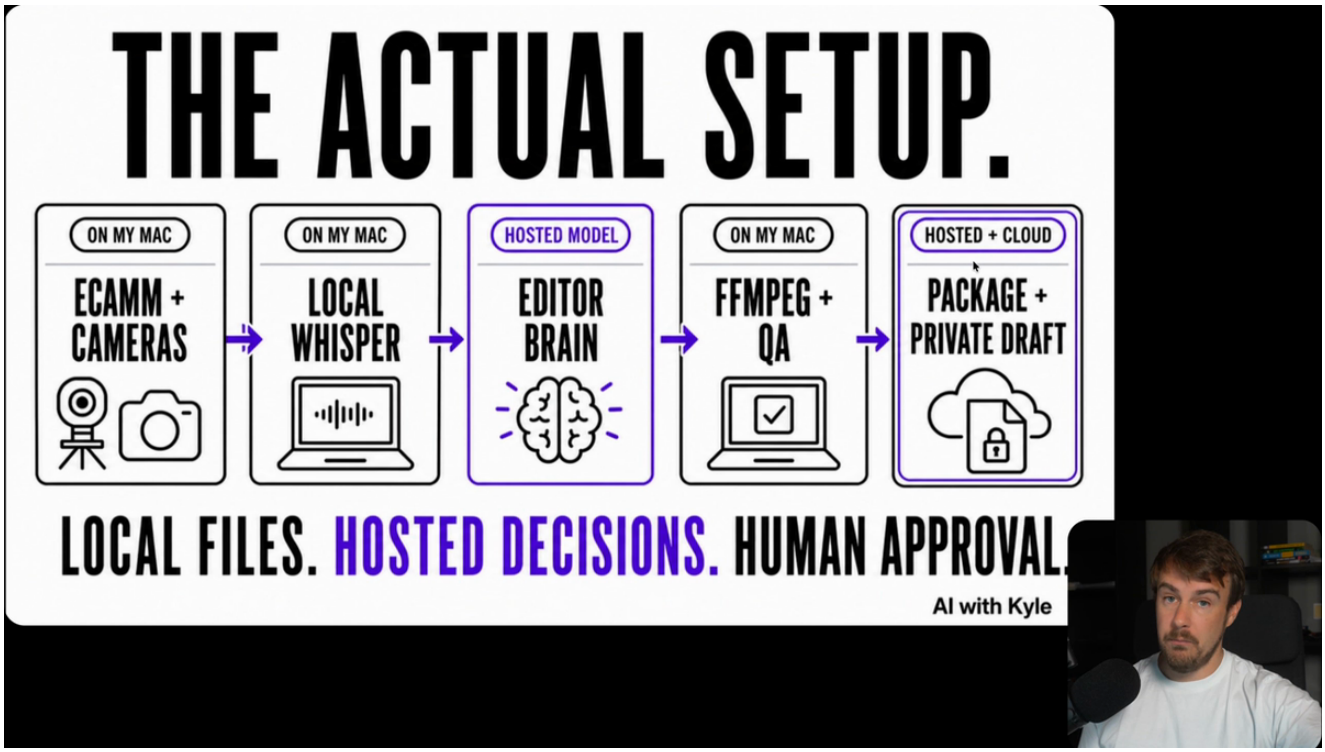
THE PRACTICAL HOW-TO

Build Your Own AI Video Editor

Codex or Claude Code + Whisper + FFmpeg



The version that actually works

A local, non-destructive pipeline for transcripts, edit decisions, review renders, quality checks, subtitles and private YouTube drafts.

RULE NUMBER ONE

The machine creates a review. A human approves what ships.

The viral version of AI video editing is one prompt: upload a video, ask the model to make it interesting and wait for magic.

That version is not dependable yet.

The version that works is a pipeline. A capable coding agent reads the transcript and makes editorial decisions. Local tools find the exact words, cut the video, build a review file and check the result. You approve the review before the system renders a master or uploads anything.

This guide shows you how to build that version.

1. What you are building

Your first system should turn one talking-head recording into:

- a full timestamped transcript;
- a structured edit decision list;
- a 720p review video;
- a machine QA report;
- subtitles;
- a 4K or source-resolution master after approval;
- a private YouTube draft after a second explicit instruction.

It should never overwrite raw footage, render the final master before review or publish a video by itself.

The production loop is:

1. Copy or reference the raw recording.
2. Extract and transcribe the audio locally.
3. Ask a smart model to make editorial decisions from the transcript.
4. Resolve those decisions against exact words in the transcript.
5. Cut and render a lightweight review file locally.
6. Run machine checks and then watch the review.
7. Approve, render the master and optionally upload a private draft.

2. Keep version one boring

Do not start with animated captions, stock footage, seven cameras and fifty zooms.

