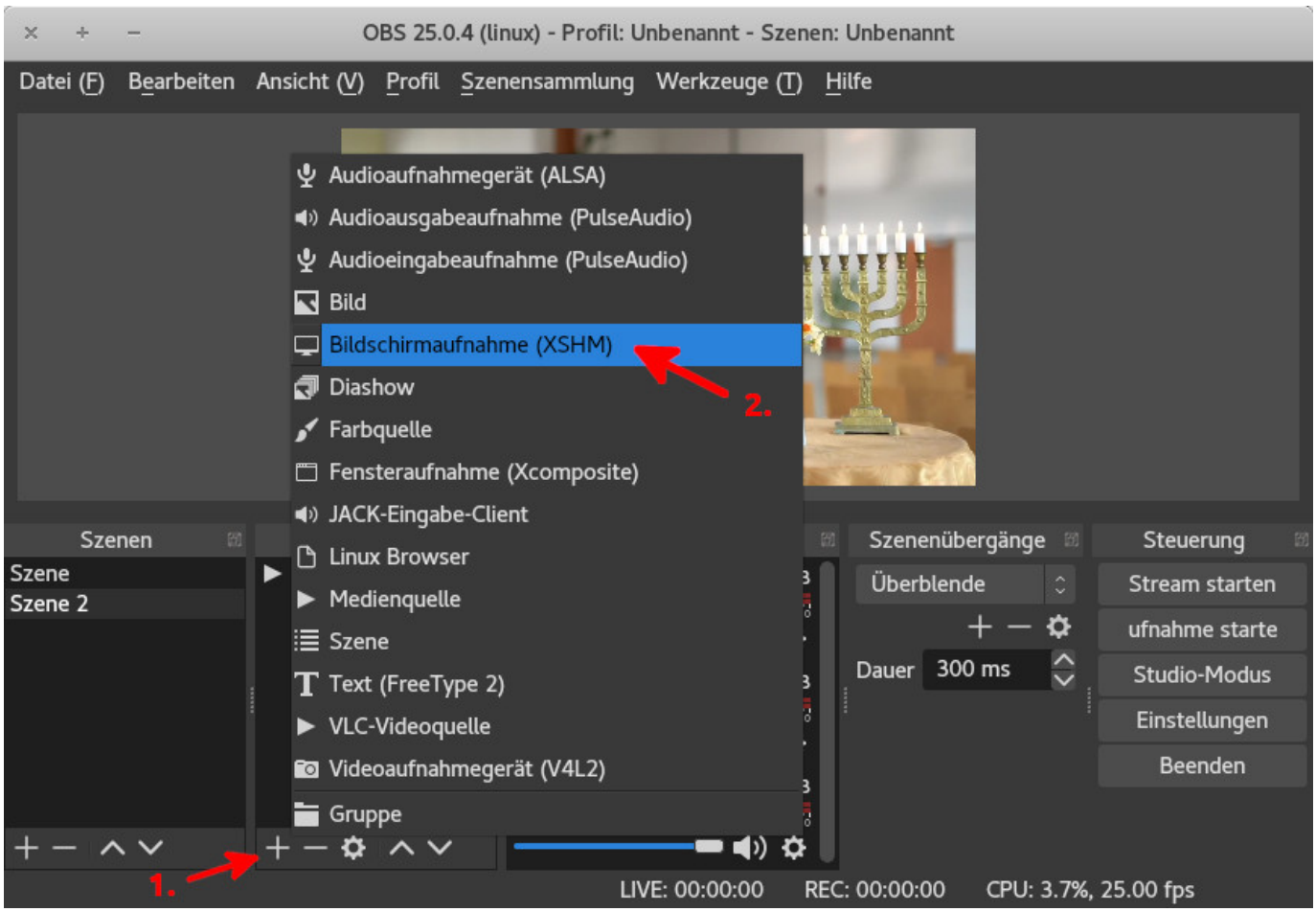
Jetzt muss noch im Feld "Eingabe" die Adresse des Encoder-Streams eintragen werden. Danach sollte das Live-Bild zu sehen sein.



