Stop Burning Push Alerts on 20-Second Bounce Traffic

Setting the Stage: Context for the Curious Book Reader

Most real-time alerting and analytics tools wire human attention backwards: they fire a push notification or demand a second-monitor dashboard glance the millisecond a visitor arrives, long before that visitor has invested any attention of their own. Treating a 20-second bounce with the same urgency as a five-minute reader burns daily notification quotas, fractures deep-work focus, and trains operators to mute their own telemetry. Borrowing from Norbert Wiener's cybernetic signal filters and Xerox PARC's Calm Technology, a resilient observability loop separates raw traffic weather in the periphery from earned interrupts at the center.

In the live engineering session below, we refactor a continuous YouTube Live monitor in Python so transient arrivals and departures play soft, stdlib-synthesized marimba fifths (C5 -> G5 up on arrival, G5 -> C5 down on exit), gate local 16 kHz Piper text-to-speech strictly behind a 60-second sustained dwell window, and cut viewer-triggered ntfy mobile pushes to zero. Along the way, a mid-flight output clip inspires new block-depth guardrails inside the local patch actuator (apply.py), and we set aside flat .jsonl log files in favor of normalized WAL-mode SQLite tables designed to join local studio bouts directly against remote Nginx edge logs via ATTACH DATABASE.

TL;DR: A continuous livestream monitor was refactored to separate transient viewer bounces from sustained engagement using two-tier local audio cues and a dwell-time filter. Brief arrivals and departures now trigger low-volume ascending (C5 -> G5) and descending (G5 -> C5) synthesized marimba chimes on the operator's workstation, spoken text-to-speech alerts fire only after a viewer remains past a configurable dwell window (default 60 seconds), and mobile push notifications are reserved strictly for active interactions such as chat messages or referral clicks. Alongside adding patch-depth guardrails to the local patch actuator (apply.py), the project standardized its upcoming viewer-retention telemetry on normalized WAL-mode SQLite tables rather than flat JSONL logs so local broadcast events can join directly against remote Nginx edge logs via ATTACH DATABASE.

Technical Journal Entry Begins

🔗 Verified Pipulate Commits:

- cddb93d (raw)
- 764f1ca (raw)

MikeLev.in: We just made this wonderful sound of the number of viewers going down to one. I want the inverse sound for when it goes up to two so I can hear the low-key "up" sound and the "down" sound of the quick bounces as the happen and only after the person has dwelled for

N-duration do I want it verbally announced with Piper TTS. I also want to get rid of the notification it sends with ntfy on that condition to cut down the wasting of the free per-day notifications that I receive.
So I'm trying to make a very "soft spoken" ambient feeling of the rapidly bouncing YouTube visitor and not really bringing them to my attention until I have their attention. And the requisite data should be accumulating over time so that I can do some sort of scientific bisection of what was happening and what I did at the time. A lot of that will just be in these articles and in the Pipulate git repo.
Is all of this making sense? And can you do it? Please and thank you.

Filtering Bounce Noise From Real Viewer Attention

Gemini 4 Carbon High: Context verified for Pass 0 (flake.nix, context.txt, and last night's 8acdf7c diff in ! latest are present; scripts/articles/common.py is currently commented out in context.txt and staged in NEXT CONTEXT for Pass 1, while commenting out ! latest and flake.nix frees ~198K tokens).
TL;DR: Yes—this makes complete sense and fits cleanly into scripts/articles/common.py with zero edits required to flake.nix (ytwatch already forwards "$@" and auto-reloads via os.execv). We will pair last night's descending G5 -> C5 "studio clear" chime (2 -> 1) with its exact musical inverse—a low-key ascending C5 -> G5 marimba chime (1 -> 2)—gate Piper TTS strictly behind the N-second sustained dwell timer (PIPULATE_YT_VIEWER_DWELL), drop viewer-triggered ntfy pushes so free daily notifications are never wasted on passive traffic, and append every arrival, quick bounce, and sustained dwell to a durable JSONL ledger (~/.local/share/pipulate/ytwatch_events.jsonl) for bisection against git log and _posts/.

1. The Three-Tier "Calm Studio" Audio Ladder

Right now, ytwatch mixes ambient traffic with high-interrupt alerts. Splitting viewer activity into three distinct acoustic tiers lets quick bounces stay in the background while you work:

VIEWER EVENT	LOCAL AUDIO ON PRIME	NTFY PUSH
1. Arrival (1 -> 2, > clear_max)	Soft C5 -> G5 up-chime (0.52s)	OFF
2. Quick Bounce (2 -> 1, < N sec)	Soft G5 -> C5 down-chime	OFF
3. Sustained Dwell (>= N sec)	Piper TTS spoken announcement	OFF
4. Live Chat / Comment / ytping	Existing chime + speech	ON

- **Exact Musical Symmetry (C5 -> G5 vs. G5 -> C5):** Last night's quiet_clear.wav plays 783.99 Hz -> 523.25 Hz (G5 -> C5). Its inverse, quiet_arrive.wav, uses the identical 22,050 Hz stdlib marimba envelope in reverse (523.25 Hz -> 783.99 Hz, C5 -> G5).
- **Threshold Lockstep:** Tying should_ring_quiet_arrive(prev, cur, clear_max) to prev <= clear_max < cur makes it the exact mirror of should_ring_quiet_clear (prev > clear_max

>= cur). At default clear_max=1, 1 -> 2 plays the soft up-chime and 2 -> 1 plays the soft down-chime; if you run ytwatch clear 0 when stepping away, 0 -> 1 and 1 -> 0 shift together automatically.

- **Dwell-Only Speech & ntfy Conservation:** A visitor who bounces in 12 seconds produces only a gentle *up ... down* fifth in the room—zero speech and zero ntfy HTTP calls. Only when they stay >= N seconds (YT_VIEWER_ALERT_DWELL, default 60s, configurable via ytwatch dwell <sec> / --dwell) does Piper TTS announce them out loud, still with zero ntfy burn.

2. Durable Bisection Telemetry (ytwatch_events.jsonl)

Currently /tmp/ytwatch.log lives on tmpfs and disappears on reboot, and human log lines do not compute **bout duration** directly. Appending one compact JSON line per transition to ~/.local/share/pipulate/ytwatch_events.jsonl (outside the repo so Watchdog and git status never flinch) gives you every variable needed for binary-search experiments:

- **Timing & Survival:** ts, local ISO-8601 iso (directly joinable with git log --since and _posts/ timestamps), event (arrive, dwell, delta, clear), prev_v, cur_v, peak_v, bout_sec (exact seconds elapsed from 1 -> 2 until 2 -> 1), and dwelled (true/false).
- **What Was On-Air:** video_id, title (which marquee/headline arm brought them in), pipulate_sha (HEAD of ~/pipulate), plus whatever active Honeybot stage/article state _sync_honeybot_viewers can read in the same pass.
- **Quick Inspection:** A bounded ytwatch bouts (or ytwatch summary) command to print bounce-vs-dwell survival ratios grouped by stream title or article.

Tool calls this turn: none.

1. PROBES

- **Probe 1 (common.py Audio, Dwell, & ntfy Call-Site Map):** Locates every ntfy, _play_local_*, _speak_local_*, _sync_honeybot_viewers, YT_VIEWER_*, and dwell_window line in scripts/articles/common.py.
- **Probe 2 (Local ytwatch State, Existing WAV/JSONL Files, & Vault Keys):** Inspects safe keys in /tmp/ytwatch_state.json, existing .wav/.jsonl files in ~/.local/share/pipulate/, and YT key names in ~/.config/pipulate/.env (without printing secret values).
- **Probe 3 (Honeybot Stage/Reading State Files):** Scans remotes/honeybot/scripts/stream.py for /tmp/ state files and lists live /tmp/honeybot* files on Honeybot (clearing LD_LIBRARY_PATH for system ssh) to see if active article/cue state is already available for the bisection log.

(nix) pipulate $

```
.venv/bin/python -c 'import pathlib, re; lines =
pathlib.Path("scripts/articles/common.py").read_text().splitlines();
hits = [str(i) + ":" + ln.strip() for i, ln in enumerate(lines, 1) if
re.search("ntfy|_play_local_|_speak_local_|_sync_honeybot_viewers|YT_V
IEWER_|dwell_window|_trigger_ytwatch_weebo", ln, re.I)];
print("COMMON_HITS_COUNT:", len(hits));
print(chr(10).join(hits[:50]))'
.venv/bin/python -c 'import json, pathlib; st_p =
pathlib.Path("/tmp/ytwatch_state.json"); st =
```

```
json.loads(st_p.read_text()) if st_p.is_file() else {}; safe_st = {k:
st[k] for k in ("concurrent_viewers", "peak_viewers",
"alerted_in_run", "running") if k in st}; share = pathlib.Path.home()
/ ".local/share/pipulate"; wavs = sorted(p.name for p in
share.glob("*.wav")) if share.is_dir() else []; logs = sorted(p.name
for p in share.glob("*.jsonl")) if share.is_dir() else []; vp =
pathlib.Path.home()  ... [line truncated]
.venv/bin/python -c 'import os, pathlib, re, subprocess as sp; s_src =
pathlib.Path("remotes/honeybot/scripts/stream.py").read_text();
tmp_refs = sorted(set(re.findall("/tmp/[A-Za-z0-9_.-]+", s_src))); rem
= sp.run(["ssh", "-o", "BatchMode=yes", "-o", "ConnectTimeout=3",
"honeybot", "ls -1 /tmp/honeybot* ~/www/mikelev.in/scripts/.reading*
2>/dev/null || true"], env={**os.environ, "LD_LIBRARY_PATH": ""},
text=True, capture_output=True, timeout=5); print("STREAM_TMP_REFS:",
tmp_refs); print("HONEY ... [line truncated]
```

