The YouTube Agent Skill Setup Playbook

How I install all 12 skills, fill in the voice profile every one of them reads, and run the first six commands.

What this actually is

The YouTube Agent Skill is twelve Claude Skills that sit in your **~/.claude/skills/** folder and run a YouTube channel between them. Eleven of them write. The twelfth uploads, and only after you have typed the word publish. It is MIT licensed and it costs nothing to install.

I keep seeing people install a script generator, get four paragraphs of AI mush back, and decide the whole category is useless. That happens because a generic model has no idea what you sound like and no idea what is working in your niche this month. This pack fixes both of those with one file you fill in and two fetchers that go and get real data off the YouTube Data API before anything gets written.

It is a fork of Jake Schincariol's original pack. The six Python tools, the 21 hook formulas in **hooks.json** and the voice template are byte identical to his. What this edition adds is the fetch layer, **tools/channel.py** and **tools/transcript.py**, so six of the skills stopped asking you to paste data in by hand, plus **/yt-publish** as a twelfth skill. That is the whole difference.

By the end of this you will have all twelve installed, a voice profile Claude reads before it writes a word, a free YouTube Data API key, and a first command that tells you which videos in your niche actually broke out.

01

How it actually works

1. You call a skill by name in Claude Code, like **/yt-script** or **/yt-viral**. Claude loads that skill's SKILL.md, which tells it the job, the format and the rules for that one task, so a general model starts behaving like a specialist instead of a chatbot.
2. Before it writes, every skill reads **~/.claude/youtube/voice.md**. That file is your sentence length, your vocabulary, the things you refuse to say, and who you are talking to. This is why two people running the same command get two different scripts.
3. Where a skill needs real numbers, it shells out to a Python tool rather than guessing. **hookscore.py** scores a hook on five properties, **swipe.py** ranks competitor videos by how far each beat its own channel's median, and **retention.py** reads your retention export against the transcript so it can say what you were mid-sentence on when people left.

That last part is the bit worth understanding. Six of the twelve skills call a real script that does arithmetic on your data rather than writing a plausible-sounding number, and those scripts run on a clean Python 3 with nothing installed.

02

The one thing almost everyone gets wrong

They skip **voice.md**, run **/yt-script**, read the output, and conclude the pack is average. The pack is only ever as specific as that one file. It is the difference between a script that sounds like you and a script that sounds like every other AI YouTube video published that week.

> **SPEND THE TEN MINUTES**
>
> Copy templates/voice.md to ~/.claude/youtube/voice.md and fill it in properly, or paste three of your own video transcripts into Claude and say "write my voice.md from these". Every one of the twelve skills reads that file. A blank profile gives you generic output from all of them, and no amount of re-prompting later will fix it.

The second thing people get wrong is running **/yt-script** first. Run **/yt-viral** first. Writing a great script for a topic nobody in your niche is watching is a waste of a good script.

03

The files

> **SOURCE CODE**
>
> The full repo, MIT licensed, clone it or paste the link straight into Claude:
>
> **https://github.com/nessalazne/youtube-agent-skill**

This is what you get when you clone it. The Python tools are the reason the skills are more than prompts, so it helps to know which file does what before you start.

REPO STRUCTURE

```
tools/channel.py             a channel's recent uploads, off the Data API (stdlib only)
tools/transcript.py          a video's caption track as text or .srt
skills/yt-script/hooks.json  the 21 hook formulas, shared with the classifier
skills/yt-script/hookscore.py    the five-property hook panel
skills/yt-package/title.py       title + thumbnail linter
skills/yt-edit/deadair.py        edit decision list
skills/yt-chapters/chapters.py   chapter boundaries, validated
skills/yt-retention/retention.py hook leak, cliffs, slide
skills/yt-viral/swipe.py         outliers by own-channel median
skills/yt-publish/publish.py     the Blotato upload
templates/voice.md               the profile every skill reads
```

The config lives in its own folder, not in the repo, so a **git pull** never touches your keys. Create this file by hand:

~/.CLAUDE/YOUTUBE/.ENV

```
YOUTUBE_API_KEY=AIza...

# only needed if you want /yt-publish to upload for you
BLOTATO_API_KEY=...
BLOTATO_ACCOUNT_YOUTUBE=1234
```

