The Claude Sound Effects Stack

The three free tools that put real sound effects on a Claude edit, with the install commands and the prompts I run on a finished render.

Why I built this

Claude Code got good at the picture before it got good at the sound. It will cut, grade, caption and render a vertical video that looks like someone spent an afternoon on it, and then it will put a whoosh half a beat late, at twice the level of the voiceover, on a transition that did not need one.

The reason is simple. Claude cannot hear. It wrote every frame of the picture, so it can reason about the picture. The audio comes back as a file it has no sense for, so it guesses, and a guess about sound is audible immediately.

What follows is the stack I run instead: a library an agent can search on its own, a skill that reads the timeline and places a real recording on the exact frame, and a toolkit that makes a new sound in code when the library has nothing. All three are free. Two of them never leave your machine.

01

How it actually works

1. Openverse fills the shelf. It is WordPress's search engine for openly licensed media, and its REST API is open with no key and no login. As of October 2026 the audio index holds about 5.2 million records: 3,983,517 from Wikimedia Commons, 644,709 from Jamendo and 591,897 from Freesound. Freesound is the one that matters here, because that is where raw effects live rather than music.
2. real-sfx places from that shelf. It is a Claude Code skill that reads your render, works out which moments need a sound, picks a recording from your indexed library, processes it, and puts it on the exact frame. Your music and voiceover stay where they are. You get back a new MP4 plus a separate SFX track.
3. earshot fills the gaps and then scores the result. It gives a coding agent ears: describe a file in words, score a recording against a description, find the best moment inside a long take, and critique a finished film against a profile of professional launch videos. It also ships a bank of 252 sounds stored as code that render by id in seconds, with no models and no network.

02

The one thing almost everyone gets wrong

They install real-sfx and expect sounds to come with it. The skill ships with no sound library at all. It is a placement engine, not a pack. Point it at an empty folder and it will do nothing useful, and the failure looks like the skill being broken rather than the library being empty.

WATCH OUT

Run setup with --uisfx on the first install. That flag gives you 78 CC0 UI cues so the pipeline has something to place while you build the real library from Openverse. Without it, your first run produces a silent SFX track and a cue report full of unmatched moments.

The second half of the same mistake is FFmpeg. On macOS the plain brew formula leaves out the drawtext and rubberband filters, and real-sfx uses both for its frame sheets and pitch shifts. Install ffmpeg-full and put it first in PATH.

03

The tools and the search calls

SOURCE CODE

real-sfx, the Claude Code skill that places real recordings on the frame:

https://github.com/Clipwing/skills/tree/main/skills/real-sfx

SOURCE CODE

earshot, ears for coding agents plus the 252-sound code bank:

https://github.com/Leonxlnx/earshot

Openverse needs no install and no account. A search is one GET request, and every result carries the direct file URL and the attribution line you need to credit it.

OPENVERSE-SEARCH.SH

```
# Any sound, CC0 only, from Freesound
curl -s 'https://api.openverse.org/v1/audio/?
q=metal+door+slam&license=cc0&source=freesound&page_size=10' \
  | python3 -m json.tool

# Just the download URLs
curl -s 'https://api.openverse.org/v1/audio/?q=whoosh&license=cc0&page_size=20' \
  | python3 -c 'import sys,json; [print(r["url"]) for r in json.load(sys.stdin)
["results"]]'
```