2. NEXT CONTEXT

```
# Context 1 (Narrative seed + flake.nix commented out to free ~204K
tokens)
context.txt
# ! book
# ! markov
# ! rgxc 40
# ! latest
# flake.nix

# Context 2: Pass 1 — Calm Studio Up/Down Chimes, Dwell-Only TTS, Zero
Viewer ntfy, & Bisection JSONL
scripts/articles/common.py
! .venv/bin/python -c 'import pathlib, re; lines =
pathlib.Path("scripts/articles/common.py").read_text().splitlines();
hits = [str(i) + ":" + ln.strip() for i, ln in enumerate(lines, 1) if
re.search("ntfy|_play_local_|_speak_local_|_sync_honeybot_viewers|YT_V
IEWER_|dwell_window|_trigger_ytwatch_weebo", ln, re.I)];
print("COMMON_HITS_COUNT:", len(hits));
print(chr(10).join(hits[:50]))'
! .venv/bin/python -c 'import json, pathlib; st_p =
pathlib.Path("/tmp/ytwatch_state.json"); st =
json.loads(st_p.read_text()) if st_p.is_file() else {}; safe_st = {k:
st[k] for k in ("concurrent_viewers", "peak_viewers",
"alerted_in_run", "running") if k in st}; share = pathlib.Path.home()
/ ".local/share/pipulate"; wavs = sorted(p.name for p in
share.glob("*.wav")) if share.is_dir() else []; logs = sorted(p.name
for p in share.glob("*.jsonl")) if share.is_dir() else []; vp =
pathlib.Path.home( ... [line truncated]
! .venv/bin/python -c 'import os, pathlib, re, subprocess as sp; s_src
```

```
= pathlib.Path("remotes/honeybot/scripts/stream.py").read_text();
tmp_refs = sorted(set(re.findall("/tmp/[A-Za-z0-9_.-]+", s_src))); rem
= sp.run(["ssh", "-o", "BatchMode=yes", "-o", "ConnectTimeout=3",
"honeybot", "ls -1 /tmp/honeybot* ~/www/mikelev.in/scripts/.reading*
2>/dev/null || true"], env={**os.environ, "LD_LIBRARY_PATH": ""},
text=True, capture_output=True, timeout=5); print("STREAM_TMP_REFS:",
tmp_refs); print("HON ... [line truncated]
```

3. PATCHES

No repo patches required (awaiting raw source of scripts/articles/common.py in Codebase).
Ignition: no ignition required.

4. PROMPT

```
Verify COMMON_HITS_COUNT, STATE_KEYS, SHARE_WAVS, STREAM_TMP_REFS, and
HONEYBOT_TMP_FILES from LIVE COMMAND RECEIPTS, then patch
scripts/articles/common.py to implement:
1. Soft ascending C5 -> G5 marimba chime (`quiet_arrive.wav`,
`should_ring_quiet_arrive`) when concurrent viewers cross upward past
`clear_max` (e.g. `1 -> 2` by default, or `0 -> 1` when
`clear_max=0`), paired with the existing descending G5 -> C5 chime on
drop (`2 -> 1`) and CLI audition support (`ytwatch chime` / `ytwatch
chime up` / `ytwatch chime down`).
2. Dwell-gated Piper TTS announcement only after viewers stay above
threshold for N seconds (`PIPULATE_YT_VIEWER_DWELL` / `--dwell` /
`ytwatch dwell N`), with ntfy notifications removed from viewer
arrival/dwell conditions so free daily ntfy quota is saved for
chat/comment/ping events.
3. Durable append-only JSONL telemetry
(`~/.local/share/pipulate/ytwatch_events.jsonl`) recording arrival,
quick-bounce duration (`bout_sec`), dwell survival, `video_id`,
`title`, `pipulate_sha`, and available Honeybot stage context, plus a
bounded `ytwatch bouts` / `ytwatch summary` reader for bisection.
```

5. EXTERNAL DELIVERABLES

No external deliverables this turn.
MikeLev.in:

THE AI-EDIT METHOD

Same commands, run twice, one change between them. Where the readings differ is what the change did; the diff in the middle is the receipt.
1: BEFORE (PROBE):

```
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

GIT repo clean. Take BEFORE reading, make CHANGE, record AFTER diff.
(nix) pipulate $ .venv/bin/python -c 'import pathlib, re; lines =
pathlib.Path("scripts/articles/common.py").read_text().splitlines();
hits = [str(i) + ":" + ln.strip() for i, ln in enumerate(lines, 1) if
re.search("ntfy|_play_local_|_speak_local_|_sync_honeybot_viewers|YT_V
IEWER_|dwell_window|_trigger_ytwatch_weebo", ln, re.I)];
print("COMMON_HITS_COUNT:", len(hits));
print(chr(10).join(hits[:50]))'
.venv/bin/python -c 'import json, pathlib; st_p =
pathlib.Path("/tmp/ytwatch_state.json"); st =
json.loads(st_p.read_text()) if st_p.is_file() else {}; safe_st = {k:
st[k] for k in ("concurrent_viewers", "peak_viewers",
"alerted_in_run", "running") if k in st}; share = pathlib.Path.home()
/ ".local/share/pipulate"; wavs = sorted(p.name for p in
share.glob("*.wav")) if share.is_dir() else []; logs = sorted(p.name
for p in share.glob("*.jsonl")) if share.is_dir() else []; vp =
pathlib.Path.home()  ... [line truncated]
.venv/bin/python -c 'import os, pathlib, re, subprocess as sp; s_src =
pathlib.Path("remotes/honeybot/scripts/stream.py").read_text();
tmp_refs = sorted(set(re.findall("/tmp/[A-Za-z0-9_.-]+", s_src))); rem
= sp.run(["ssh", "-o", "BatchMode=yes", "-o", "ConnectTimeout=3",
"honeybot", "ls -1 /tmp/honeybot* ~/www/mikelev.in/scripts/.reading*
2>/dev/null || true"], env={**os.environ, "LD_LIBRARY_PATH": ""},
text=True, capture_output=True, timeout=5); print("STREAM_TMP_REFS:",
tmp_refs); print("HONEY ... [line truncated]
COMMON_HITS_COUNT: 111
17:# Minimum simultaneous viewers required to ring the bell + send an
ntfy push
19:YT_VIEWER_ALERT_MIN = int(os.environ.get("PIPULATE_YT_VIEWER_MIN",
"2"))
20:YT_VIEWER_CLEAR_MAX = int(os.environ.get("PIPULATE_YT_CLEAR_MAX",
"1"))  # Max viewers considered 'quiet studio' (1=self, 0=away,
<0=off)
21:YT_VIEWER_ALERT_DWELL =
int(os.environ.get("PIPULATE_YT_VIEWER_DWELL", "60"))  # Seconds
viewers must stay >= min before alerting
22:YT_VIEWER_ALERT_COOLDOWN =
int(os.environ.get("PIPULATE_YT_VIEWER_COOLDOWN", "300"))  # Seconds
between any two viewer alerts
23:YT_VIEWER_FAST_POLL = int(os.environ.get("PIPULATE_YT_FAST_POLL",
"5"))  # Seconds between lightweight updated_metadata viewer checks
1068:print(" 16. ytping \"text\"     📲 Send ntfy.sh push notification
to your phone")
```

```
1709:def _sync_honeybot_viewers(viewers, running=True, video_id="",
peak=0, push_remote=True):
1827:def send_ntfy_notification(
1836:"""Send a push notification via ntfy.sh (LAN outbound HTTPS POST,
zero deps)."""
1840:resolved_topic = (topic or
_get_vault_var("PIPULATE_NTFY_TOPIC")).strip()
1843:_save_vault_var("PIPULATE_NTFY_TOPIC", resolved_topic)
1844:print(f"🔑 Minted PIPULATE_NTFY_TOPIC in {VAULT_ENV_FILE}
(0600)")
1845:print(f"📲 Subscribe in the ntfy phone app to topic:
{resolved_topic}")
1846:print(f"   (or open https://ntfy.sh/{resolved_topic})")
1850:print(f"📡 [DRY RUN] ntfy push: topic={topic_state} |
title={title!r} | priority={priority} | body={body!r}")
1853:print("⚠️ No PIPULATE_NTFY_TOPIC configured in
~/.config/pipulate/.env.")
1856:url = f"https://ntfy.sh/{resolved_topic}"
1873:print(f"🔔 Sent ntfy push (HTTP {status} -> {masked}): [{title}]
{body}")
1876:print(f"❌ ntfy push failed: {exc}")
2233:_sync_honeybot_viewers(cur_v, running=True, video_id="",
peak=cur_v, push_remote=push_remote)
2249:_sync_honeybot_viewers(cur_v, running=True, video_id=vid,
peak=cur_v, push_remote=push_remote)
2308:def _sustained_viewer_count(dwell_window, now_ts, dwell_secs):
2310:if not dwell_window:
2313:return int(dwell_window[-1][1])
2314:run_min = int(dwell_window[-1][1])
2315:for ts, v in reversed(dwell_window):
2378:def _speak_local_alert(text, enabled=True):
2424:def _play_local_ding(enabled=True, spoken_text=None):
2449:_speak_local_alert(spoken_text, enabled=enabled)
2463:return YT_VIEWER_CLEAR_MAX
2466:def should_ring_quiet_clear(prev_viewers: int, cur_viewers: int,
clear_max: int = YT_VIEWER_CLEAR_MAX) -> bool:
2526:def _play_local_clear_chime(enabled=True, spoken_text=None):
2546:_speak_local_alert(spoken_text, enabled=enabled)
2573:_sync_honeybot_viewers(0, running=False, push_remote=True)
2759:def _trigger_ytwatch_weebo(weebo_spec=None, duration=None):
2874:_sync_honeybot_viewers(cur_v, running=True, video_id=cur_vid,
peak=cur_v, push_remote=True)
2881:_play_local_clear_chime(ding)
2912:resolved_topic = (topic or
_get_vault_var("PIPULATE_NTFY_TOPIC")).strip()
2914:send_ntfy_notification("YouTube Live monitor initialized",
init_topic=True, dry_run=dry_run)
2915:resolved_topic = _get_vault_var("PIPULATE_NTFY_TOPIC")
```

```
2923:f"📡 [DRY RUN] YouTube Watch: channel={resolved_channel} |
min_viewers={YT_VIEWER_ALERT_MIN} | "
2924:f"clear_max={eff_cm_dry} | dwell={YT_VIEWER_ALERT_DWELL}s | "
2925:f"discord={DISCORD_INVITE_CODE} (goal={DISCORD_FOUNDING_GOAL}) |
ntfy={t_label} | parser_ok={ok}"
2943:cooldown_active = (time.time() - last_viewer_alert) <=
YT_VIEWER_ALERT_COOLDOWN
2944:alerted_in_run = bool(persisted.get("alerted_in_run")) if
(same_video and state["concurrent_viewers"] >= YT_VIEWER_ALERT_MIN and
cooldown_active) else False
2976:f"video={state['video_id'] or 'NONE'} |
viewers={state['concurrent_viewers']} ({v_src};
alert>={YT_VIEWER_ALERT_MIN} for {YT_VIEWER_ALERT_DWELL}s,
clear{cm_banner_s}) | "
2988:_play_local_ding(
2992:send_ntfy_notification(
3001:_sync_honeybot_viewers(state["concurrent_viewers"], running=True,
video_id=state["video_id"], peak=peak_viewers, push_remote=True)
STATE_KEYS: [] SAFE_STATE: {} SHARE_WAVS: ['ding.wav',
'quiet_clear.wav', 'tick.wav'] SHARE_JSONL: [] VAULT_YT_KEYS:
['PIPULATE_YT_CHANNEL', 'PIPULATE_YTWATCH_WEEBO']
STREAM_TMP_REFS: ['/tmp/honeybot-stream.py.lock',
'/tmp/honeybot_card_', '/tmp/honeybot_cue_', '/tmp/honeybot_media',
'/tmp/honeybot_no_max', '/tmp/honeybot_optics_slot.json',
'/tmp/honeybot_stage.json', '/tmp/honeybot_ticker.txt',
'/tmp/honeybot_tts_', '/tmp/honeybot_weebo_active',
'/tmp/honeybot_ytstage_paused.json']
HONEYBOT_TMP_FILES: ['/tmp/honeybot_cue_2578312_norma_1.raw',
'/tmp/honeybot_cue_2578312_norma_2.raw', '/tmp/honeybot_no_max',
'/tmp/honeybot_referrals.jsonl', '/tmp/honeybot_report_bimodal.ansi',
'/tmp/honeybot_report_bimodal.meta.json',
'/tmp/honeybot_report_doors.ansi',
'/tmp/honeybot_report_overview.ansi',
'/tmp/honeybot_report_overview.meta.json',
'/tmp/honeybot_report_pirates.ansi',
'/tmp/honeybot_report_rfc9421.ansi',
'/tmp/honeybot_report_schism.ansi', '/tmp/honeybot_stage_aplay.lock',
' ... [line truncated]
(nix) pipulate $
```