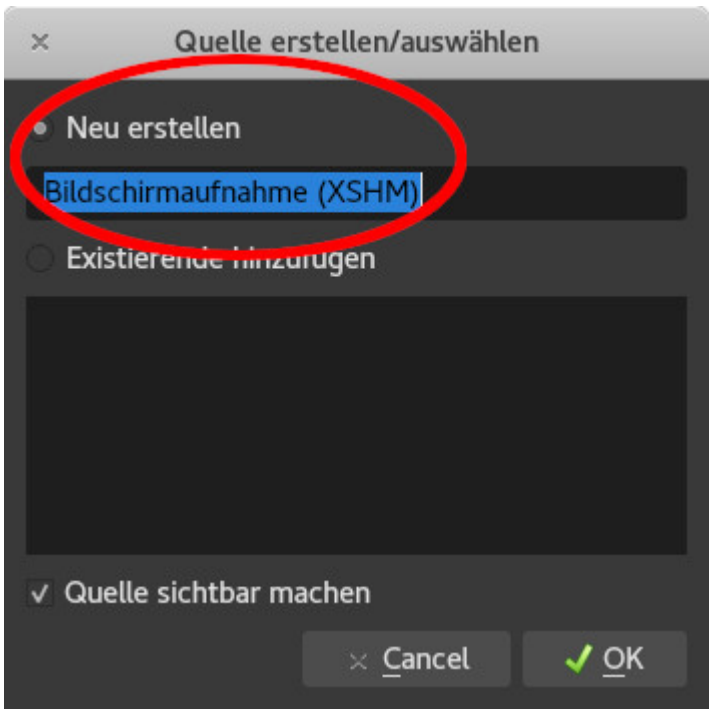
Damit der Ablauf eines Live-Gottesdienstes für die Techniker besser zu kontrollieren ist, haben wir verschiedene Szenen angelegt. Wenn man mit der rechten Maustaste auf eine "Szene" klickt, und dann auf "Duplizieren", hat man eine neue Szene mit dem Live-Video als Ausgangspunkt.



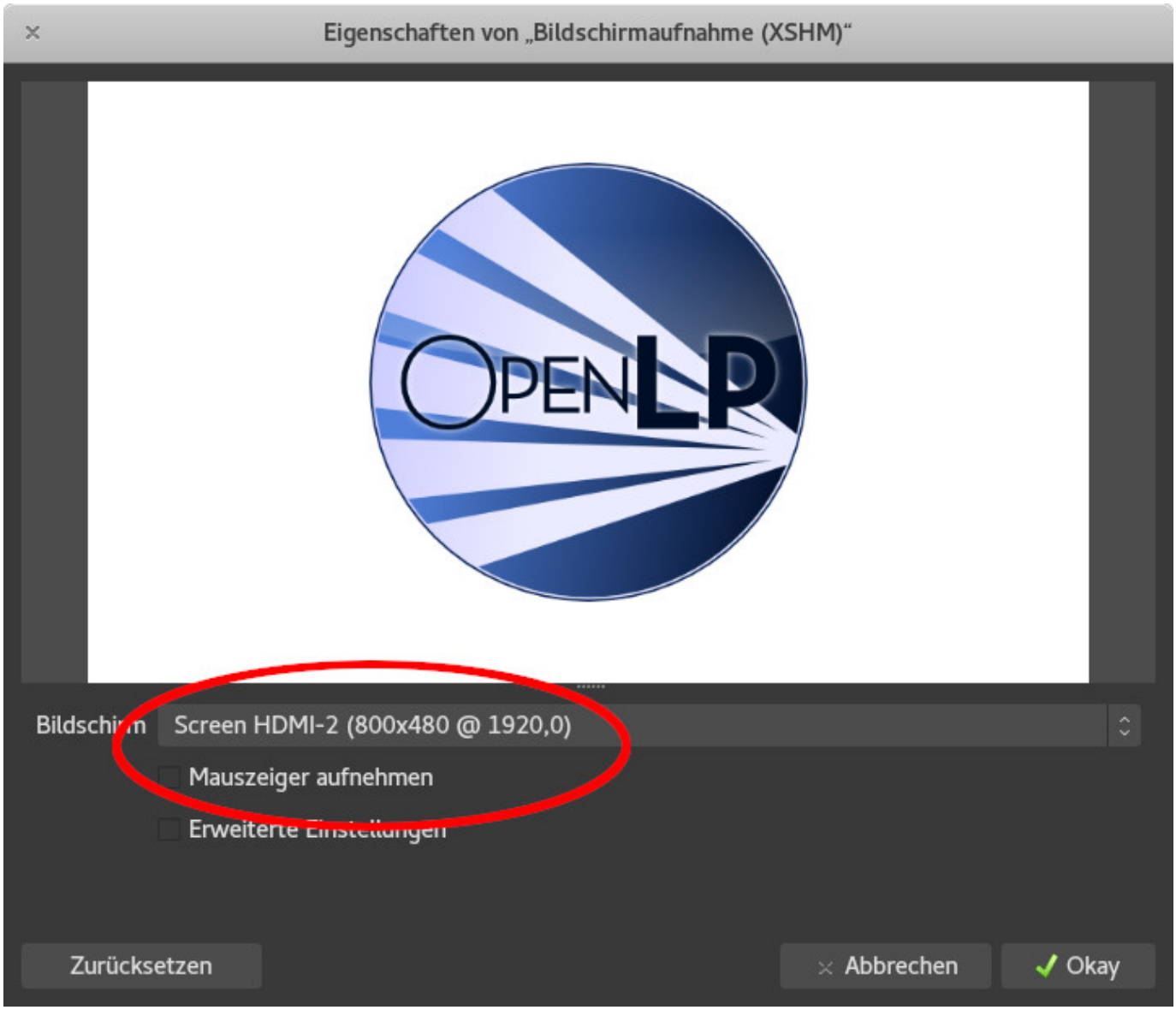
In der neuen Szene wird jetzt eine neue Quelle hinzugefügt. Wir verwenden z.B. noch die Bildschirmaufnahme vom zweiten Monitor (Beamer), um die Ausgabe von OpenLP mit in den Live-Stream zu integrieren.