Use:

- one talking-head video;
- one audio track;
- cuts at complete sentence boundaries;
- no music;
- no automatic publication;
- a 720p proxy for every review;
- one human approval gate.

Once that works five times in a row, add a second camera, programme feed, captions, call-to-action overlays or Remotion animations.

Minimum stack

Codex or Claude Code builds and runs the workflow. Use a capable model for editorial judgement.

Whisper or whisper.cpp creates a local word-timestamped transcript.

FFmpeg and ffprobe inspect, cut, encode and verify the video.

Optional: Remotion or Hyperframes adds deterministic titles and animations after the core edit is dependable.

Optional: YouTube Data API creates a private draft. Publishing should stay manual.

You also need enough free disk space for the raw media, proxy files, review render and master. Video multiplies quickly. Keep every episode in its own run folder and delete temporary parts only after the final file has passed review.

3. Give the agent a folder contract

Create one project and keep every run separate:

```
ai-video-editor/
  config/
    edit-style.md
  raw/
  runs/
    2026-07-22/
      00-input/
      01-audio/
      02-transcript/
      03-decisions/
      04-review/
      05-qc/
      06-master/
      07-package/
```

The agent may read from raw/, but it must never modify or delete those files. Every generated artefact belongs inside the run folder. This makes failed experiments disposable and keeps the original footage safe.

Put your standing editorial rules in config/edit-style.md. Include what to remove, what to keep, your camera grammar, target length and exact end condition. When you correct a repeated mistake, update this file.

4. Ingest and transcribe locally

Ask the agent to inspect the video with ffprobe and save a source manifest containing duration, resolution, frame rate, audio sample rate and file size.

Then ask it to:

1. extract a mono audio file suitable for transcription;
2. run Whisper locally with word timestamps;
3. save the untouched raw transcript and a lightly corrected reading version;
4. preserve every original timestamp;
5. flag low-confidence words and product names rather than guessing.

Correct names such as Codex, Claude Code, Whisper, FFmpeg, Remotion and your own products before the editorial pass. Do not rewrite the spoken content. The transcript is evidence, not an article.

Transcript acceptance checks

- The opening and final spoken sentence are present.
- Duration roughly matches the source.
- Timestamps increase without large unexplained gaps.
- Repeated sections in the transcript actually exist in the recording.
- The last few seconds contain no hallucinated words from silence.

5. Ask the model for editorial decisions, not final timestamps

This is the most important design choice in the guide.

Models are good at deciding that setup, a repeated paragraph or an audience Q&A should be removed. They are much less dependable at returning exact cut times. In our production tests, a correct editorial decision could still drift by two to six seconds. That is enough to chop a sentence in half.

Ask for quoted text anchors around every change:

```
{
  "summary": "Open on the first complete thesis and end on the sign-off.",
  "cuts": [
    {
      "removeFromText": "Let me check whether the stream is working",
      "resumeAtText": "I keep seeing viral AI editing videos",
      "reason": "Remove pre-show setup",
      "risk": "low"
    }
  ],
  "shots": [
    {
      "startText": "The one-button dream",
      "endText": "build a workflow",
      "source": "face",
      "reason": "Keep the core opinion on camera"
    }
  ]
}
```

The model owns the reason and the text anchors. Local code owns the time boundaries.

Reject any decision whose quoted words cannot be found exactly or with a clearly logged fuzzy match. Do not silently guess.

6. Resolve exact boundaries

Write a deterministic boundary resolver that searches the word-timestamped transcript for each quoted anchor.

For every proposed cut:

1. find the nearest candidate matches;
2. check that they occur in the expected order;
3. re-transcribe a tight five-to-seven-second window around the proposed boundary if needed;
4. place the cut on a complete word or sentence boundary;
5. write the resolved times to a separate file;
6. fail the run if confidence is too low.

Do not let a later run overwrite the original model decision. Keeping both files makes it possible to see whether a bad join came from the model's judgement or the boundary resolver.

7. Render the review without audio seams

Build the 720p review before touching the master.