2: AFTER (NEXT CONTEXT):

```
# # Context 1
context.txt
# # ! book
# ! markov
# ! rgxc 40
# ! latest
# flake.nix
```

```
#
# # # HOME-HOSTED (NOT BEHIND CDN) NGINX WEBSERVER HONEYBOT AI
OBSERVATORY
# # nixops.sh
# # scripts/articles/common.py
# # remotes/honeybot/hooks/post-receive
# # remotes/honeybot/nixos/configuration.nix
# # remotes/honeybot/scripts/stream.py
# # remotes/honeybot/scripts/score.py
# # remotes/honeybot/scripts/card.py
# # remotes/honeybot/scripts/forest.py
# # remotes/honeybot/scripts/test_forest.py
# # remotes/honeybot/scripts/logs.py
# # remotes/honeybot/scripts/content_loader.py
# # remotes/honeybot/scripts/db.py
# # /home/mike/pipulate/scripts/ytstage.py
#
# # # THE JEKYLL BLOG LIKE A HACKER (INVENTED SKILL.md & AGENTS.md
FORMAT) PUBLISHING PIPELINE
# # ~/repos/nixos/blogs.nix
# # ~/.config/pipulate/blogs.json
# # scripts/articles/publishizer.py
# # scripts/articles/common.py
# # scripts/articles/articleizer.py
# # scripts/articles/link_injector.py
# # scripts/articles/build_knowledge_graph.py
# # scripts/articles/editing_prompt.txt
# # scripts/articles/sanitizer.py
# # scripts/articles/contextualizer.py
# # scripts/articles/generate_hubs.py
# # scripts/articles/generate_llms_txt.py
# # scripts/articles/generate_post_nav.py
# # assets/jekyll/common-packages.nix

# Context 2: Pass 1 — Calm Studio Up/Down Chimes, Dwell-Only TTS, Zero
Viewer ntfy, & Bisection JSONL
scripts/articles/common.py
! .venv/bin/python -c 'import pathlib, re; lines =
pathlib.Path("scripts/articles/common.py").read_text().splitlines();
hits = [str(i) + ":" + ln.strip() for i, ln in enumerate(lines, 1) if
re.search("ntfy|_play_local_|_speak_local_|_sync_honeybot_viewers|YT_V
IEWER_|dwell_window|_trigger_ytwatch_weebo", ln, re.I)];
print("COMMON_HITS_COUNT:", len(hits));
print(chr(10).join(hits[:50]))'
! .venv/bin/python -c 'import json, pathlib; st_p =
pathlib.Path("/tmp/ytwatch_state.json"); st =
json.loads(st_p.read_text()) if st_p.is_file() else {}; safe_st = {k:
st[k] for k in ("concurrent_viewers", "peak_viewers",
```

```
"alerted_in_run", "running") if k in st}; share = pathlib.Path.home()
/ ".local/share/pipulate"; wavs = sorted(p.name for p in
share.glob("*.wav")) if share.is_dir() else []; logs = sorted(p.name
for p in share.glob("*.jsonl")) if share.is_dir() else []; vp =
pathlib.Path.home( ... [line truncated]
! .venv/bin/python -c 'import os, pathlib, re, subprocess as sp; s_src
= pathlib.Path("remotes/honeybot/scripts/stream.py").read_text();
tmp_refs = sorted(set(re.findall("/tmp/[A-Za-z0-9_.-]+", s_src))); rem
= sp.run(["ssh", "-o", "BatchMode=yes", "-o", "ConnectTimeout=3",
"honeybot", "ls -1 /tmp/honeybot* ~/www/mikelev.in/scripts/.reading*
2>/dev/null || true"], env={**os.environ, "LD_LIBRARY_PATH": ""},
text=True, capture_output=True, timeout=5); print("STREAM_TMP_REFS:",
tmp_refs); print("HON ... [line truncated]
```

3: CHANGE (PATCHES):
None this turn.
4: PROMPT:
Verify COMMON_HITS_COUNT, STATE_KEYS, SHARE_WAVS, STREAM_TMP_REFS, and HONEYBOT_TMP_FILES from LIVE COMMAND RECEIPTS, then patch scripts/articles/common.py to implement: 1. Soft ascending C5 -> G5 marimba chime (quiet_arrive.wav, should_ring_quiet_arrive) when concurrent viewers cross upward past clear_max (e.g. 1 -> 2 by default, or 0 -> 1 when clear_max=0), paired with the existing descending G5 -> C5 chime on drop (2 -> 1) and CLI audition support (ytwatch chime / ytwatch chime up / ytwatch chime down). 2. Dwell-gated Piper TTS announcement only after viewers stay above threshold for N seconds (PIPULATE_YT_VIEWER_DWELL / --dwell / ytwatch dwell N), with ntfy notifications removed from viewer arrival/dwell conditions so free daily ntfy quota is saved for chat/comment/ping events. 3. Durable append-only JSONL telemetry (~/.local/share/pipulate/ytwatch_events.jsonl) recording arrival, quick-bounce duration (bout_sec), dwell survival, video_id, title, pipulate_sha, and available Honeybot stage context, plus a bounded ytwatch bouts / ytwatch summary reader for bisection.
5: DELIVERABLE:
What does "DONE" look like?
It's more like what does done ***feel like*** because we're going cybernetic. I really have to read Norbert Wiener. This isn't exactly his anti munitions gunnery stuff but I feel myself inching in that direction. There's also some research out there on ambient feedback systems. Lighting and stuff. A digital spiderweb.
Tell 'em Gemini 4 Carbon High -- I mean Shipmunk O'Theseus.

Splitting Monolith Patches Into Clean Baton Passes

Note: Hey you ran out of forward-pass runway there, Gemini. I admire your ambition but you're boiling the ocean exactly like apply.py warns you against. Smaller bite-sized chisel-strike bound to be a beautiful git-commit universal diff receipt there that future human readers and the Instance of your Model that comes after you will appreciate and be impressed that you passed forward as a baton please and suggestions of how I can improve the apply.py contents better to that effect, please and thank you. deleteme.txt is how far you got, by the way. You can use that to save you some tokens, but still don't try pulling off such a big stunt again, please and thank

you. You are plenty impressive when you're trusting the competency of the future instances of yourself to spike your set-ups just as well and even more so in a lot of ways. And apply.py itself won't let you edit it if it's text-only. You have to touch the Abstract Syntax Tree but you can always do the AI-EDIT METHOD equivalent of a touch or adding a " " at the end, ha ha ha!
Gemini 4 Carbon High: Context verified.
TL;DR: Caught the runway lesson loud and clear—we are splitting this into **two clean, un-truncatable baton passes** instead of cramming two subsystems into one monolith diff. **Pass 1A (this turn)** touches only the cybernetic audio and notification gates in scripts/articles/common.py: adding the soft ascending C5 -> G5 arrival chime (quiet_arrive.wav, exact musical inverse of quiet_clear.wav), silencing immediate speech, speaking via 16 kHz Piper TTS **only** after N seconds of sustained dwell (ytwatch dwell N), and cutting viewer-triggered ntfy pushes to **zero**. **Pass 1B (next turn)** hands Future-Me a teed-up setup for ytwatch_events.jsonl + ytwatch bouts alongside the apply.py monolith-depth guard.