FETCH_SFX.PY

```
#!/usr/bin/env python3
"""Download CC0 sound effects from Openverse into a folder real-sfx can index."""
import json, sys, urllib.parse, urllib.request
from pathlib import Path

API = "https://api.openverse.org/v1/audio/"
OUT = Path("sfx_library")

def fetch(query, count=10):
    params = urllib.parse.urlencode({
        "q": query,
        "license": "cc0",
        "source": "freesound",
        "page_size": count,
    })
    req = urllib.request.Request(API + "?" + params,
                                 headers={"User-Agent": "sfx-stack/1.0"})
    with urllib.request.urlopen(req) as r:
        return json.load(r)["results"]

def main(queries):
    OUT.mkdir(exist_ok=True)
    credits = []
    for q in queries:
        for i, item in enumerate(fetch(q)):
            name = f"{q.replace(' ', '_')}_{i}.{item['filetype']}"
            urllib.request.urlretrieve(item["url"], OUT / name)
            credits.append(f"{name}  {item['duration']} ms  {item['attribution']}")
            print("saved", name)
    (OUT / "CREDITS.txt").write_text("\n".join(credits))

if __name__ == "__main__":
    main(sys.argv[1:] or ["whoosh", "click", "impact", "swish", "riser"])
```

Fields worth knowing on each result: **url** is the direct file, **attribution** is the ready-made credit line, **foreign_landing_url** is the source page, **duration** is in milliseconds, and **filetype** is usually mp3. Filter with **license=cc0** when the video is commercial and you want to skip attribution bookkeeping entirely.

> **EVERYTHING REFERENCED HERE**
>
> - real-sfx skill (Clipwing)
>   https://github.com/Clipwing/skills/tree/main/skills/real-sfx
> - earshot (Leon Lin)
>   https://github.com/LeonxInx/earshot
> - Openverse audio search
>   https://openverse.org/search/audio
> - Openverse API docs
>   https://api.openverse.org/v1/

Set it up in Claude Code or Codex

1. Get the prerequisites in place. Python 3.10 or newer covers both tools, and FFmpeg with ffprobe has to be on your PATH. On macOS run **brew install ffmpeg-full** and then **export PATH="$(brew --prefix ffmpeg-full)/bin:$PATH"** in your shell profile. On Windows, **winget install Python.Python.3.12** and **winget install Gyan.FFmpeg**.
2. Clone the real-sfx skill: **git clone https://github.com/Clipwing/skills**, then copy **skills/real-sfx** into **~/.claude/skills/real-sfx** on macOS or **%USERPROFILE%\.claude\skills\real-sfx** on Windows. The folder name has to match the name field in SKILL.md, so leave it as real-sfx. Restart Claude Code afterwards or the skill will not be listed.
3. For Codex, put the same folder in the project itself or in **.codex/skills/real-sfx/**, then add a line to **AGENTS.md** telling the agent the skill is there and when to reach for it. Codex does not scan a global skills folder the way Claude Code does.
4. Build the sound library. Run **python fetch_sfx.py whoosh click impact swish riser** from the previous section to pull CC0 effects into **sfx_library/**, then index it: **python scripts/setup.py init --library main="$PWD/sfx_library" --uisfx --index**. Confirm with **python scripts/setup.py check**.
5. Install earshot in its own virtualenv: **git clone https://github.com/Leonxlnx/earshot && cd earshot**, then **python -m venv .venv**, **.venv/bin/python -m pip install torch --index-url https://download.pytorch.org/whl/cpu**, and **.venv/bin/python -m pip install -e ".[mcp,llm,dev]"**. Finish with **.venv/bin/python -m earshot doctor --load**, which downloads the models and tells you if anything is missing.
6. Register earshot as an MCP server so the models stay loaded between calls: **claude mcp add earshot --scope user -e EARSHOT_THREADS=3 -- /path/to/earshot/.venv/bin/python -m earshot mcp**. Use the real absolute path to that venv. Without the MCP server every CLI call reloads the models and costs 20 to 40 seconds before it answers.
7. Test the whole thing on one render. Ask Claude to sound-design a short MP4 you already have, then run **earshot critique out_sfx.mp4 --video out_sfx.mp4** on the result. A successful first run leaves you an ***_sfx.mp4**, an ***_sfx_track.wav**, a cue report listing each placed sound and its timestamp, and a critique scorecard with timestamped notes.

