abdulive-toolkit

Handbuch

Aus einem Zuruf im Chat wird ein fertiges Hochformat-Video.

Version	0.5.4
Stand	9. September 2026
Gilt für	Windows 10 und 11, 64 Bit
Quelltext	`https://git.zeasy.dev/ZeasyAlex/abdulive`

Das Toolkit speichert Chatverläufe. Die Datenbank enthält damit personenbezogene Daten und gehört weder ins Repository noch in eine Cloud ohne Absprache.

Inhaltsverzeichnis

1 Was das Werkzeug tut **3**

1.1 Was dabei herauskommt . . . 3

1.2 Was es noch nicht tut . . . 3

2 Installation **3**

2.1 Wohin es installiert . . . 4

3 Einrichtung **4**

3.1 Schritt 1: Zwei Twitch-Anwendungen anlegen . . . 4

3.2 Schritt 2: Broadcaster-Konto verbinden . . . 4

3.3 Schritt 3: Bot-Konto verbinden . . . 4

3.4 Schritt 4: OBS-Passwort . . . 5

3.5 Fertig? . . . 5

4 OBS einrichten **5**

4.1 Wiedergabepuffer . . . 5

4.2 Getrennte Tonspuren . . . 5

4.3 Bildaufteilung . . . 5

4.4 Fremder Text im Bild . . . 6

5 Der laufende Betrieb **6**

5.1 Clips auslösen . . . 6

5.2 Befehle . . . 6

5.3 Läuft es? . . . 6

6 Updates **7**

7 Sicherung **7**

8 Deinstallation **7**

9 Wenn etwas nicht geht **7**

10 Wo was liegt **8**

1 Was das Werkzeug tut

Jemand tippt `!clip` in den Chat. Etwa eine Minute später liegt ein fertiges 9:16-Video mit Untertiteln auf der Platte, bereit für TikTok, Reels oder Shorts.

Dazwischen passiert Folgendes:

1. Das Toolkit liest den Chat mit und erkennt den Befehl.
2. Es weist OBS an, den Wiedergabepuffer zu speichern. OBS hat die letzten 90 Sekunden ohnehin im Speicher, der Moment ist also schon vorbei und trotzdem noch da.
3. OBS meldet den Dateipfad zurück. Das Toolkit wartet, bis die Datei fertig geschrieben ist.
4. Aus den letzten 45 Sekunden wird ein Hochformat gebaut: Facecam oben, Gameplay unten, beides ohne Verzerrung zugeschnitten.
5. Die Mikrofonspur wird transkribiert und als Untertitel ins Bild gebrannt.

> **Während des Streams wird nicht gerendert.** Das Encodieren nimmt Rechenzeit, die der Stream braucht. Die Warteschlange arbeitet erst nach Streamende. Wer das anders will, setzt `renderWhileLive: true`.

1.1 Was dabei herauskommt

Je Clip entsteht ein Ordner mit:

`final.mp4`	das fertige Video mit eingebrannten Untertiteln
`vertical.mp4`	dasselbe ohne Untertitel
`thumb.jpg`	Vorschaubild
`subs.srt`	Untertitel als Textdatei, praktisch für die Beschreibung
`transcript.txt`	reiner Text des Gesagten

1.2 Was es noch nicht tut

Es lädt nichts hoch. Es entscheidet nicht, welcher Clip gut ist. Die Sichtungsseite, auf der man Clips durchgeht und aussortiert, ist noch nicht gebaut.

2 Installation

Vorausgesetzt wird Windows 10 oder 11 in 64 Bit und OBS Studio 30 oder neuer. Sonst nichts. Node, ffmpeg, whisper.cpp und die Schriften sind im Paket enthalten.

In PowerShell:

```
irm https://git.zeasy.dev/ZeasyAlex/abdulive/raw/branch/main/scripts/install.ps1 | iex
```

Das Skript lädt das Paket, prüft die Prüfsumme, installiert, holt das Sprachmodell (etwa 465 MB), startet den Dienst und öffnet die Einrichtungsseite im Browser.

Es braucht keine Administratorrechte. Beim ersten Start warnt Windows SmartScreen vor einem unbekannten Herausgeber, weil das Paket nicht signiert ist. Einmal auf *Weitere Informationen*, dann *Trotzdem ausführen*.

2.1 Wohin es installiert

Programm	`%LOCALAPPDATA%\Programs\abdulive`
Daten	`%LOCALAPPDATA%\abdulive`
Autostart	Aufgabe `abdulive`, bei jeder Anmeldung
Updates	Aufgabe `abdulive-update`, täglich 5:00 Uhr

Programm und Daten liegen getrennt. Ein Update tauscht den Programmordner komplett aus, die Datenbank bleibt davon unberührt.

3 Einrichtung

Die Einrichtungsseite liegt unter `http://localhost:4792/setup` und ist nur vom eigenen Rechner erreichbar. Sie zeigt Zugangsdaten im Klartext, also nicht während des Streams öffnen.

