



TWEETDECK ON STEROIDS

TACTICAL MEDIA BROADCAST HUB

A self-hosted, multi-platform broadcast newsroom for collectives: a column wall that watches every platform at once, chats that organize themselves, votes that turn into posts, and a live channel behind it all.

CALL FOR HELP — 2026-10-09

We're looking for **testers** across the whole flow and **intake hackers** for the hard problem at the end of this document. Everything under "running today" is built and running — this is an invitation to use it, break it, and help finish the rest.

TweetDeck died when X locked its doors. We built what should have replaced it — and went much further. Watch everything, decide together, publish everywhere, mix live, and keep an archive no platform can take away. Self-hosted. No algorithm in between. No platform holding your keys.

THE COLUMN WALL — EVERY PLATFORM IN ONE WALL

Live columns watch all your connected accounts at once: Mastodon, Telegram, Bluesky, PeerTube, Twitch, and more. The whole collective sees the same picture at the same moment.

Every post is rendered safe (hostile HTML is parsed into a typed format — scripts, tracking pixels and trick links are stripped at ingest and tested against an attack corpus), media is proxied through your server, and duplicates are deduplicated into the archive automatically.

Columns are becoming **editable**: a column can be a platform feed, a **keyword search across platforms** (one query like "*acampada26*" pulling from every network that allows it), a curated **list of accounts across platforms**, or a saved search. Columns persist per group, can be reordered, paused, and shared.

COLLABORATIVE CHATS

Chats from every platform bridge into one shared overlay and one web chat — viewers, contributors and moderators in the same room, with flood governors and moderation tools keeping it usable during action.

The chat is a coordination surface, not just a mirror: it sits next to the live program, next to the file store, next to the drafts. What people say in the chat drives what happens in the system.

VOTING AND DECISION MAKING — IN THE CHAT, AUTOMATICALLY

Run a poll inside any bridged chat: "**post this?**" — and set the rules for what counts as a pass. Configurable per poll:

- ▶ approved by **x of y** admins

- ▶ approved by **x of y participants**
- ▶ mixed quorum (admins + participants), veto rights, timeboxes, anonymous or named voting

Votes come in through Telegram buttons and the web overlay — live tally bars, quorum meters, countdowns. Ineligible votes are stored but never counted; vetoes fail the poll instantly; revotes are supported; everything lands in a tamper-evident audit trail.

When the quorum passes, the posting bot fires automatically — the approved content goes out to the channel, topic, or account the poll was about. No admin phone in the loop. Decisions happen at the speed of the chat, with the full trail preserved.

CONFIGURABLE POSTING BOTS — EVERY ACCOUNT, ITS OWN RULES

Every connected account carries its own settable rules for what it takes to post:

- ▶ **immediate** — no gate, post as today
- ▶ **admin-only** — an admin's approval is the whole quorum
- ▶ **quorum** — configurable thresholds of admins and/or participants
- ▶ per **topic** and per **content type** overrides, dry-run shadow mode, timeboxes, reset-on-edit

A rules editor shows the resolved effective policy before you save it — and a blocked queue shows every post waiting on its quorum with a live gap meter. Nothing is ambiguous about why something did or didn't go out.

SUBCHATS AND TOPICS — STRUCTURE THAT FOLLOWS THE WORK

Forum topics become **subchats**: working groups inside a channel, created and managed by the bot. Accounts are **assigned to topics** — the press crew on the press topic, the stream crew on the stream topic — and each topic carries its own posting rules and its own view of the store. Topics can be bound to broadcast channels, so a topic isn't just conversation: it's a control surface.

CHAT-LINKED FILES — THE STORE ORGANIZES ITSELF

Files posted into the chat land in the **group's file store automatically** — content-addressed, encrypted, deduplicated (the same clip shared five times is stored once).

Admins organize: assign items to folders, copy them across subfolder trees, refile by type or date — from **chat commands** (`/ls`, `/find`, `/move`) or from the **mobile file screen** that lives next to the chat, with deep links from bot messages straight into the filtered folder view.

Bot listings, search across the whole library, and an audit of every move: the file storage stays organized because the people in the chat keep it organized, from their phones.

CROSS-AMPLIFICATION AND RETWEETS — ONE APPROVAL, EVERY PLATFORM

One approved draft fans out to every connected account, native to each platform. Where a platform has native quoting (Mastodon quote posts, Bluesky, Telegram forwards), it's used. Where it doesn't, the system generates a proper attributed quote — author, instance, link back — never a silent copy-paste.

Amplifying a sister collective's post keeps their names on it by default. Scheduling, per-account timing, and rate awareness come along for free.

LIVE VIDEO MIXING AND CHANNEL PRODUCTION

The **Control Room** runs in the browser: a multiview grid of every feed, TAKE/CUT between preview and program, scene layouts (single, picture-in-picture, two-up, splits, grid), lower-thirds, scrolling chyrons, clocks and logos as overlays, per-source audio mixing with a tally of what's hot.

Field streamers push cameras from a phone straight to your server. The director mixes in the browser; the program goes to your **PeerTube channel**; and when no one is live, the channel keeps broadcasting a scheduled playlist so the station never goes dark.

Auto-Director switches sources by priority rules — a streamer goes down, the next source fades in, the fallback slate catches anything else.

Distributed transcode workers: laptops and GPUs join as render workers over Tailscale — an RTX 3090 laptop is already transcoding real PeerTube jobs with hardware encoding, verified end to end. More workers join the pool as load rises.

THE BACKBONE

- ▶ **Encrypted vault** for every platform credential — the database never sees a plaintext key.
 - ▶ **Hash-chained audit log** — every editorial action tamper-evident, forever.
 - ▶ **DST-proof scheduling** — the occurrence engine survives clock changes; two scheduled posts that collapse onto the same instant fire exactly once.
 - ▶ **Media sandbox** — every FFmpeg job runs in a per-job jail: unprivileged, no network, resource-capped.
 - ▶ **An archive that can't be purged** — content-addressed, encrypted, deduplicated. Deleted threads stay citable; yesterday's footage is tomorrow's playlist.
 - ▶ **Teams without password-sharing** — roles per account, invite links, people can remove their own access when they move on.
 - ▶ **Self-hosted end to end** — no corporate cloud, no accounts needed to run the system itself.
-

WHAT WE'RE ASKING FOR

Testers. Take the whole flow for a spin with your real feeds and your real accounts: post through the approval flow, run the column wall, mix a stream in the Control Room, vote in a bridged chat, break the scheduler across timezones, try to break the archive. Every finding makes the next collective's deployment safer.

Intake hackers. The hard problem is pulling content from the locked platforms — hashtags, accounts, search columns on X, Instagram, TikTok — without burning accounts or exposing the collective. We have a verified architecture for the sanctioned sources (RSS, fediverse, Bluesky firehose, Telegram, PeerTube, YouTube RSS) and per-URL fetchers for single public posts. What we want is your design input and build help on: bridge services, community-submitted link intake, archive harvesters, and honest assessments of what is and isn't worth the risk. If you've built scrapers, run intake pipelines, or know the platform-integrity arms race from the inside — this is the problem to bring your scars to.

Stack if you want to hack: Next.js + NestJS + Postgres (RLS) + Temporal + Redis + S3 + FFmpeg/PeerTube. Self-hosted end to end, multi-tenant, everything audited.