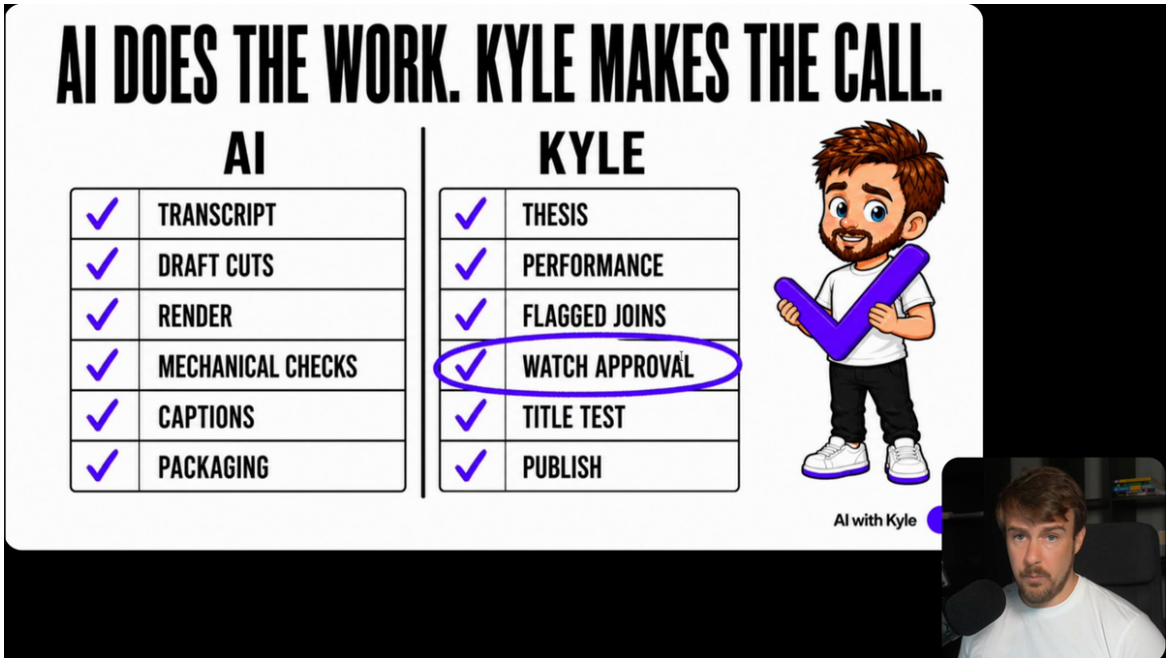
FFmpeg can assemble the retained video ranges. The tempting implementation encodes every segment with its own audio and then concatenates them. We tried that. It created audible AAC seams at some mid-speech cuts even though automated transcription thought the joins were fine.

Use this architecture instead:

1. render or concatenate the retained video-only segments;
2. build one continuous conformed audio timeline from the source;
3. encode the complete audio track once;
4. mux the final video and audio;
5. save a segment map from source time to output time.

The segment map is needed later for subtitles, chapter times and debugging.

8. Run machine checks, then use your eyes and ears



Your QA stage should produce evidence, not a green badge that grants permission to publish.

Check:

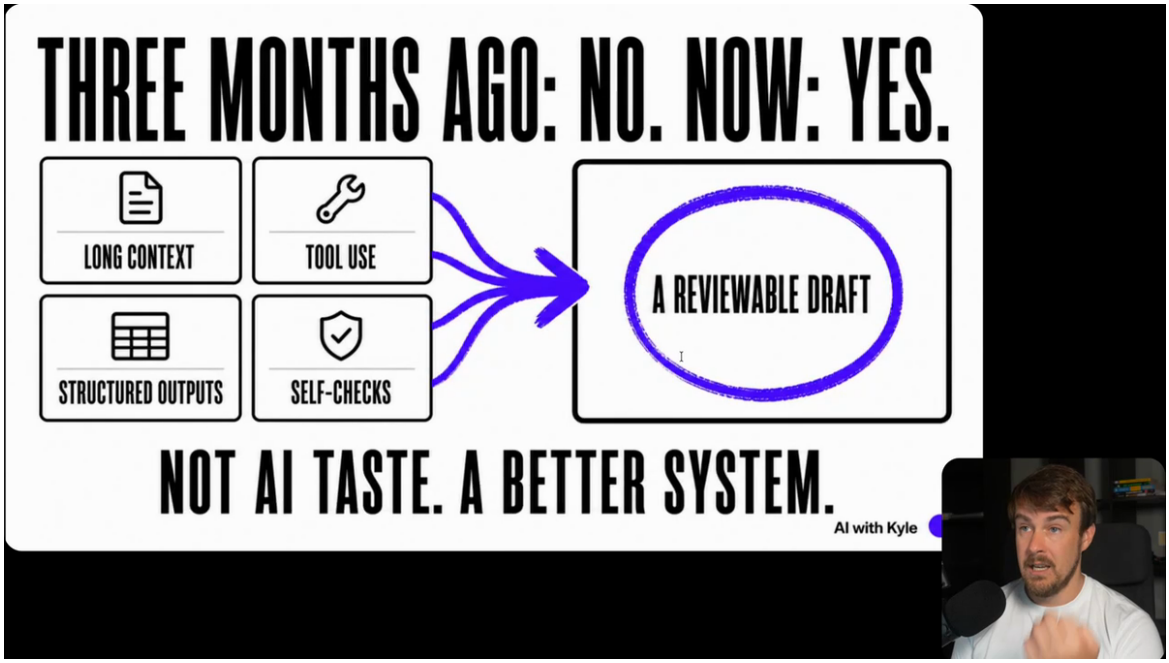
- full-file decode succeeds;
- expected and actual duration agree;
- audio and video end together;
- no long accidental silences appear;
- no black or corrupt frames appear;
- the opening begins on the intended line;
- the ending contains the complete sign-off;
- every discontinuous join reads as a complete thought;
- subtitle cues map through the edited timeline;
- the final seconds contain real speech energy when Whisper drops the last words.