> 💡 **QUICK TIPS**
>
> - Only earshot's optional Gemini critique wants an API key, and it is off by default. If you want it, ask Claude to add GEMINI_API_KEY to your shell profile, then open a new terminal so the variable is actually loaded.
> - Keep the CREDITS.txt the fetch script writes. Anything that is not CC0 needs its attribution line published with the video, and recovering that later from a folder of mp3 files is miserable.
> - Index your library once per folder, not once per video. Re-running setup with --index on an unchanged folder just costs you time.

04

The commands I actually run

STOCK THE LIBRARY BEFORE ANYTHING ELSE

```
Search the Openverse audio API for CC0 sound effects matching these types: [WHOOSH, CLICK, IMPACT, SWISH, RISER, UI POP]. Use https://api.openverse.org/v1/audio/ with license=cc0 and source=freesound. Download the top 10 of each into ./sfx_library, name the files by type, and write every attribution string to CREDITS.txt. Skip anything longer than 5 seconds.
```

SOUND-DESIGN A FINISHED RENDER

```
Use the real-sfx skill to sound-design [PATH/TO/RENDER.mp4]. The library is indexed at ./sfx_library. Keep the existing music and voiceover exactly as they are. Place effects on the cut frames, not near them. When you are done, show me the cue report and tell me which placements you were least confident about.
```

REPLACE A SOUND THE SKILL GOT WRONG

```
The sound at [TIMESTAMP] in the cue report is wrong, it should be [A SHORT DRY WOOD KNOCK, NOT A METAL HIT]. Render a replacement with earshot sfx, check it against that description with earshot match, and swap it into the SFX track at the same frame and the same level as the one it replaces.
```

BUILD ONE KIT FOR A WHOLE FILM

```
Run earshot kit init --preset launch for [PROJECT FOLDER] so every effect in the film comes from one coherent set and gets mixed against the music bed instead of at a fixed level. Then render the kit and give me the measured onset and peak for each role.
```

SCORE IT BEFORE YOU POST IT

```
Run earshot critique on [PATH/TO/FINAL_sfx.mp4] with the video attached. Give me the verdict, the metrics that fell out of range, and the three fixes that would move the score most. Then fix them and run it again so I can see the before and after.
```

When it breaks

ERROR	FIX
Claude says it cannot find the real-sfx skill	The folder is in the wrong place or Claude Code was not restarted. It must sit at ~/.claude/skills/real-sfx, and the folder name must match the name field inside SKILL.md.
No such filter: drawtext / rubberband	You have the slim FFmpeg build. On macOS install ffmpeg-full and put it first in PATH; the plain formula omits both filters and real-sfx needs them for frame sheets and pitch shifts.
ffprobe not found	FFmpeg is installed but ffprobe is not on PATH. Both binaries have to be reachable, not just ffmpeg. Check with 'ffprobe -version' in a new terminal.
The SFX track comes out silent and the cue report shows unmatched moments	The library is empty or was never indexed. Re-run setup.py init with --uisfx and --index, then setup.py check.
Openverse returns 0 results for an obvious sound	license=cc0 plus source=freesound is a narrow filter. Drop source first, then license, and re-run. Wikimedia Commons holds the largest share of the index but far fewer short effects.
Every earshot command hangs for half a minute before answering	Each CLI call is a fresh process loading the models. Register the MCP server so the models stay resident, or run several calls from one Python script with 'from earshot import api'.
earshot doctor reports missing models	Run it as 'earshot doctor --load' so it downloads them instead of only checking. Make sure you are calling the binary inside .venv, not a system-wide python.
The new sounds double up with the ones already in the render	The project's synthesized audio layer is still in the mix. real-sfx can subtract the project's own sfx stem or rebuild the score without effects, but it needs to know where that stem is, so tell it in the prompt.

Ready to go deeper?

Want the videos made for you? Paste any link or idea into Content Engine and get a published video in under 3 minutes, with DM automation armed to capture leads.

Meet Content Engine →

https://builds.digicuratoragency.com/content-engine/