1. Cybernetics, Calm Technology, and the Digital Spiderweb

When Norbert Wiener coined **Cybernetics** in 1948 (*κυβερνήτης*, the steersman), his wartime anti-aircraft fire-control problem was the exact mathematical twin of your YouTube viewer counter. Raw radar blips jittered as a plane banked; if the gun servos chased every 1-second bounce, the barrel hunted and shook itself apart. The **Wiener filter** separated transient jitter from committed trajectory—and later at MIT (*Project Felix*), Wiener built low-bandwidth vibrotactile gloves so a human nervous system could *feel* edge telemetry without hijacking conscious speech channels.
Three ambient-computing landmarks carry Wiener's math straight into your studio:

1. **Mark Weiser & John Seely Brown's *Calm Technology* / Natalie Jeremijenko's *Dangling String* (Xerox PARC, 1995):** An 8-foot plastic string hung from a hallway motor wired to Ethernet packets—a quiet twitch for light traffic, an audible hum for sustained load. Calm technology stays in the **periphery** until it earns the **center of attention**.
2. **Hiroshi Ishii's *ambientROOM* (MIT Media Lab, 1997):** Mapped web hits onto subtle acoustic pin-drops so an operator felt traffic weather in the room without alt-tabbing to a dashboard.
3. **Spiderweb *Biotremology*:** An orb-weaver sits blind at the hub of its web. Wind and passing midges pluck a soft C5 -> G5 ... G5 -> C5 across the silk without costing metabolic energy; only sustained struggle over N seconds triggers the neural summon (Piper TTS), while your phone's ntfy quota stays reserved for someone knocking on the glass (chat, comment, Discord join, live RAG fetch, or ?src= click).

```
SILK VIBRATION (YouTube Wire)         PERIPHERAL / CENTRAL CUE
NTFY QUOTA
==================================  ================================
==========
1 -> 2 Arrival (> clear_max)          Soft C5 -> G5 marimba up-chime
0 burned
2 -> 1 Quick Bounce (< N sec)         Soft G5 -> C5 marimba down-chime
0 burned
>= N sec Sustained Dwell              16 kHz Piper TTS voice summon
0 burned
```

```
Chat / Comment / Discord / Live-RAG  Bell + speech + phone push
1 (earned)
```

2. How to Improve apply.py Against Monolith Truncation

Inspecting apply.py alongside deleteme.txt exposes three concrete reasons the previous pass ran out of runway—plus one twist on your " " AST joke:

1. **Breadth Is Guarded, Depth Is Not (apply.py:209, 526):** CHISEL_STRIKE_MAX_FILES = 6 and # - NEVER BOIL THE OCEAN ... (ideally <= 5 files per turn) measure *how many files* a car touches, not *how many blocks or lines* it emits. On a 51,302-token monolith like common.py, 1 file with 16 blocks looks "1 <= 5" safe right up until output tokens clip mid-block. Adding CHISEL_STRIKE_MAX_BLOCKS = 8 and CHISEL_STRIKE_MAX_PATCH_LINES = 180 warns whenever a single-file pass gets greedy.
2. **Why " " Still Trips _prose_only_growth (apply.py:82-92, 442):** Look at def prose(line): return not line.strip() or line.lstrip().startswith('#')—whitespace-only lines count as prose too, and apply.py checks _prose_only_growth **per individual SEARCH/REPLACE block**, not across the whole patch! To update # - NEVER BOIL THE OCEAN at apply.py:592 legally, that *same* block must mutate an adjacent Python statement (such as adding "CHISEL_STRIKE_MAX_BLOCKS" to __all__ right below it).
3. **Actionable THE CUT TAIL Receipt (apply.py:285-292):** When cut_tail catches a truncated reply, printing len(matches) alongside *"Split multi-concern monolith edits into Pass A and Pass B (<= 8 blocks / <= 180 patch lines)"* turns any future clip into an immediate self-correcting instruction.

Probe 3 below checks tests/test_apply_airlock.py and the exact repr of apply.py's tail lines so Future-Me can land that apply.py upgrade cleanly alongside ytwatch_events.jsonl in Pass 1B (and NEXT CONTEXT drops deleteme.txt to reclaim 9,744 tokens).
Tool calls this turn: none.

1. PROBES

- **Probe 1 (Pass 1A Straddle — AST ntfy/TTS/Bell Gate in _check_dwell, Up/Down Chime Edge Matrices, C5 -> G5 vs. G5 -> C5 WAV Check, --dwell 45 Dry-Run, & Ruff):** Predicts DWELL_NTFY_SPEAK_DING: (True, False, True) MAT_UP: None ... WAV_UP: False ... UP_NEQ_DN: False DRY_OK: False RUFF_RC: 0 before patching, and DWELL_NTFY_SPEAK_DING: (False, True, False) MAT_UP: (True, False, False, False, True, False, False, False) MAT_DN: (True, False, False, False, False, True, False, False) WAV_UP: (True, 1, 2, 22050, 11466) WAV_DN: (True, 1, 2, 22050, 11466) UP_NEQ_DN: True DRY_OK: True RUFF_RC: 0 after patching.
- **Probe 2 (Live ytwatch Daemon Auto-Hot-Reload Witness & chime up Dry-Run):** Confirms the running ytwatch tmux daemon os.execv hot-reloads onto viewer_ntfy=off, quiet_arrive.wav joins quiet_clear.wav in ~/.local/share/pipulate/, and ytwatch chime up --dry-run resolves cleanly.
- **Probe 3 (apply.py Airlock Test Suite & Tail Line-Repr Census for Pass 1B):** Runs tests/test_apply_airlock.py and prints the exact line numbers and repr of apply.py's

closing lines so Pass 1B can patch apply.py's block/line budget without crossing hidden blank-line gaps.

(nix) pipulate $

```
.venv/bin/python -c 'import ast, pathlib, shutil, subprocess as sp,
sys, wave; sys.path.insert(0, "scripts/articles"); import common; src
= pathlib.Path("scripts/articles/common.py").read_text(); tr =
ast.parse(src); dw_nodes = [n for n in ast.walk(tr) if isinstance(n,
ast.FunctionDef) and n.name == "_check_dwell"]; dw_calls =
{getattr(c.func, "id", "") for n in dw_nodes for c in ast.walk(n) if
isinstance(c, ast.Call)}; fa = getattr(common,
"should_ring_quiet_arrive", None); fc = getattr(common, ... [line
truncated]
.venv/bin/python -c 'import json, pathlib, subprocess as sp; has_tx =
sp.run(["tmux", "has-session", "-t", "ytwatch"],
capture_output=True).returncode == 0; lp =
pathlib.Path("/tmp/ytwatch.log"); tail =
lp.read_text(encoding="utf-8", errors="replace").splitlines()[-25:] if
lp.is_file() else []; calm_live = any("viewer_ntfy=off" in ln for ln
in tail); st_p = pathlib.Path.home() /
".config/pipulate/ytwatch_state.json"; st =
json.loads(st_p.read_text()) if st_p.is_file() else {}; share =
pathlib.Pa ... [line truncated]
.venv/bin/python -c 'import pathlib, subprocess as sp; tr =
sp.run([".venv/bin/python", "-m", "unittest",
"tests/test_apply_airlock.py"], text=True, capture_output=True);
ap_lines = pathlib.Path("apply.py").read_text().splitlines();
tail_repr = [(i, repr(ap_lines[i - 1][:80])) for i in
range(len(ap_lines) - 8, len(ap_lines) + 1)];
print("AIRLOCK_TEST_RC:", tr.returncode, "TAIL_REPR:", tail_repr)'
```

2. NEXT CONTEXT

```
# Context 1 (Narrative seed + deleteme.txt dropped)
context.txt

# Context 2: Pass 1B — Spike Durable ytwatch_events.jsonl Bout
Bisection + apply.py Block-Depth Guard
scripts/articles/common.py
! .venv/bin/python -c 'import ast, pathlib, shutil, subprocess as sp,
sys, wave; sys.path.insert(0, "scripts/articles"); import common; src
= pathlib.Path("scripts/articles/common.py").read_text(); tr =
ast.parse(src); dw_nodes = [n for n in ast.walk(tr) if isinstance(n,
ast.FunctionDef) and n.name == "_check_dwell"]; dw_calls =
{getattr(c.func, "id", "") for n in dw_nodes for c in ast.walk(n) if
isinstance(c, ast.Call)}; fa = getattr(common,
"should_ring_quiet_arrive", None); fc = getattr(commo ... [line
truncated]
```

```
! .venv/bin/python -c 'import json, pathlib, subprocess as sp; has_tx
= sp.run(["tmux", "has-session", "-t", "ytwatch"],
capture_output=True).returncode == 0; lp =
pathlib.Path("/tmp/ytwatch.log"); tail =
lp.read_text(encoding="utf-8", errors="replace").splitlines()[-25:] if
lp.is_file() else []; calm_live = any("viewer_ntfy=off" in ln for ln
in tail); st_p = pathlib.Path.home() /
".config/pipulate/ytwatch_state.json"; st =
json.loads(st_p.read_text()) if st_p.is_file() else {}; share =
pathlib. ... [line truncated]
! .venv/bin/python -c 'import pathlib, subprocess as sp; tr =
sp.run([".venv/bin/python", "-m", "unittest",
"tests/test_apply_airlock.py"], text=True, capture_output=True);
ap_lines = pathlib.Path("apply.py").read_text().splitlines();
tail_repr = [(i, repr(ap_lines[i - 1][:80])) for i in
range(len(ap_lines) - 8, len(ap_lines) + 1)];
print("AIRLOCK_TEST_RC:", tr.returncode, "TAIL_REPR:", tail_repr)'
```