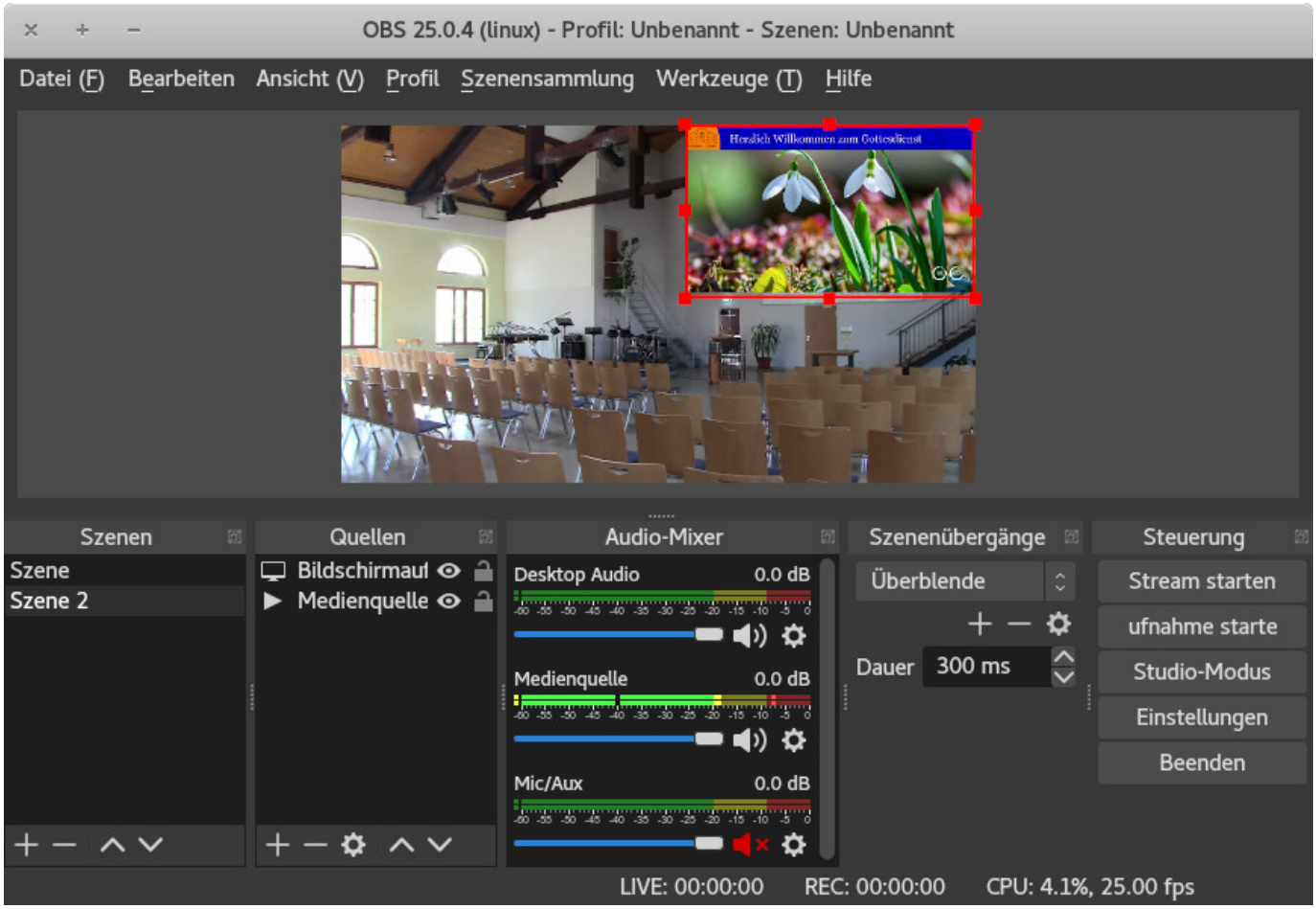
Im folgenden Eingabefenster wird wieder ausgewählt, ob es sich um eine neue Quelle handelt, oder ob sie aus einer anderen Szene übernommen werden soll.