And these are the twelve commands, so you know what you are installing:

COMMAND	WHAT IT DOES
/yt-script	One idea into a script. Five hooks off 21 formulas, scored, then the spoken script with retention beats marked.
/yt-package	Title and thumbnail as one pairing, linted for truncation, duplication and vagueness.
/yt-edit	A transcript into an edit decision list: dead air, filler cues, retakes, with timecodes.
/yt-comment	The comment section sorted into four piles, replies in your voice, and which one to pin.
/yt-plan	A week that fits the hours you have. One anchor, one cheap one, three Shorts.
/yt-viral	What is working in your niche, ranked by multiple over each channel's own median.
/yt-retention	Your retention export read properly: hook leak, the cliffs, the slide, and what to change.
/yt-shorts	The Shorts already inside a long video, with a new first line written for each.
/yt-seo	The description, the tags worth having, and the three queries this should win.
/yt-chapters	Chapters from a transcript, validated against YouTube's rules so they render.
/yt-audit	The whole channel, ending in one fix rather than twenty.
/yt-publish	Uploads or schedules through Blotato. Private by default, two yeses.

Set it up in Claude Code or Codex

1. **Clone the repo.** Open a terminal anywhere and run **git clone https://github.com/nessalazne/youtube-agent-skill.git**. It is small and it has no install step.
2. **Copy the skills into place.** For Claude Code, run **cp -r youtube-agent-skill/skills/yt-* ~/.claude/skills/**. Each skill ends up in its own folder, and the folder name has to match the **name:** field inside that folder's SKILL.md. If you rename a folder, the command stops resolving.
3. **Or install it as a plugin instead.** In Claude Code, run **/plugin marketplace add nessalazne/youtube-agent-skill** then **/plugin install youtube-agent**. Same twelve skills, and updates come down with the marketplace.
4. **Or let Claude do it.** Paste the repo URL into a Claude Code session and say "Install this skill, then confirm /yt-script works." It reads the README and does the copying itself.
5. **For Codex, put them where Codex looks.** Copy the same **yt-*** folders into your project folder or into **.codex/skills/**, then add a pointer in **AGENTS.md** saying where the skills live and that they should be read before YouTube work. Codex does not scan **~/.claude/skills/** on its own.
6. **Project-local instead of global.** If you only want these in one repo, copy the same folders into that repo's **.claude/skills/** rather than your home directory.
7. **Write your voice profile.** Copy **templates/voice.md** to **~/.claude/youtube/voice.md** and fill it in. If you would rather not start from a blank file, paste three of your own transcripts into Claude and ask it to write the profile from them, then edit what it gets wrong.
8. **Get a free YouTube Data API key.** In the Google Cloud console, enable "YouTube Data API v3" and create an API key. Ask Claude to add it to **~/.claude/youtube/.env** as **YOUTUBE_API_KEY**, then open a new terminal window so the change is picked up.
9. **Install the one dependency.** Run **pip install youtube-transcript-api**. That is only for **tools/transcript.py**. Every other tool in the pack runs on a clean Python 3.
10. **Test it.** Run **python3 tools/channel.py @yourhandle**. A working setup prints your last 15 uploads with their view counts in a readable list. If that works, type **/yt-script** in Claude Code and check that it asks you for an idea instead of saying it does not know that command.

> 💡 **QUICK TIPS**
>
> - Always look channels up by **@handle**, never by name. Three channels by handle costs 9 units of your daily 10,000. The same three by name costs 306.
> - Every skill still works without the API key. The fetchers save you typing, they are never a requirement, so you can paste data in and lose nothing but time.
> - No Claude Code at all? Paste a single SKILL.md at the top of a chat and it runs as a mode. You lose the Python tools, which is most of the point of /yt-script, /yt-retention and /yt-edit.