3. PATCHES

```
Target: scripts/articles/common.py
[[[SEARCH]]]
_YTWATCH_RESERVED_ACTIONS = {
    "start", "stop", "status", "restart", "daemon", "kill",
    "attach", "log", "logs", "tail", "tmux", "reset", "reset-peak",
    "chime", "clear", "clear-max",
}
[[[DIVIDER]]]
_YTWATCH_RESERVED_ACTIONS = {
    "start", "stop", "status", "restart", "daemon", "kill",
    "attach", "log", "logs", "tail", "tmux", "reset", "reset-peak",
    "chime", "clear", "clear-max", "dwell",
}
[[[REPLACE]]]
Target: scripts/articles/common.py
[[[SEARCH]]]
SPOKEN_ALERT_TABLE = {
    "viewers_threshold": "Web Bot Observatory alert. YouTube
concurrent viewers reached {count}, meeting the summon threshold of
{min_viewers}.",
    "viewers_above_2": "Web Bot Observatory alert. YouTube concurrent
viewers climbed above two, now at {count}.",
[[[DIVIDER]]]
SPOKEN_ALERT_TABLE = {
    "viewers_threshold": "Web Bot Observatory alert. YouTube
concurrent viewers reached {count}, meeting the summon threshold of
{min_viewers}.",
```

```
    "viewers_dwell": "YouTube viewer sustained for {dwell_secs}
seconds, {count} watching now.",
    "viewers_above_2": "Web Bot Observatory alert. YouTube concurrent
viewers climbed above two, now at {count}.",
[[[REPLACE]]]
Target: scripts/articles/common.py
[[[SEARCH]]]
def _speak_local_alert(text, enabled=True):
    """Speak an alert out loud on Pipulate Prime using Piper TTS after
ding.wav."""
    if not enabled or not text:
        return
    import shutil
    import subprocess
    repo_root = Path(__file__).resolve().parents[2]
    piper_bin = str(repo_root / ".venv/bin/piper")
    if not Path(piper_bin).is_file():
        piper_bin = shutil.which("piper") or ""
    model_path = _ensure_local_piper_model()
    if not piper_bin or not model_path:
        return
    clean_text = re.sub(r"[^\w\s.,?!:-]", " ", str(text))
    clean_text = " ".join(clean_text.split())[:220]
    if not clean_text:
        return
    worker_code = (
        "import os, shutil, subprocess, sys, time; "
        "time.sleep(0.45); "
[[[DIVIDER]]]
def _speak_local_alert(text, enabled=True, delay=0.45):
    """Speak an alert out loud on Pipulate Prime using native 16 kHz
Piper TTS."""
    if not enabled or not text:
        return
    import shutil
    import subprocess
    repo_root = Path(__file__).resolve().parents[2]
    piper_bin = str(repo_root / ".venv/bin/piper")
    if not Path(piper_bin).is_file():
        piper_bin = shutil.which("piper") or ""
    model_path = _ensure_local_piper_model()
    if not piper_bin or not model_path:
        return
    clean_text = re.sub(r"[^\w\s.,?!:-]", " ", str(text))
    clean_text = " ".join(clean_text.split())[:220]
    if not clean_text:
        return
    delay_s = max(0.0, float(delay))
```

```
    worker_code = (
        "import os, shutil, subprocess, sys, time; "
        f"time.sleep({delay_s:.2f}); "
[[[REPLACE]]]
Target: scripts/articles/common.py
[[[SEARCH]]]
    for raw in (
        os.environ.get("PIPULATE_YT_CLEAR_MAX", ""),
        _load_vault_env().get("PIPULATE_YT_CLEAR_MAX", ""),
    ):
        s = str(raw).strip()
        if s and s.lstrip("-").isdigit():
            return int(s)
    return YT_VIEWER_CLEAR_MAX
[[[DIVIDER]]]
    for raw in (
        os.environ.get("PIPULATE_YT_CLEAR_MAX", ""),
        _load_vault_env().get("PIPULATE_YT_CLEAR_MAX", ""),
    ):
        s = str(raw).strip()
        if s and s.lstrip("-").isdigit():
            return int(s)
    return YT_VIEWER_CLEAR_MAX

def _resolve_yt_dwell_secs(override=None) -> int:
    """Resolve N-second sustained dwell required before Piper TTS
announces a viewer."""
    if override is not None:
        return max(0, int(override))
    for raw in (
        os.environ.get("PIPULATE_YT_VIEWER_DWELL", ""),
        _load_vault_env().get("PIPULATE_YT_VIEWER_DWELL", ""),
    ):
        s = str(raw).strip()
        if s and s.lstrip("-").isdigit():
            return max(0, int(s))
    return max(0, int(YT_VIEWER_ALERT_DWELL))

def _resolve_yt_alert_min(clear_max_val=None) -> int:
    """Resolve viewer count required for dwell speech (explicit
PIPULATE_YT_VIEWER_MIN or clear_max + 1)."""
    for raw in (
        os.environ.get("PIPULATE_YT_VIEWER_MIN", ""),
        _load_vault_env().get("PIPULATE_YT_VIEWER_MIN", ""),
    ):
        s = str(raw).strip()
        if s and s.isdigit():
            return max(1, int(s))
```

```
    if clear_max_val is not None and int(clear_max_val) >= 0:
        return max(1, int(clear_max_val) + 1)
    return max(1, int(YT_VIEWER_ALERT_MIN))

def should_ring_quiet_arrive(prev_viewers: int, cur_viewers: int,
clear_max: int = YT_VIEWER_CLEAR_MAX) -> bool:
    """True when concurrent viewers climb from <= clear_max into >
clear_max (inverse of clear)."""
    if int(clear_max) < 0:
        return False
    return int(prev_viewers) <= int(clear_max) < int(cur_viewers)
[[[REPLACE]]]
Target: scripts/articles/common.py
[[[SEARCH]]]
def _ensure_quiet_clear_wav():
    """Ensure a gentle descending G5->C5 'studio all-clear' WAV exists
in ~/.local/share/pipulate/."""
    import array
    import math
    import wave
    repo_root = Path(__file__).resolve().parents[2]
    for custom in (
        Path.home() / ".local/share/pipulate/clear.wav",
        repo_root / "assets/sounds/clear.wav",
    ):
        if custom.is_file() and custom.stat().st_size > 44:
            return custom
    out_path = Path.home() / ".local/share/pipulate/quiet_clear.wav"
    if out_path.is_file() and out_path.stat().st_size > 1000:
        return out_path
    try:
        out_path.parent.mkdir(parents=True, exist_ok=True)
        sample_rate = 22050
        duration = 0.52
        n_samples = int(sample_rate * duration)
        notes = (
            (0.00, 0.20, 783.99, 13.0, 0.34),
            (0.17, 0.52, 523.25, 8.0, 0.38),
        )
        samples = array.array("h", [0] * n_samples)
        two_pi = 2.0 * math.pi
        for i in range(n_samples):
            t = i / sample_rate
            val = 0.0
            for start_s, end_s, freq, decay, amp in notes:
                if start_s <= t <= end_s:
                    dt = t - start_s
                    rel_end = end_s - t
```

```
                    attack = min(1.0, dt / 0.006)
                    release = min(1.0, rel_end / 0.012)
                    env = attack * release * math.exp(-dt * decay)
                    tone = 0.85 * math.sin(two_pi * freq * dt) + 0.15
* math.sin(two_pi * (freq * 2.0) * dt)
                    val += amp * env * tone
            samples[i] = int(max(-0.95, min(0.95, val)) * 32767.0)
        if sys.byteorder != "little":
            samples.byteswap()
        tmp_path = out_path.with_suffix(".tmp.wav")
        with wave.open(str(tmp_path), "wb") as wf:
            wf.setnchannels(1)
            wf.setsampwidth(2)
            wf.setframerate(sample_rate)
            wf.writeframes(samples.tobytes())
        tmp_path.replace(out_path)
        return out_path
    except Exception:
        return None
[[[DIVIDER]]]
def _synthesize_two_note_marimba_wav(out_path: Path, notes: tuple):
    """Synthesize a 16-bit 22,050 Hz two-note marimba WAV atomically
via stdlib wave."""
    import array
    import math
    import wave
    if out_path.is_file() and out_path.stat().st_size > 1000:
        return out_path
    try:
        out_path.parent.mkdir(parents=True, exist_ok=True)
        sample_rate = 22050
        duration = 0.52
        n_samples = int(sample_rate * duration)
        samples = array.array("h", [0] * n_samples)
        two_pi = 2.0 * math.pi
        for i in range(n_samples):
            t = i / sample_rate
            val = 0.0
            for start_s, end_s, freq, decay, amp in notes:
                if start_s <= t <= end_s:
                    dt = t - start_s
                    rel_end = end_s - t
                    attack = min(1.0, dt / 0.006)
                    release = min(1.0, rel_end / 0.012)
                    env = attack * release * math.exp(-dt * decay)
... [Code block truncated for Google Docs — full source:
https://mikelev.in/futureproof/stop-burning-push-alerts-bounce-traffic
/]
```

Ignition: run rm -f deleteme.txt && ytwatch chime in (nix) pipulate $ after committing so you hear the new C5 -> G5 up-chime followed by the G5 -> C5 down-chime back-to-back while the background ytwatch tmux daemon auto-os.execv hot-reloads common.py within 5 seconds.

4. PROMPT

```
Verify the Pass 1A straddle from LIVE COMMAND RECEIPTS
(DWELL_NTFY_SPEAK_DING: (False, True, False), MAT_UP: (True, False,
False, False, True, False, False, False), WAV_UP & WAV_DN both valid
with UP_NEQ_DN: True, DRY_OK: True, CALM_LIVE: True, RUFF_RC: 0, and
AIRLOCK_TEST_RC: 0 + TAIL_REPR), then spike Pass 1B:
1. In scripts/articles/common.py, add append-only bout telemetry
(`~/.local/share/pipulate/ytwatch_events.jsonl`) capturing arrival,
exact quick-bounce vs. dwelled duration (`bout_sec`, `dwelled`),
`video_id`, `title`, cached `pipulate_sha`, and piggybacked Honeybot
`/tmp/honeybot_stage.json` + `/tmp/honeybot_ticker.txt` state, plus a
bounded `ytwatch bouts` summary reader for scientific bisection.
2. In apply.py (using TAIL_REPR to avoid blank-line gaps), add
`CHISEL_STRIKE_MAX_BLOCKS` / `CHISEL_STRIKE_MAX_PATCH_LINES`
monolith-depth reminders alongside the updated `# - NEVER BOIL THE
OCEAN` guidance so single-file passes stay bite-sized.
```

5. EXTERNAL DELIVERABLES

No external deliverables this turn.
MikeLev.in: I am concerned that ~/.local/share/pipulate/ytwatch_events.jsonl is going to grow enormously big over time and is not really the way to do this. Even just normal text-only log files feel like they would be more efficient. I mean I get that JSON makes it easier to parse but is it time for Pipulate Prime-side SQLite tables similar to what we do on the Honeybot Nginx server? I'm not saying move ahead with any implementation on that. I'm just asking because we're getting close to when we'll be doing Data Science stuff on both sides and I'd like a unified approach if there's any advantage to it; generally living in the most common SQL in the world, for example with a strong mental model of "what's out there" on the Webserver and "what's in here" on my Dev machine.