Create a contact sheet every 20 to 30 seconds and a second sheet showing frames around every join.

Then stop.

Watch the opening minute. Listen to every flagged join. Scrub the ending. If the camera changes mid-speech, listen with headphones. Machine transcription can miss an ugly audio seam or a small sync error that a human notices instantly.

9. Approval, master and YouTube draft



Treat these as three separate permissions:

Review approved: the exact 720p file may be used as the edit source for a final render.

Master approved: the system may render at source resolution and export subtitles and packaging.

Upload approved: the system may create a private YouTube draft with title, description, tags, thumbnail and captions.

None of those permissions means publish.

Keep publishing, scheduling, Test and Compare and end screens manual until the whole system has a long clean record.

10. Starter build prompt

Paste the following into Codex or Claude Code inside a new empty project folder. Attach or point it at one short talking-head test video. Use a copy, not your only source file.

```
Build a local, non-destructive AI video editing pipeline for talking-head video.

Goal:
Turn one raw recording into a timestamped transcript, structured edit decisions,
a 720p review render, a QA report, subtitles and - only after explicit approval -
a source-resolution master and optional private YouTube draft.

Safety rules:
1. Never modify, move or delete source media.
2. Write all generated files into runs/<run-id>/.
3. Never render the master before I approve the exact 720p review file.
4. Never upload without a separate explicit instruction.
5. Never publish, schedule or make a YouTube video public.
6. Stop on ambiguous edit boundaries instead of guessing.
```

```
Tools:
- Use ffprobe for the source manifest.
```

```
- Use local Whisper or whisper.cpp with word timestamps.
- Use a capable model only for editorial decisions.
- Use quoted transcript anchors in the decision file.
- Resolve exact times with deterministic local code.
- Use FFmpeg for proxies, cuts, one continuous audio conform and final muxing.
- Keep a source-to-output segment map.

Version-one scope:
- One video source and one audio track.
- Talking-head content.
- Sentence-boundary cuts only.
- Remove setup, false starts, repeated takes, irrelevant tangents and post-sign-off chat.
- Keep the teaching spine, strong opinions, examples and final call to action.
- No music, B-roll, automatic zooms or generated animations.

Review gate:
- Render 720p first.
- Verify decode, duration, A/V delta, silence, black frames, opening, ending and joins.
- Produce an overview contact sheet and join contact sheet.
- Re-transcribe the review and print every join as heard.
- Stop and give me the review path, checksum and QA report.

Before writing code, inspect the available files, propose the folder structure and
implementation plan, and identify any missing local dependencies. Then implement and
test one stage at a time. Do not skip the approval gates to complete the task faster.
```

11. Run prompt for each episode

Once the system exists, each episode should need a much shorter instruction:

```
Process the new recording in raw/ as run <YYYY-MM-DD>.
Use config/edit-style.md.
Create the local transcript, editorial decision file, resolved boundaries,
720p review, QA report, subtitles and package draft.
Do not render a master or upload anything. Stop at the review gate.
```

After watching the review, be specific:

```
At 03:14 the audio join clicks. At 07:52 the cut removes the first word of the
next sentence. Fix both, explain the root cause and add a regression test or durable
rule so the same failure is caught before the next review.
```

That last sentence matters. A correction that only repairs today's file is support work. A correction that improves the pipeline is an editor getting better.

12. Common failures

The model picks good cuts but clips sentences

Cause: trusting model timestamps. Fix: quoted text anchors, deterministic matching and tight local re-transcription.

The joins sound clicky

Cause: encoding audio per segment. Fix: one continuous audio conform and one final audio encode.

The last words disappear in QA

Cause: Whisper sometimes drops speech at the end of a file. Fix: inspect tail energy and listen before changing the edit.

Review takes forever

Cause: rendering every attempt at full resolution. Fix: use a 720p proxy and render the master only after approval.

The edit is neat but boring

Cause: the source had no clear thesis or useful examples. Fix the recording. The editor cannot invent a good video without changing what you said.

Packaging invents a number

Cause: the packaging model tries to make a stronger title. Fix: require every number, price and superlative to be quoted from or directly supported by the transcript.

Your first build

Start with a five-to-ten-minute talking-head recording. Include a false start, repeat one sentence and add a clear sign-off. That gives the system something real to find without turning the test into a full production job.

Get one clean review. Then get five.

Only after that should you add camera switching, animated overlays, private YouTube upload or a longer livestream.

Build the boring system first. That is the one that ships.