3.1 Schritt 1: Zwei Twitch-Anwendungen anlegen

Unter `https://dev.twitch.tv/console/apps` zwei Anwendungen erstellen. Bei beiden als OAuth-Redirect-URL exakt eintragen:

```
http://localhost:4792/callback
```

Die Anwendungen hängen nicht am Kanal. Sie können auch unter einem anderen Konto angelegt werden.

3.2 Schritt 2: Broadcaster-Konto verbinden

Client-ID und Secret der ersten Anwendung auf der Einrichtungsseite eintragen, speichern, dann auf *Mit Twitch verbinden* klicken.

> **Das kann nur der Streamer selbst.** Twitch bindet die Berechtigungen an das Konto, das sich anmeldet. Es gibt keine Rolle, mit der jemand anderes das übernehmen kann. Moderator und Redakteur helfen hier nicht.

Damit bekommt das Toolkit Zugriff auf Clips, Titel und Kategorie, Abonnenten, Follower, Raids, Kanalpunkte und Bits.

3.3 Schritt 3: Bot-Konto verbinden

Dasselbe mit der zweiten Anwendung, angemeldet als Bot-Konto. Das ist ein eigenes, kostenloses Twitch-Konto, damit Antworten im Chat als Bot erkennbar sind.

Ohne dieses Konto liest das Toolkit den Chat trotzdem mit. Es kann nur nicht antworten.

> **Das Bot-Konto sollte Moderator im Kanal sein.** Der Streamer tippt dafür einmal `/mod botname` in seinen Chat. Ohne Moderatorrechte gilt ein Limit von 20 Nachrichten je 30 Sekunden statt 100, und im Follower-only-Modus kommt der Bot gar nicht durch.

3.4 Schritt 4: OBS-Passwort

In OBS unter *Werkzeuge* → *WebSocket-Servereinstellungen* den Server aktivieren, Port 4455. Das dort angezeigte Passwort auf der Einrichtungsseite eintragen.

3.5 Fertig?

Unten auf der Seite steht `bereit` oder `unvollständig`. Solange das Broadcaster-Konto fehlt, läuft das Toolkit im Lesemodus: Chat wird mitgeschrieben, Zuschauerzahlen und Clips bleiben aus.

4 OBS einrichten

Ohne diese vier Punkte entstehen keine Clips oder sie sitzen schief.

4.1 Wiedergabepuffer

Einstellungen → *Ausgabe* → *Wiedergabepuffer* aktivieren, 90 Sekunden.

Unter *Einstellungen* → *Hotkeys* eine Taste auf *Wiedergabepuffer speichern* legen. Das ist der Clip-Knopf für den Streamer selbst. Ein eigener Windows-Hotkey wird bewusst nicht gebaut: OBS meldet den gespeicherten Puffer ohnehin über WebSocket, das ist derselbe Auslöser.

4.2 Getrennte Tonspuren

Einstellungen → *Ausgabe* → *Aufnahme* → *Audiospuren*, und im Audiomixer je Quelle die Zuordnung prüfen:

Spur 1	Mischung, kommt ins fertige Video
Spur 2	Mikrofon, daraus entstehen die Untertitel
Spur 3	Spielton

Das ist bei lauten Spielen der Unterschied zwischen brauchbarem und unbrauchbarem Transkript.

4.3 Bildaufteilung

Der Zuschnitt nimmt feste Bereiche aus dem fertigen Streambild. Verrutscht die Kamera um 50 Pixel, sitzt in jedem Short der Kopf schief. Deshalb liegt die Position fest:

Element	Position	Größe
Facecam	x 0, y 0	480 × 270
Gameplay-Band	x 515 bis 1405	890 × 1080
Chat, Alerts, Overlays	ab x 1405	515 px Spalte

Zwischen Kamera und Gameplay-Band liegen 35 Pixel Luft. Ein leicht verrutschter Kamerarahmen schiebt so nicht sofort einen Streifen Gesicht ins Spielbild.

Zum Ausrichten liegt eine Vorlage bei. Als Bildquelle in OBS laden, ganz nach oben ziehen, Deckkraft über einen Farbkorrektur-Filter auf etwa 60 Prozent, eigene Quellen einpassen, Vorlage wieder ausblenden.

Steht die Kamera in OBS als 4:3, vorher in den Kameraeinstellungen auf 16:9 umstellen. Ein 4:3-Bild in einen 16:9-Rahmen zu ziehen staucht das Gesicht.

4.4 Fremder Text im Bild

Was im Streambild steht, ist eingebrannt und lässt sich nachträglich nicht entfernen. Untertitel des Spiels und Live-Caption-Overlays landen also im Short, zusätzlich zu den eigenen Untertiteln.

- **Spieluntertitel:** abschalten. Sie laufen über die volle Bildbreite, werden vom Zuschnitt angeschnitten und bringen im Short nichts.
- **Live-Captions:** verschieben statt abschalten. Für Zuschauer ohne Ton sind sie wertvoll. Sie müssen nur aus dem Gameplay-Band heraus, etwa nach unten links oder in die rechte Spalte.