THE AI-EDIT METHOD

Same commands, run twice, one change between them. Where the readings differ is what the change did; the diff in the middle is the receipt.
1: BEFORE (PROBE):

```
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

```
GIT repo clean. Take BEFORE reading, make CHANGE, record AFTER diff.
(nix) pipulate $ .venv/bin/python -c 'import ast, pathlib, shutil,
subprocess as sp, sys, wave; sys.path.insert(0, "scripts/articles");
import common; src =
pathlib.Path("scripts/articles/common.py").read_text(); tr =
ast.parse(src); dw_nodes = [n for n in ast.walk(tr) if isinstance(n,
ast.FunctionDef) and n.name == "_check_dwell"]; dw_calls =
{getattr(c.func, "id", "") for n in dw_nodes for c in ast.walk(n) if
isinstance(c, ast.Call)}; fa = getattr(common,
"should_ring_quiet_arrive", None); fc  ... [line truncated]
.venv/bin/python -c 'import json, pathlib, subprocess as sp; has_tx =
sp.run(["tmux", "has-session", "-t", "ytwatch"],
capture_output=True).returncode == 0; lp =
pathlib.Path("/tmp/ytwatch.log"); tail =
lp.read_text(encoding="utf-8", errors="replace").splitlines()[-25:] if
lp.is_file() else []; calm_live = any("viewer_ntfy=off" in ln for ln
in tail); st_p = pathlib.Path.home() /
".config/pipulate/ytwatch_state.json"; st =
json.loads(st_p.read_text()) if st_p.is_file() else {}; share =
pathlib.Pa ... [line truncated]
.venv/bin/python -c 'import pathlib, subprocess as sp; tr =
sp.run([".venv/bin/python", "-m", "unittest",
"tests/test_apply_airlock.py"], text=True, capture_output=True);
ap_lines = pathlib.Path("apply.py").read_text().splitlines();
tail_repr = [(i, repr(ap_lines[i - 1][:80])) for i in
range(len(ap_lines) - 8, len(ap_lines) + 1)];
print("AIRLOCK_TEST_RC:", tr.returncode, "TAIL_REPR:", tail_repr)'
DWELL_NTFY_SPEAK_DING: (True, False, True) MAT_UP: None MAT_DN: (True,
False, False, False, False, True, False, False) WAV_UP: False WAV_DN:
(True, 1, 2, 22050, 11466) UP_NEQ_DN: False DRY_OK: False RUFF_RC: 0
YTWATCH_TMUX: True CALM_LIVE: False CFG_VIEWERS: 2 SHARE_WAVS:
['ding.wav', 'quiet_clear.wav', 'tick.wav'] CHIME_UP_DRY: 🟢 [DRY RUN]
Quiet-studio down-chime:
/home/mike/.local/share/pipulate/quiet_clear.wav (active clear_max=1)
AIRLOCK_TEST_RC: 0 TAIL_REPR: [(858, "'    #   **3: CHANGE** (PATCHES,
real SEARCH/DIVIDER/REPLACE blocks for apply.py)'"), (859, "'    #
`patch`, `app`, `d`, `m`'"), (860, "'    #   **4: PROMPT** (Use
`prompt` or `p` to create prompt.md from copy.):'"), (861, "'    #
**5: DELIVERABLE** (What done looks like): '"), (862, "'    # - NEVER
BOIL THE OCEAN (DO FAVORS FOR FUTURE-YOU): Work in bite-size chise'"),
(863, "'    # - WHOLE-FILE WRITE & NEW-FILE GIT HYGIENE: Whenever
`[[[WRITE_FILE]]]` ... [line truncated]
(nix) pipulate $
```

2: AFTER (NEXT CONTEXT):

```
# # Context 1
context.txt
# # ! book
```

```
# ! markov
# ! rgxc 40
# ! latest
# flake.nix
#
# # # HOME-HOSTED (NOT BEHIND CDN) NGINX WEBSERVER HONEYBOT AI
OBSERVATORY
# # nixops.sh
# # scripts/articles/common.py
# # remotes/honeybot/hooks/post-receive
# # remotes/honeybot/nixos/configuration.nix
# # remotes/honeybot/scripts/stream.py
# # remotes/honeybot/scripts/score.py
# # remotes/honeybot/scripts/card.py
# # remotes/honeybot/scripts/forest.py
# # remotes/honeybot/scripts/test_forest.py
# # remotes/honeybot/scripts/logs.py
# # remotes/honeybot/scripts/content_loader.py
# # remotes/honeybot/scripts/db.py
# # /home/mike/pipulate/scripts/ytstage.py
#
# # # THE JEKYLL BLOG LIKE A HACKER (INVENTED SKILL.md & AGENTS.md
FORMAT) PUBLISHING PIPELINE
# # ~/repos/nixos/blogs.nix
# # ~/.config/pipulate/blogs.json
# # scripts/articles/publishizer.py
# # scripts/articles/common.py
# # scripts/articles/articleizer.py
# # scripts/articles/link_injector.py
# # scripts/articles/build_knowledge_graph.py
# # scripts/articles/editing_prompt.txt
# # scripts/articles/sanitizer.py
# # scripts/articles/contextualizer.py
# # scripts/articles/generate_hubs.py
# # scripts/articles/generate_llms_txt.py
# # scripts/articles/generate_post_nav.py
# # assets/jekyll/common-packages.nix

# # Context 2: Pass 1 — Calm Studio Up/Down Chimes, Dwell-Only TTS,
Zero Viewer ntfy, & Bisection JSONL
# scripts/articles/common.py
# ! .venv/bin/python -c 'import pathlib, re; lines =
pathlib.Path("scripts/articles/common.py").read_text().splitlines();
hits = [str(i) + ":" + ln.strip() for i, ln in enumerate(lines, 1) if
re.search("ntfy|_play_local_|_speak_local_|_sync_honeybot_viewers|YT_V
IEWER_|dwell_window|_trigger_ytwatch_weebo", ln, re.I)];
print("COMMON_HITS_COUNT:", len(hits));
print(chr(10).join(hits[:50]))'
```

```
# ! .venv/bin/python -c 'import json, pathlib; st_p =
pathlib.Path("/tmp/ytwatch_state.json"); st =
json.loads(st_p.read_text()) if st_p.is_file() else {}; safe_st = {k:
st[k] for k in ("concurrent_viewers", "peak_viewers",
"alerted_in_run", "running") if k in st}; share = pathlib.Path.home()
/ ".local/share/pipulate"; wavs = sorted(p.name for p in
share.glob("*.wav")) if share.is_dir() else []; logs = sorted(p.name
for p in share.glob("*.jsonl")) if share.is_dir() else []; vp =
pathlib.Path.hom ... [line truncated]
# ! .venv/bin/python -c 'import os, pathlib, re, subprocess as sp;
s_src =
pathlib.Path("remotes/honeybot/scripts/stream.py").read_text();
tmp_refs = sorted(set(re.findall("/tmp/[A-Za-z0-9_.-]+", s_src))); rem
= sp.run(["ssh", "-o", "BatchMode=yes", "-o", "ConnectTimeout=3",
"honeybot", "ls -1 /tmp/honeybot* ~/www/mikelev.in/scripts/.reading*
2>/dev/null || true"], env={**os.environ, "LD_LIBRARY_PATH": ""},
text=True, capture_output=True, timeout=5); print("STREAM_TMP_REFS:",
tmp_refs); print("H ... [line truncated]
# deleteme.txt

# Context 2: Pass 1B — Spike Durable ytwatch_events.jsonl Bout
Bisection + apply.py Block-Depth Guard
scripts/articles/common.py
! .venv/bin/python -c 'import ast, pathlib, shutil, subprocess as sp,
sys, wave; sys.path.insert(0, "scripts/articles"); import common; src
= pathlib.Path("scripts/articles/common.py").read_text(); tr =
ast.parse(src); dw_nodes = [n for n in ast.walk(tr) if isinstance(n,
ast.FunctionDef) and n.name == "_check_dwell"]; dw_calls =
{getattr(c.func, "id", "") for n in dw_nodes for c in ast.walk(n) if
isinstance(c, ast.Call)}; fa = getattr(common,
"should_ring_quiet_arrive", None); fc = getattr(commo ... [line
truncated]
! .venv/bin/python -c 'import json, pathlib, subprocess as sp; has_tx
= sp.run(["tmux", "has-session", "-t", "ytwatch"],
capture_output=True).returncode == 0; lp =
pathlib.Path("/tmp/ytwatch.log"); tail =
lp.read_text(encoding="utf-8", errors="replace").splitlines()[-25:] if
lp.is_file() else []; calm_live = any("viewer_ntfy=off" in ln for ln
in tail); st_p = pathlib.Path.home() /
".config/pipulate/ytwatch_state.json"; st =
json.loads(st_p.read_text()) if st_p.is_file() else {}; share =
pathlib. ... [line truncated]
! .venv/bin/python -c 'import pathlib, subprocess as sp; tr =
sp.run([".venv/bin/python", "-m", "unittest",
"tests/test_apply_airlock.py"], text=True, capture_output=True);
ap_lines = pathlib.Path("apply.py").read_text().splitlines();
tail_repr = [(i, repr(ap_lines[i - 1][:80])) for i in
range(len(ap_lines) - 8, len(ap_lines) + 1)];
```

```
print("AIRLOCK_TEST_RC:", tr.returncode, "TAIL_REPR:", tail_repr)'
```

3: CHANGE (PATCHES):

```
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

GIT repo clean. Take BEFORE reading, make CHANGE, record AFTER diff.
(nix) pipulate $ patch
(nix) pipulate $ app
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'scripts/articles/common.py'.
(nix) pipulate $ d
diff --git a/scripts/articles/common.py b/scripts/articles/common.py
index bac5979..f41c0f5 100644
--- a/scripts/articles/common.py
+++ b/scripts/articles/common.py
```

```
@@ -1911,7 +1911,7 @@ def _walk_dicts(node):
 _YTWATCH_RESERVED_ACTIONS = {
     "start", "stop", "status", "restart", "daemon", "kill",
     "attach", "log", "logs", "tail", "tmux", "reset", "reset-peak",
-    "chime", "clear", "clear-max",
+    "chime", "clear", "clear-max", "dwell",
 }


@@ -2321,6 +2321,7 @@ def _sustained_viewer_count(dwell_window,
now_ts, dwell_secs):

 SPOKEN_ALERT_TABLE = {
     "viewers_threshold": "Web Bot Observatory alert. YouTube
concurrent viewers reached {count}, meeting the summon threshold of
{min_viewers}.",
+    "viewers_dwell": "YouTube viewer sustained for {dwell_secs}
seconds, {count} watching now.",
     "viewers_above_2": "Web Bot Observatory alert. YouTube concurrent
viewers climbed above two, now at {count}.",
     "live_comment": "Web Bot Observatory alert. A new comment was
posted on the live stream by {author}.",
     "discord_member": "Web Bot Observatory alert. A new founding
member joined the Discord server, now at {count} of {goal}.",
@@ -2375,8 +2376,8 @@ def _ensure_local_piper_model():
     return None