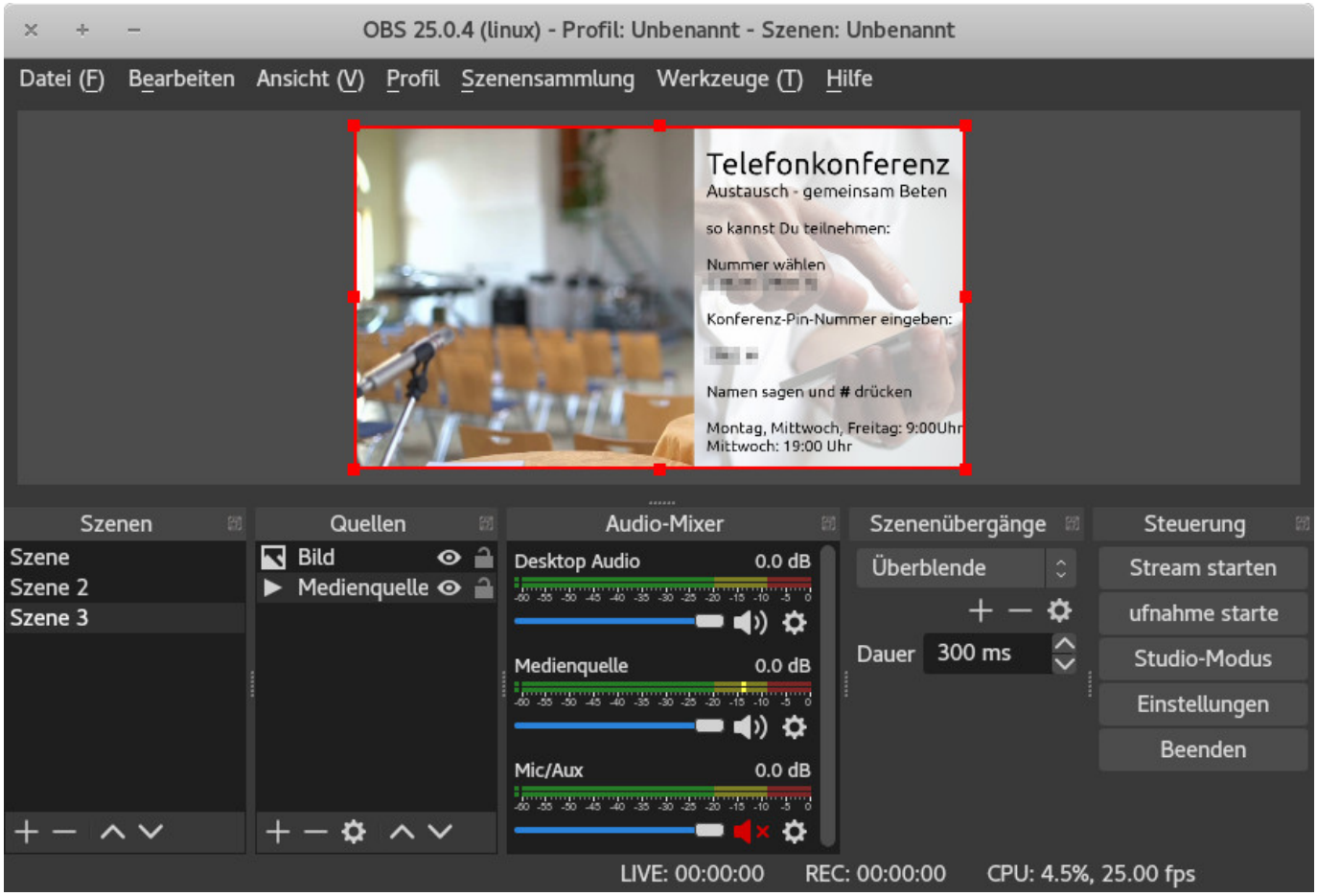
Wenn ein zweiter Bildschirm angeschlossen ist, kann man diesen in dem nächsten Eingabefenster auswählen und bekommt eine Vorschau angezeigt. Den Mauszeiger benötigen wir in unserem Stream nicht und deaktivieren ihn daher.



Nun kann die Bildschirmaufnahme auf der Anzeige verschoben, bzw. vergrößert oder verkleinert werden, damit sie gut in das Gesamtbild passt (hier nur als Beispiel).



Eine weitere Szene nutzen wir bei Gottesdiensten, die ausschließlich online übertragen werden (z.B. während der Corona-Krise 2020), um Informationen direkt neben dem Sprecher zu platzieren. Die Vorgehensweise ist dabei in etwa so, wie bereits beschriebenen. Zunächst wird eine neuen Szene angelegt, die wieder aus der ersten Szene dupliziert wird, damit das Live-Video als Hintergrund da ist. Dann kommt eine neue Quelle vom Typ "Bild" hinzu, die auf eine Datei mit dem entsprechendem Inhalt verweist. Die Bild-Datei sollte vom Format so erstellt werden, dass sie auf eine Hälfte des Live-Streams passt.