Welche OBS-Quelle den Text überhaupt malt, sagt `abdulive sources`.

5 Der laufende Betrieb

5.1 Clips auslösen

`!clip` im Chat	jeder Zuschauer
OBS-Hotkey	der Streamer
Chat-Spitze	erkennt auffällige Reaktionen, standardmäßig aus

Abklingzeiten sind nicht optional. Ohne sie ist `!clip` nach zwei Streams ein Spielzeug: 60 Sekunden global, 300 Sekunden je Zuschauer. Wer zu früh dran ist, bekommt eine Antwort im Chat.

Die Spitzen-Erkennung läuft von Anfang an mit und schreibt jeden Kandidaten samt Punktzahl mit, löst aber nichts aus. Sobald ein paar Streams protokolliert sind, lässt sie sich an echten Daten kalibrieren und scharfschalten.

5.2 Befehle

`abdulive check-env`	Selbsttest: Programme, Schriften, Modell, Verzeichnisse, OBS
`abdulive sources`	listet die OBS-Quellen und sagt, welche im 9:16-Ausschnitt landen
`abdulive calibrate`	prüft den Bildzuschnitt an einer echten Aufnahme
`abdulive render <datei>`	rendert eine Rohdatei nach, etwa nach einer Layoutänderung
`abdulive fetch-model`	lädt ein Sprachmodell
`abdulive update`	prüft auf eine neue Fassung

5.3 Läuft es?

```
Start-ScheduledTask -TaskName abdulive     # starten
Stop-ScheduledTask  -TaskName abdulive     # stoppen
```

Status und Zahlen liefert `http://127.0.0.1:4792/health`. Die Protokolle liegen unter `%LOCALAPPDATA%\abdulive\logs`.

> Bei einer Adresse im Browser immer `127.0.0.1` statt `localhost` verwenden, wenn etwas hängt. Der Dienst lauscht nur auf IPv4, und Windows probiert bei `localhost` zuerst IPv6. Das kostet zwei Sekunden.

6 Updates

Die Aufgabe `abdulive-update` sieht täglich um 5:00 Uhr nach. Findet sie eine neue Fassung, lädt sie das Paket, prüft die Prüfsumme und legt es bereit.

Eingespielt wird beim nächsten Start. Mitten im Stream wird nichts ausgetauscht. Schlägt der Tausch fehl, läuft die alte Fassung weiter.

Sofort übernehmen:

```
Stop-ScheduledTask -TaskName abdulive; Start-ScheduledTask -TaskName abdulive
```

7 Sicherung

```
& "$env:LOCALAPPDATA\Programs\abdulive\backup.ps1"
```

Schreibt eine in sich stimmige Kopie der Datenbank und hält 30 Generationen. Die Datenbank wird im laufenden Betrieb beschrieben, ein einfaches Kopieren ergäbe eine unbrauchbare Sicherung.

8 Deinstallation

```
& "$env:LOCALAPPDATA\Programs\abdulive\uninstall.ps1"
```

Nimmt die beiden Aufgaben, beendet den Dienst und löscht den Programmordner. Die Daten bleiben stehen, weil in der Datenbank Chatverläufe liegen. Wer neu installiert, findet seine Einrichtung wieder.

Wirklich alles weg, auch Datenbank, Clips und Sprachmodell:

```
& "$env:LOCALAPPDATA\Programs\abdulive\uninstall.ps1" -AuchDaten
```

9 Wenn etwas nicht geht

Symptom	Ursache und Abhilfe
Keine Clips	OBS-WebSocket aus oder Passwort fehlt. `check-env` zeigt es unter *OBS*.
Clips ohne Untertitel	Sprachmodell fehlt. `abdulive fetch-model`.
Untertitel voller Unsinn	Transkribiert wird die falsche Tonspur. Mikrofon gehört auf Spur 2.
Kopf sitzt schief	OBS-Layout weicht von der Vorgabe ab. `abdulive calibrate`.
Fremder Text im Short	Eine OBS-Quelle liegt im Gameplay-Band. `abdulive sources`.
Chat wird nur gelesen	Broadcaster-Konto nicht verbunden.
Oberfläche antwortet nicht	Protokolle unter `%LOCALAPPDATA%\abdulive\logs` ansehen.

10 Wo was liegt

`config\config.yaml`	Einstellungen. Keine Geheimnisse.
`config\layout.yaml`	Bildaufteilung für den Zuschnitt
`db\`	Datenbank und verschlüsselte Zugangsdaten
`clips\`	fertige Videos
`work\`	Rohdateien der Verarbeitung
`models\`	Sprachmodelle
`logs\`	Protokolle

Alles unterhalb von `%LOCALAPPDATA%\abdulive`.

Der Arbeitsordner wird noch nicht aufgeräumt. Rohdateien sammeln sich unbegrenzt an, mehrere hundert MB je Clip. Bis eine Aufbewahrungsregel gebaut ist, gelegentlich selbst nachsehen.