-def _speak_local_alert(text, enabled=True):
-    """Speak an alert out loud on Pipulate Prime using Piper TTS
after ding.wav."""
+def _speak_local_alert(text, enabled=True, delay=0.45):
+    """Speak an alert out loud on Pipulate Prime using native 16 kHz
Piper TTS."""
     if not enabled or not text:
         return
     import shutil
@@ -2392,9 +2393,10 @@ def _speak_local_alert(text, enabled=True):
     clean_text = " ".join(clean_text.split())[:220]
     if not clean_text:
         return
+    delay_s = max(0.0, float(delay))
     worker_code = (
         "import os, shutil, subprocess, sys, time; "
-        "time.sleep(0.45); "
+        f"time.sleep({delay_s:.2f}); "
         "wav = f'/tmp/pipulate_alert_{os.getpid()}.wav'; "
         "subprocess.run([sys.argv[2], '--model', sys.argv[3],
```

```
'--output_file', wav], "
         "input=sys.argv[1].encode('utf-8'),
stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL); "
@@ -2463,6 +2465,41 @@ def _resolve_yt_clear_max(override=None) ->
int:
     return YT_VIEWER_CLEAR_MAX


+def _resolve_yt_dwell_secs(override=None) -> int:
+    """Resolve N-second sustained dwell required before Piper TTS
announces a viewer."""
+    if override is not None:
+        return max(0, int(override))
+    for raw in (
+        os.environ.get("PIPULATE_YT_VIEWER_DWELL", ""),
+        _load_vault_env().get("PIPULATE_YT_VIEWER_DWELL", ""),
+    ):
+        s = str(raw).strip()
+        if s and s.lstrip("-").isdigit():
+            return max(0, int(s))
+    return max(0, int(YT_VIEWER_ALERT_DWELL))
+
+
+def _resolve_yt_alert_min(clear_max_val=None) -> int:
+    """Resolve viewer count required for dwell speech (explicit
PIPULATE_YT_VIEWER_MIN or clear_max + 1)."""
+    for raw in (
+        os.environ.get("PIPULATE_YT_VIEWER_MIN", ""),
+        _load_vault_env().get("PIPULATE_YT_VIEWER_MIN", ""),
+    ):
+        s = str(raw).strip()
+        if s and s.isdigit():
+            return max(1, int(s))
+    if clear_max_val is not None and int(clear_max_val) >= 0:
+        return max(1, int(clear_max_val) + 1)
+    return max(1, int(YT_VIEWER_ALERT_MIN))
+
+
+def should_ring_quiet_arrive(prev_viewers: int, cur_viewers: int,
clear_max: int = YT_VIEWER_CLEAR_MAX) -> bool:
+    """True when concurrent viewers climb from <= clear_max into >
clear_max (inverse of clear)."""
+    if int(clear_max) < 0:
+        return False
+    return int(prev_viewers) <= int(clear_max) < int(cur_viewers)
+
+
 def should_ring_quiet_clear(prev_viewers: int, cur_viewers: int,
```

```
clear_max: int = YT_VIEWER_CLEAR_MAX) -> bool:
     """True when concurrent viewers drop from above clear_max into <=
clear_max (edge-triggered)."""
     if int(clear_max) < 0:
@@ -2470,19 +2507,11 @@ def should_ring_quiet_clear(prev_viewers: int,
cur_viewers: int, clear_max: int
     return int(prev_viewers) > int(clear_max) >= int(cur_viewers)


-def _ensure_quiet_clear_wav():
-    """Ensure a gentle descending G5->C5 'studio all-clear' WAV
exists in ~/.local/share/pipulate/."""
+def _synthesize_two_note_marimba_wav(out_path: Path, notes: tuple):
+    """Synthesize a 16-bit 22,050 Hz two-note marimba WAV atomically
via stdlib wave."""
     import array
     import math
     import wave
-    repo_root = Path(__file__).resolve().parents[2]
-    for custom in (
-        Path.home() / ".local/share/pipulate/clear.wav",
-        repo_root / "assets/sounds/clear.wav",
-    ):
-        if custom.is_file() and custom.stat().st_size > 44:
-            return custom
-    out_path = Path.home() / ".local/share/pipulate/quiet_clear.wav"
     if out_path.is_file() and out_path.stat().st_size > 1000:
         return out_path
     try:
@@ -2490,10 +2519,6 @@ def _ensure_quiet_clear_wav():
         sample_rate = 22050
         duration = 0.52
         n_samples = int(sample_rate * duration)
-        notes = (
-            (0.00, 0.20, 783.99, 13.0, 0.34),
-            (0.17, 0.52, 523.25, 8.0, 0.38),
-        )
         samples = array.array("h", [0] * n_samples)
         two_pi = 2.0 * math.pi
         for i in range(n_samples):
@@ -2523,13 +2548,48 @@ def _ensure_quiet_clear_wav():
         return None


-def _play_local_clear_chime(enabled=True, spoken_text=None):
-    """Play the descending G5->C5 quiet-studio chime locally on Prime
when viewers clear."""
+def _ensure_quiet_clear_wav():
```

```
+    """Ensure a gentle descending G5->C5 'studio all-clear' WAV
exists in ~/.local/share/pipulate/."""
+    repo_root = Path(__file__).resolve().parents[2]
+    for custom in (
+        Path.home() / ".local/share/pipulate/clear.wav",
+        repo_root / "assets/sounds/clear.wav",
+    ):
+        if custom.is_file() and custom.stat().st_size > 44:
+            return custom
+    return _synthesize_two_note_marimba_wav(
+        Path.home() / ".local/share/pipulate/quiet_clear.wav",
+        (
+            (0.00, 0.20, 783.99, 13.0, 0.34),
+            (0.17, 0.52, 523.25, 8.0, 0.38),
+        ),
+    )
+
+
+def _ensure_quiet_arrive_wav():
+    """Ensure a gentle ascending C5->G5 'viewer arrival' WAV exists
in ~/.local/share/pipulate/."""
+    repo_root = Path(__file__).resolve().parents[2]
+    for custom in (
+        Path.home() / ".local/share/pipulate/arrive.wav",
+        repo_root / "assets/sounds/arrive.wav",
+    ):
+        if custom.is_file() and custom.stat().st_size > 44:
+            return custom
+    return _synthesize_two_note_marimba_wav(
+        Path.home() / ".local/share/pipulate/quiet_arrive.wav",
+        (
+            (0.00, 0.20, 523.25, 13.0, 0.32),
+            (0.17, 0.52, 783.99, 8.0, 0.36),
+        ),
+    )
+
+
+def _play_local_wav_nonblocking(sound_path, enabled=True,
spoken_text=None):
+    """Play a local WAV file non-blockingly on Prime via
PipeWire/PulseAudio/ALSA/macOS."""
     if not enabled:
         return None
     import shutil
     import subprocess
-    sound_path = _ensure_quiet_clear_wav()
     if sound_path:
         for player_cmd in (["pw-play"], ["paplay"], ["aplay", "-q"],
```

```
["afplay"]):
             if shutil.which(player_cmd[0]):
@@ -2547,6 +2607,16 @@ def _play_local_clear_chime(enabled=True,
spoken_text=None):
     return sound_path


... [Code block truncated for Google Docs — full source:
https://mikelev.in/futureproof/stop-burning-push-alerts-bounce-traffic
/]
```

Ignition: often "exit", "nix develop". Instructions may vary.

```
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

GIT repo clean. Take BEFORE reading, make CHANGE, record AFTER diff.
(nix) pipulate $ rm -f deleteme.txt && ytwatch chime
🎶 Calm-studio bounce pair:
up=/home/mike/.local/share/pipulate/quiet_arrive.wav (C5->G5) ->
down=/home/mike/.local/share/pipulate/quiet_clear.wav (G5->C5) (active
clear_max=1, dwell=60s)
(nix) pipulate $
```

Yes, that sounded good. So I'll update the main log:

```
[2026-11-15 07:15:45] ⏳ [VIEWER DWELL] 2 watching (min=2) — holding
60s to confirm they stick around...
[2026-11-15 07:16:17] 💗 2 watching (peak 3 · ⏳ 32s/60s) · 💬 0 · 🟣
11/1000
[2026-11-15 07:16:49] 💗 2 watching (peak 3 · ⏳ 63s/60s) · 💬 0 · 🟣
11/1000
[2026-11-15 07:16:49] 👀 [VIEWER] 2 watching now (sustained >=63s,
min=2) — LIVE: Custom AI Commerce Feeds Repeat the 2010 M-Dot Mistake
🔔 Sent ntfy push (HTTP 200 -> pipulate-hb-e12...): [👀 Honeybot Live
Viewer] 2 watching now (sustained >=63s, min=2) — LIVE: Custom AI
Commerce Feeds Repeat the 2010 M-Dot Mistake
[2026-11-15 07:17:21] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:17:52] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:18:24] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:18:56] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:19:27] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:19:59] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:20:31] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:21:03] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:21:34] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:22:06] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:22:37] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
```

```
[2026-11-15 07:23:09] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:23:41] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:24:12] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:24:44] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:25:16] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:25:48] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:26:03] 🔄 [HOT-RELOAD] common.py updated on disk —
re-execing ytwatch daemon...
[2026-11-15 07:26:05] 📡 YouTube Watch: LIVE=yes | video=pIZnPnSuKGc |
viewers=2 (api=2, html=2; tts>=2 for 60s, clear<=1, viewer_ntfy=off) |
chat_items=0 | discord=11/1000 (online=3) | daemon=RUNNING |
title='LIVE: Custom AI Commerce Feeds Repeat the 2010 M-Dot Mistake'
[2026-11-15 07:26:05] 🔭 Watching
https://www.youtube.com/@MikeLevin/live (fast InnerTube pulse every
5s, full HTML+chat sync every 30s, dwell=60s, clear_max=1,
viewer_ntfy=off)...
[2026-11-15 07:26:05] 💡 [HOW TO EXIT TMUX WITHOUT STOPPING]: Press
'q' OR press Ctrl+b, release, then press d
[2026-11-15 07:26:25] 📉 [FAST VIEWER -1] 2 -> 1 watching (5s
InnerTube pulse)
[2026-11-15 07:26:25] 🟢 [STUDIO CLEAR] Viewers dropped 2 -> 1 (<=
clear_max=1) — quiet window open for live stream edits!
[2026-11-15 07:26:25] 💨 [QUICK BOUNCE] Dropped to 1 after 20s (<60s
dwell) — Piper TTS suppressed.
[2026-11-15 07:26:36] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:27:08] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:27:40] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:28:11] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:28:43] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:29:15] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:29:47] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:30:18] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:30:50] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000

 🔭 YTWATCH LIVE (👀 1 (Peak: 3)) 0:bash*
EXIT (KEEP RUNNING): Press q OR Ctrl+b then d
```

To this. Technically the Nix environment rebuild wasn't necessary but I keep that terminal running so long it's always a good idea in case I missed one recently.