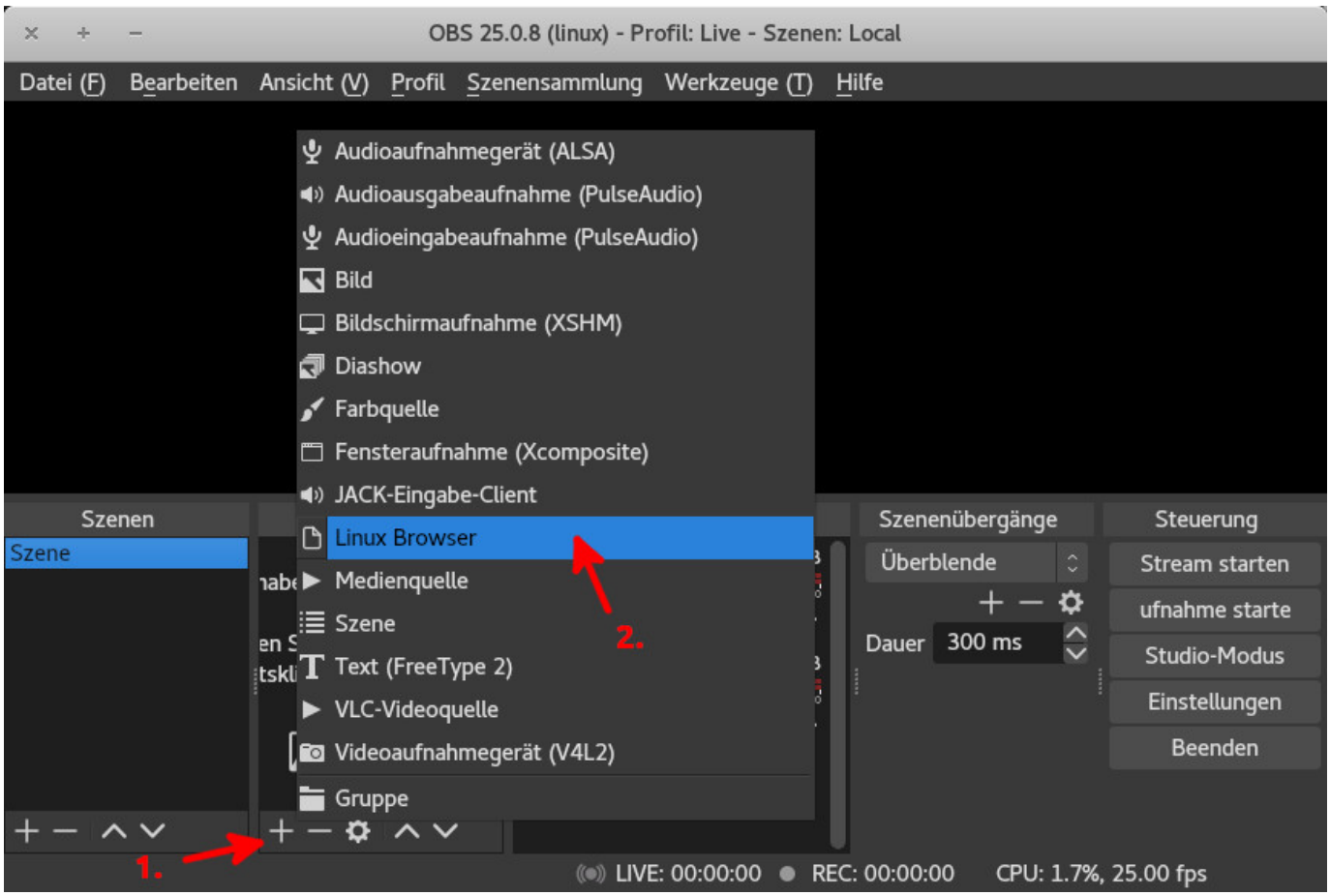
Jetzt kann während des streamens ganz einfach durch einen Mausklick auf die entsprechende Szene umgeschaltet werden. Wenn man z.B. zwischen den Szenen 1 und 3 wechselt, wird das Info-Bild sanft ein- oder ausgeblendet, ohne dass dabei der Kamera-Stream unterbrochen wird.

