Give Claude Code Eyes — the /watch setup

Let Claude watch, read, and summarize any video — visuals and words — in two lines. The exact setup from the carousel, with the real commands, so you can run it today.

You commented WATCH, so here's the whole thing. What the skill is, how to install it in whichever tool you use, how it actually works, and the two moves I'd learn first.

The thing that's actually broken

You paste a video link at an AI and ask what's in it, and it grabs the transcript. That's it. It reads the words and misses everything on the screen — the graph on slide 8, the on-screen text, the demo, the thing the person is pointing at while they talk. Half the video lives in the picture, and the AI never saw the picture.

So you get a summary that's technically correct and useless. It "watched" a design breakdown and can't tell you what was on the design, because all it ever had was the audio typed out.

The fix isn't a fancier video model. It's giving Claude both halves of the video — the words *and* the frames — so it reads the screen, not just the transcript.

The taste — what you end up with

Point it at a 45-minute talk. Under two minutes later Claude has the whole thing: a timestamped transcript AND a screenshot every few seconds. Ask "what was the hook in the first 20 seconds" and it tells you the exact words spoken, what was on screen at that moment, and the second the pattern-interrupt hit — because it actually looked.

The real cost of this, from the tool's own public numbers: over 5 hours of video transcribed for **$0.39**. Not a typo. And a lot of videos cost nothing to transcribe at all (more on that below).

(Those figures are the open-source tool's published examples, not an Axionox client result — your videos give you your own numbers.)

Here's the setup.

What it is

The skill is `/watch`, from the repo `bradautomates/claude-video` on GitHub. It's open-source, MIT-licensed, around **6k stars**, currently **v0.2.0**. The one-liner from the repo says it best:

> Give Claude the ability to watch any video. `/watch` downloads, extracts frames, transcribes, hands it all to Claude.

That's the whole idea. It's not a model you pay per-video for — it's a small toolchain that does the grunt work and hands Claude the raw material.

Step 1 — install it (pick your tool)

Claude Code:

```
/plugin marketplace add bradautomates/claude-video
/plugin install watch@claude-video
```

claude.ai: download `watch.skill` from the repo, then in the app go to **Settings** → **Capabilities** → **Skills** and add it.

Codex:

```
git clone https://github.com/bradautomates/claude-video.git
~/.codex/skills/watch
```

Zero config to start. The two tools it leans on — **yt-dlp** and **FFmpeg** — auto-install on first run, so you don't set anything up ahead of time. Install, then run.

Step 2 — how it actually works (so you trust the output)

No video model in the loop. Three plain pieces do the work:

- **yt-dlp** downloads the video.
- **FFmpeg** slices it into frames (a screenshot every few seconds) + pulls the audio.
- **Claude** reads the images and the text together.

That's the trick: it captures BOTH a screenshot every few seconds AND a timestamped transcript — visuals *and* words — so Claude gets the whole story instead of a wall of text with no picture attached.

On the transcript specifically:

- If the video has public captions, it pulls those — **free**.
- If it doesn't, it transcribes the audio with **Whisper on Groq** (`whisper-large-v3`). That's where the $0.39-for-5-hours comes from. Groq's free tier gives you roughly **2 hours of transcription every hour**, so a lot of the time you're not paying anything at all.

Nothing exotic. Download, slice, read. That's why it's cheap and why you can trust what comes back.

Step 3 — run it (two lines)

Watch a whole video:

```
/watch:watch <video-url>
```

Zoom into just one segment — saves time and tokens when you only care about one moment:

```
/watch:watch <video-url> --start 00:35 --end 01:38
```

That's the entire interface. One command to watch it all, the same command with two flags to zoom in.

The two moves I'd learn first

1. Break down a viral hook. Point it at a reel or short that blew up and ask *why*. Because it sees the frames, Claude can name the exact words in the first few seconds, what was on screen while they were said, and the

precise second the pattern-interrupt lands. That's a real teardown of what made it work — not a vibe, the actual mechanics — and you can copy the structure into your own stuff.

2. Zoom into one moment. You don't always need the whole 45 minutes. If the part you care about is the demo at 12:30, run it with `--start 12:15 --end 13:10`. Claude only processes that slice — faster, cheaper on tokens, and the answer is about the thing you actually asked. Learn this second; it's the one that keeps your costs near zero.

Where this fits in the bigger picture

This is one intake pipe for a bigger idea: feeding an AI the *real* raw material instead of a lossy summary of it.

- **Upstream:** whatever holds knowledge you keep re-explaining — a talk, a course, a competitor's ad, your own recorded call, a demo.
- **This step:** turning that video into something Claude can actually reason over — frames + transcript, not just typed-out audio.
- **Downstream:** every task after it gets better. Studying what makes content convert, pulling steps out of a tutorial, turning one long video into ten pieces of your own — all of it improves the moment the AI can see the screen.

Give it eyes once and every video task after it inherits them.

Your next step

Don't overthink it. Grab one video you'd actually want broken down — a reel that's crushing in your niche, or a talk you never finished — install `/watch`, and run the two lines on it:

```
/watch:watch <that-one-video-url>
```

Then ask it to name the hook and what was on screen. Two minutes, and you'll see the difference between an AI that read the transcript and one that actually watched.

The cost of not doing it: every video you feed an AI stays half-blind, and you keep getting summaries that missed the part that mattered. This fixes it at the source.

Built by Axionox. I build real AI systems for a living — this `/watch` setup is one small piece of how I get an AI to work off what's actually there instead of a thin summary of it. If you want to see what this looks like wired into a real workflow, reply to the DM and I'll walk you through it. Follow for more like this.