04

The commands I actually run

FIND WHAT IS WORKING BEFORE WRITING ANYTHING

```
/yt-viral

My niche is [YOUR NICHE]. Compare these channels: [@YOUR HANDLE] [@RIVAL 1] [@RIVAL 2] [@RIVAL 3].
Rank every video from the last 90 days by its multiple over that channel's own median, minimum 2.0x.
For the top five, break down the hook, the title formula and the first 15 seconds, then tell me which one I could rebuild in my voice this week.
```

TURN THE WINNER INTO A SCRIPT

```
/yt-script

Idea: [THE ANGLE YOU PICKED FROM YT-VIRAL].
Format: [SHORT or LONG FORM], target length [SECONDS OR MINUTES].
Give me five hooks scored on the panel, tell me which one you would shoot and why the other four are weaker, then write the spoken script with the retention beats marked.
```

TITLE AND THUMBNAIL AS ONE THING

```
/yt-package

Script: [PASTE THE SCRIPT OR ITS ONE-LINE PROMISE].
Give me three title and thumbnail pairings. Lint each one for truncation on mobile, for the title and the thumbnail saying the same thing twice, and for vagueness. Tell me which pairing you would run.
```

READ THE RETENTION EXPORT PROPERLY

```
/yt-retention

Retention CSV: [PATH TO THE FILE YOU EXPORTED FROM YOUTUBE STUDIO].
Transcript: [PATH TO THE .SRT].
Find the hook leak, every cliff, and the slide. For each cliff, quote what I was saying at that timestamp and tell me what to cut or move.
```

MINE A LONG VIDEO FOR SHORTS

```
/yt-shorts

Transcript: [PATH TO THE .SRT].
Find the moments that already stand alone as Shorts. For each one, give me the in and out timecodes, a new first line written to work cold with no context, and a one-line reason it holds up on its own.
```

The two fetchers run from the terminal rather than from a slash command, and they write the exact formats the older tools already read:

THE FETCH LAYER

```
# the last 15 uploads from three channels, in the shape swipe.py eats
python3 tools/channel.py @me @rival1 @rival2 --json > collected.json
python3 skills/yt-viral/swipe.py collected.json --min 2.0

# a caption track as .srt, in the shape deadair and chapters eat
python3 tools/transcript.py "$URL" --srt > t.srt
python3 skills/yt-edit/deadair.py t.srt
python3 skills/yt-chapters/chapters.py t.srt --target 8
python3 skills/yt-retention/retention.py retention.csv --transcript t.srt
```

BEFORE YOU EVER PUBLISH

Run python3 skills/yt-publish/publish.py --check first. It prints the pinned channel's real name rather than a four digit id, and it refuses outright if that id turns out to belong to a different platform. Then dry run it with --dry-run before the real thing. Nothing uploads public unless you pass --privacy public every single time.

When it breaks

ERROR	FIX
Claude says it does not know /yt-script	The folder did not land in the right place, or you renamed it. Check that ~/.claude/skills/yt-script/ exists and that the folder name matches the name: field inside its SKILL.md. Restart the session after copying.
The scripts come back generic and could be anyone's	voice.md is empty or you never copied it to ~/.claude/youtube/voice.md. Fill it in, or paste three of your own transcripts and ask Claude to write it from them.
channel.py returns a quota or 403 error	The key exists but YouTube Data API v3 was never enabled on that Google Cloud project, or you burned the daily 10,000 units looking channels up by name. Enable the API, and use @handles.
transcript.py fails on import	It is the one tool in the pack with a dependency. Run pip install youtube-transcript-api.
The key is in .env but nothing reads it	It has to be at ~/.claude/youtube/.env, not in the cloned repo folder, and the terminal you opened before saving it will not see it. Open a new window.
swipe.py returns nothing at all	--min 2.0 is filtering everything out because no video in your sample doubled its own channel's median. Lower it to 1.5, or collect from more channels.
Codex ignores the skills completely	Codex does not read ~/.claude/skills/. Copy the yt-* folders into the project or .codex/skills/ and point at them from AGENTS.md.
publish.py --check names a channel I do not recognise	BLOTATO_ACCOUNT_YOUTUBE is pointing at the wrong account. Four digit ids all look alike, which is exactly why --check prints the name. Fix the id before you run anything else.
A hook scored low but I know it works	hookscore.py was calibrated on 74 real hooks and separates bad hooks from real ones well. It separates one creator's hits from their own misses barely at all. A low score is a reason to look again, not an instruction.