1.9.3 Live-Chat mit Jitsi Meet

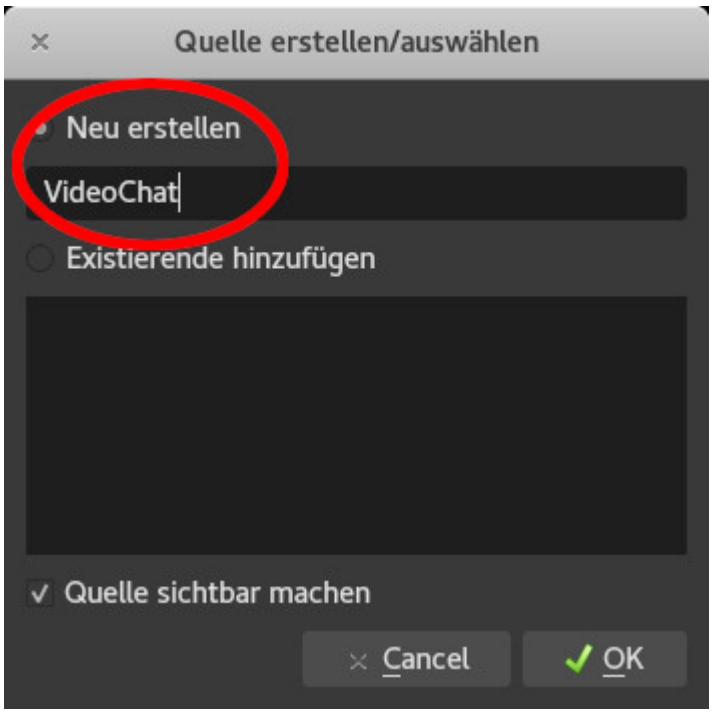
OBS-Studio kann Internetseiten als Quelle nutzen und in den Stream integrieren. Wir haben diese Möglichkeit genutzt, um während eines Gottesdienstes eine Video-Chat Verbindung über Jitsi Meet herzustellen.

Unter Linux ist es unter Umständen (je nach OBS Version) nötig, noch das Plug-in Linux-Browser zu installieren.

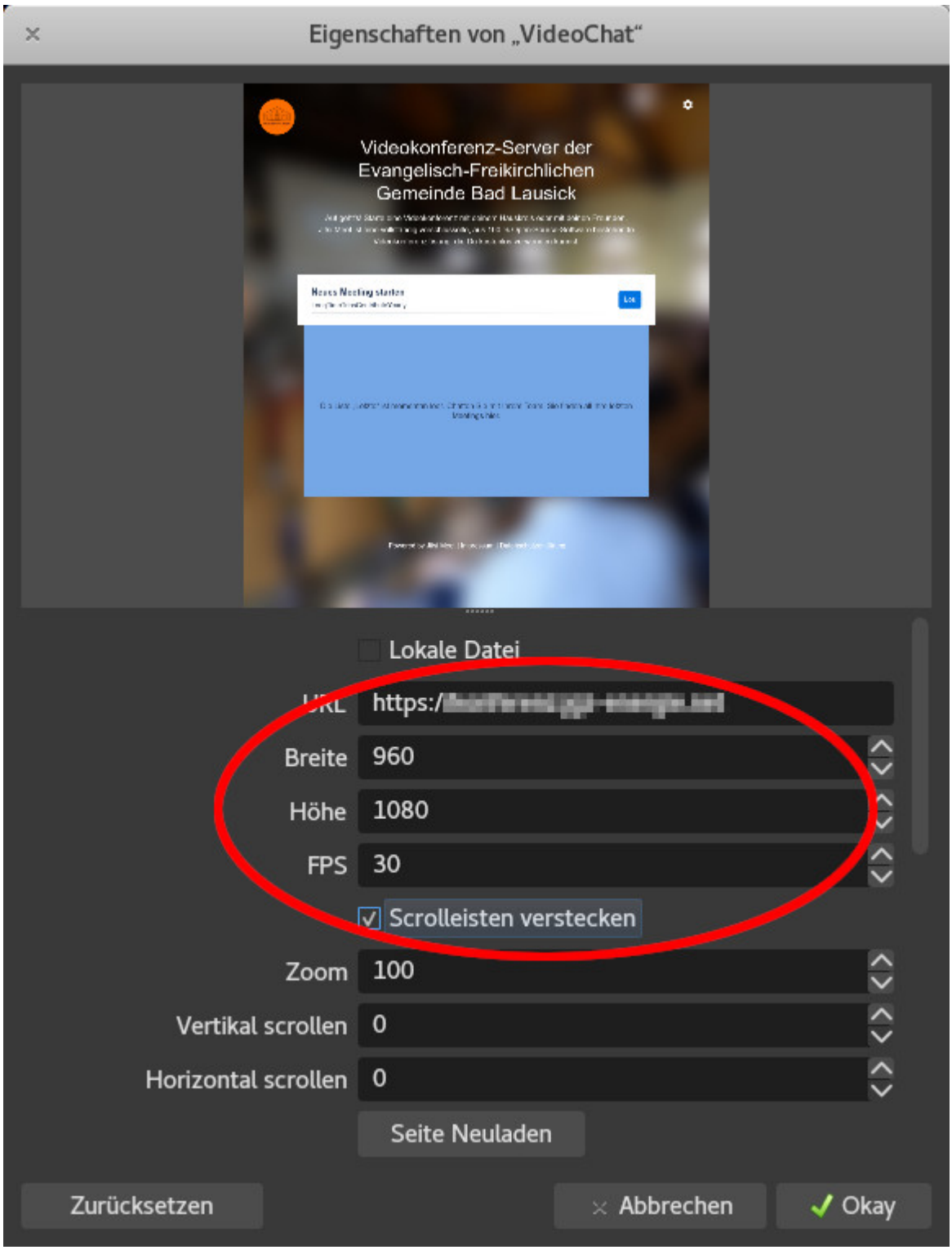
Als Quelle fügt man `Browser` bzw. `Linux Browser` hinzu:



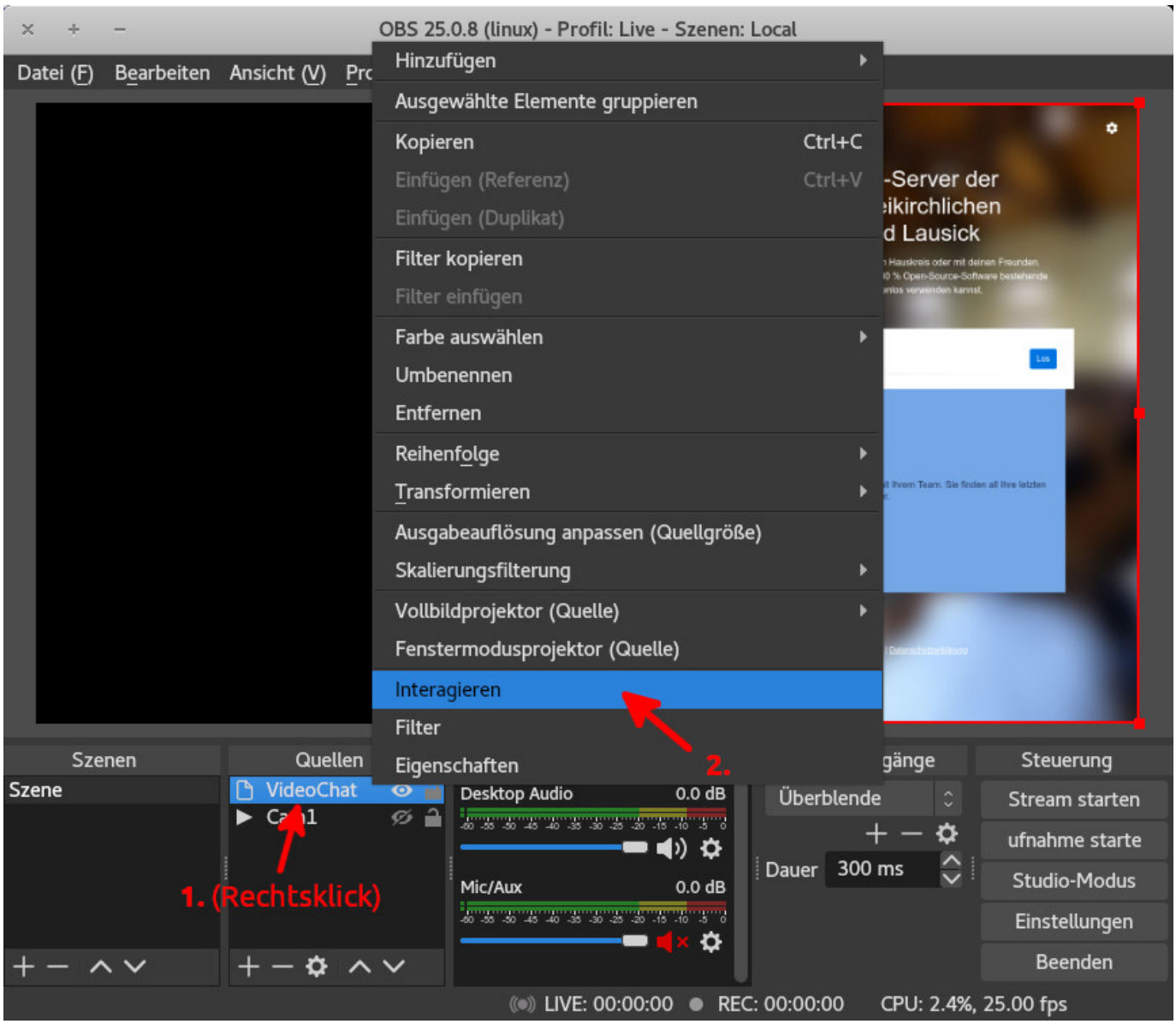
...legt den Namen fest:



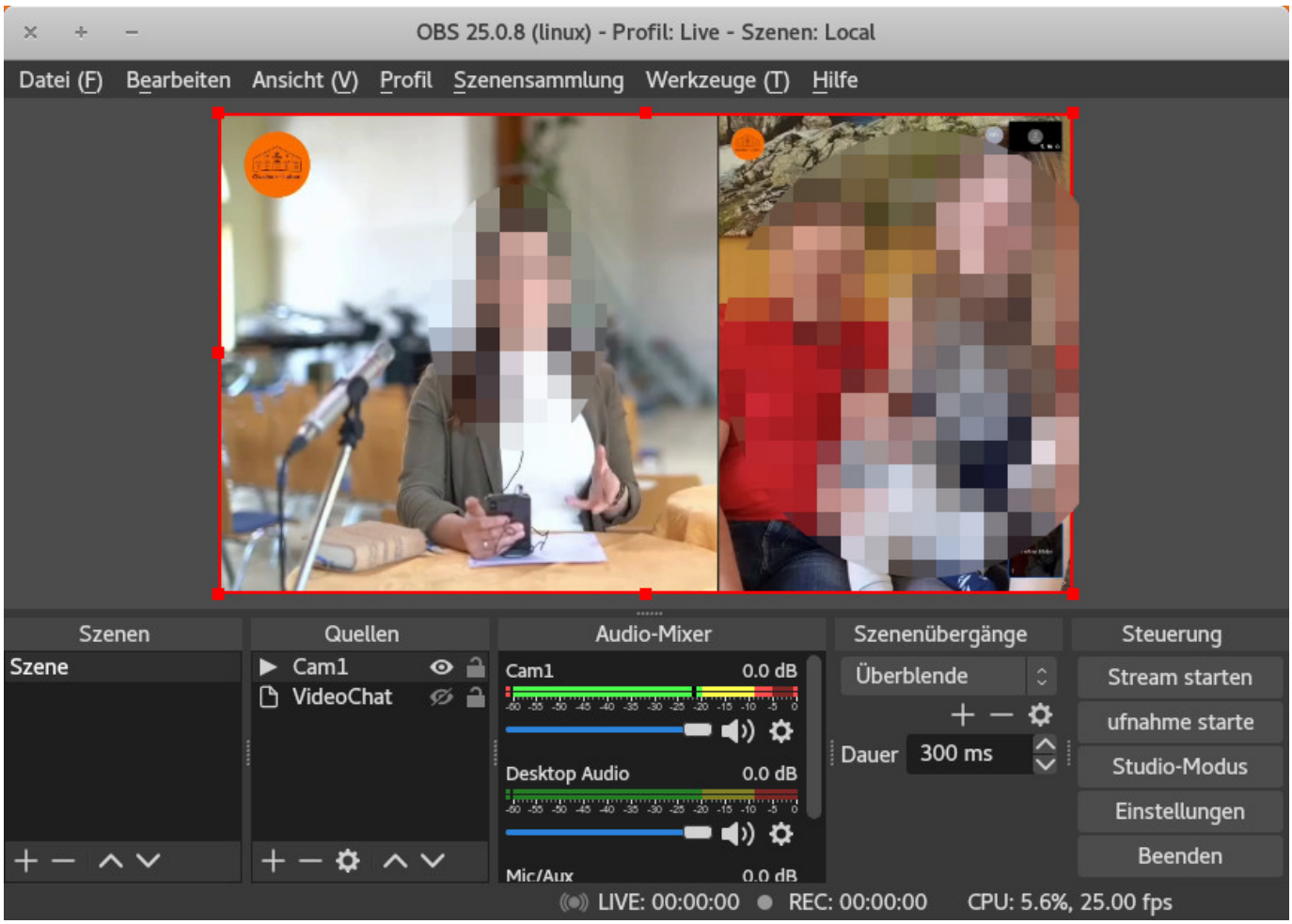
...und trägt die entsprechenden Daten ein:



Mit Rechtsklick auf die Quelle `VideoChat` und dann Klick auf `Interagieren`, kann man die Seite mit Tastatur und Maus bedienen und sich so an der Videokonferenz anmelden.



Der OBS-Browser kann nicht auf Kamera und Mikrofon zugreifen und ist somit nur Zuschauer. Die Moderatorin hat sich in unserem Fall mit Smartphone und Headset in die Jitsi Meet Konferenz eingewählt.



Achtung

Während der Videokonferenz muss der Ton des Kamera-Streams stumm geschaltet werden, da es eine zeitliche Verzögerung zum Ton von Jitsi Meet gibt und der Ton des Moderators dann doppelt (mit Echo) auf dem Live-Stream wäre. Der Moderator ist also während der Videokonferenz nur über Jitsi Meet zu hören.

Die Installation eines eigenen Jitsi Meet Servers haben wir in einer anderen Dokumentation beschrieben.

^ *zum Anfang*

Erstellt: 04-2020 von Bernd Reuther, Evangelisch-Freikirchliche Gemeinde Bad Lausick.

1.10 Kommentare, Fragen, Anregungen