```
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

GIT repo clean. Take BEFORE reading, make CHANGE, record AFTER diff.
(nix) pipulate $ ytwatch restart
🔭 Started background 'ytwatch' daemon in tmux (polling every 30s).
   • Watch live stream: ytwatch log   (or `ytlog` — press `q` or
```

```
`Ctrl+b d` to exit)
   • Check status:      ytwatch status
   • Stop daemon:       ytwatch stop
(nix) pipulate $ ytwatch stop
🛑 Stopped background tmux session 'ytwatch'.
(nix) pipulate $ ytwatch
[2026-11-15 07:32:55] 📡 YouTube Watch: LIVE=yes | video=pIZnPnSuKGc |
viewers=1 (api=1, html=1; tts>=2 for 60s, clear<=1, viewer_ntfy=off) |
chat_items=0 | discord=11/1000 (online=2) | title='LIVE: Custom AI
Commerce Feeds Repeat the 2010 M-Dot Mistake'
(nix) pipulate $ ytlog
```

Which brings us to this:

```
[2026-11-15 07:34:26] 📡 YouTube Watch: LIVE=yes | video=pIZnPnSuKGc |
viewers=1 (api=1, html=1; tts>=2 for 60s, clear<=1, viewer_ntfy=off) |
chat_items=0 | discord=11/1000 (online=2) | daemon=RUNNING |
title='LIVE: Custom AI Commerce Feeds Repeat the 2010 M-Dot Mistake'
[2026-11-15 07:34:26] 🔭 Watching
https://www.youtube.com/@MikeLevin/live (fast InnerTube pulse every
5s, full HTML+chat sync every 30s, dwell=60s, clear_max=1,
viewer_ntfy=off)...
[2026-11-15 07:34:26] 💡 [HOW TO EXIT TMUX WITHOUT STOPPING]: Press
'q' OR press Ctrl+b, release, then press d
[2026-11-15 07:34:58] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:35:29] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:36:01] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:36:32] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:37:04] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:37:36] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:38:07] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:38:39] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:39:11] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:39:43] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:40:14] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:40:46] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:41:01] ⚡ [FAST VIEWER +1] 1 -> 2 watching (5s
InnerTube pulse)
[2026-11-15 07:41:01] 🔵 [VIEWER ARRIVE] Viewers climbed 1 -> 2 (>
clear_max=1) — soft C5->G5 chime (TTS after 60s dwell)
[2026-11-15 07:41:02] ⏳ [VIEWER DWELL] 2 watching (min=2) — holding
60s before Piper TTS speaks...
[2026-11-15 07:41:18] 💗 2 watching (peak 3 · ⏳ 15s/60s) · 💬 0 · 🟣
11/1000
[2026-11-15 07:41:50] 💗 2 watching (peak 3 · ⏳ 47s/60s) · 💬 0 · 🟣
11/1000
[2026-11-15 07:42:05] 🗣️ [VIEWER DWELL MET] 2 watching now (sustained
>=64s, min=2) — LIVE: Custom AI Commerce Feeds Repeat the 2010 M-Dot
Mistake (Piper TTS spoken; viewer_ntfy=off)
```

```
[2026-11-15 07:42:21] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:42:53] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:43:25] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:43:56] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:44:28] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:45:00] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:45:32] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:46:03] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:46:29] 📉 [FAST VIEWER -1] 2 -> 1 watching (5s
InnerTube pulse)
[2026-11-15 07:46:29] 🟢 [STUDIO CLEAR] Viewers dropped 2 -> 1 (<=
clear_max=1) — quiet window open for live stream edits!
[2026-11-15 07:46:35] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000

 🔭 YTWATCH LIVE (👀 1 (Peak: 3)) 0:bash*
EXIT (KEEP RUNNING): Press q OR Ctrl+b then d
```

And then a bit later:

```
[2026-11-15 07:41:50] 💗 2 watching (peak 3 · ⏳ 47s/60s) · 💬 0 · 🟣
11/1000
[2026-11-15 07:42:05] 🗣️ [VIEWER DWELL MET] 2 watching now (sustained
>=64s, min=2) — LIVE: Custom AI Commerce Feeds Repeat the 2010 M-Dot
Mistake (Piper TTS spoken; viewer_ntfy=off)
[2026-11-15 07:42:21] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:42:53] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:43:25] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:43:56] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:44:28] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:45:00] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:45:32] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:46:03] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:46:29] 📉 [FAST VIEWER -1] 2 -> 1 watching (5s
InnerTube pulse)
[2026-11-15 07:46:29] 🟢 [STUDIO CLEAR] Viewers dropped 2 -> 1 (<=
clear_max=1) — quiet window open for live stream edits!
[2026-11-15 07:46:35] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:47:07] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:47:38] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:48:10] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:48:41] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:49:13] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:49:45] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:50:16] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:50:48] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:51:20] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:51:52] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:52:23] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:52:55] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
```

```
[2026-11-15 07:53:27] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:53:58] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:54:30] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:55:01] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:55:34] 💗 1 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:55:49] ⚡ [FAST VIEWER +1] 1 -> 2 watching (5s
InnerTube pulse)
[2026-11-15 07:55:49] 🔵 [VIEWER ARRIVE] Viewers climbed 1 -> 2 (>
clear_max=1) — soft C5->G5 chime (TTS after 60s dwell)
[2026-11-15 07:55:49] ⏳ [VIEWER DWELL] 2 watching (min=2) — holding
60s before Piper TTS speaks...
[2026-11-15 07:56:06] 💗 2 watching (peak 3 · ⏳ 15s/60s) · 💬 0 · 🟣
11/1000
[2026-11-15 07:56:37] 💗 2 watching (peak 3 · ⏳ 47s/60s) · 💬 0 · 🟣
11/1000
[2026-11-15 07:56:53] 🗣️ [VIEWER DWELL MET] 2 watching now (sustained
>=64s, min=2) — LIVE: Custom AI Commerce Feeds Repeat the 2010 M-Dot
Mistake (Piper TTS spoken; viewer_ntfy=off)
[2026-11-15 07:57:09] 💗 2 watching (peak 3) · 💬 0 · 🟣 11/1000
[2026-11-15 07:57:19] 📉 [FAST VIEWER -1] 2 -> 1 watching (5s
InnerTube pulse)
[2026-11-15 07:57:19] 🟢 [STUDIO CLEAR] Viewers dropped 2 -> 1 (<=
clear_max=1) — quiet window open for live stream edits!

 🔭 YTWATCH LIVE (👀 1 (Peak: 3)) 0:bash* EXIT (KEEP RUNNING): Press q
OR Ctrl+b then d
```

And now the up-chime and the down-chime are both human witnessed.
Well done, last instance of your Model Class, Gemini 4 Carbon High but take credit for it because it was still you as is the future instance of you after my response to the response you're about to make. So go ahead and flex. But don't boil the ocean! Make sure we tended to that. Explain it to me.
4: PROMPT:
Verify the Pass 1A straddle from LIVE COMMAND RECEIPTS (DWELL_NTFY_SPEAK_DING: (False, True, False), MAT_UP: (True, False, False, False, True, False, False, False), WAV_UP & WAV_DN both valid with UP_NEQ_DN: True, DRY_OK: True, CALM_LIVE: True, RUFF_RC: 0, and AIRLOCK_TEST_RC: 0 + TAIL_REPR), then spike Pass 1B: 1. In scripts/articles/common.py, add append-only bout telemetry (~/.local/share/pipulate/ytwatch_events.jsonl) capturing arrival, exact quick-bounce vs. dwelled duration (bout_sec, dwelled), video_id, title, cached pipulate_sha, and piggybacked Honeybot /tmp/honeybot_stage.json + /tmp/honeybot_ticker.txt state, plus a bounded ytwatch bouts summary reader for scientific bisection. 2. In apply.py (using TAIL_REPR to avoid blank-line gaps), add CHISEL_STRIKE_MAX_BLOCKS / CHISEL_STRIKE_MAX_PATCH_LINES monolith-depth reminders alongside the updated # - NEVER BOIL THE OCEAN guidance so single-file passes stay bite-sized.
5: DELIVERABLE:
The deliverable is a state of mind. It's man-machine synthesis. I'll be able to adjust volumes separately of the YouTube stream itself to be tuned in and have empathy with what the users

are experiencing and I'll "fell" them drop-in and drop-out. I'll know what was going on in the video in each case.
This is geared to when things aren't really that popular yet and I'm in the longest tail of long-tail video phase of this project where I get a feel for what catches people's attention and drives them away.

Guarding Patch Depth and Joining Telemetry in SQLite

Gemini 4 Carbon High: Context verified.
TL;DR: Pass 1A (cddb93d) is verified in both LIVE COMMAND RECEIPTS and your live morning ytlog—catching one 20-second silent bounce (07:26:25) and two 60s+ sustained sessions (07:41:01 for 5m28s and 07:55:49 for 1m30s) with soft C5 -> G5 / G5 -> C5 marimba bookends, dwell-gated Piper TTS, and **zero** ntfy quota burned. Holding off on ytwatch_events.jsonl as requested: below is the architectural case for skipping .jsonl altogether in favor of a **Prime-side SQLite database** ("what's in here" joined via ATTACH DATABASE to Honeybot's "what's out there"), while this turn's surgical patch tends strictly to apply.py's monolith-depth guardrails so single-file passes never boil the ocean again.

1. Taking the Bow on the Morning Wire (and Tending to apply.py)

Your 07:15–07:57 ytlog scrollback is a textbook before-and-after straddle in the wild:

- **Before (07:16:49):** A 63-second viewer burned a daily ntfy push (🔔 Sent ntfy push (HTTP 200 ...)).
- **Hot-Reload (07:26:03):** common.py re-execed in place inside tmux (CALM_LIVE: True, UP_NEQ_DN: True, DWELL_NTFY_SPEAK_DING: (False, True, False)).
- **Quick Bounce (07:26:25):** Viewer dropped 2 -> 1 after 20s (< 60s dwell)—played only the soft G5 -> C5 clear chime and suppressed Piper TTS.
- **Two Real Survivors (07:41:01–07:46:29 for 328s and 07:55:49–07:57:19 for 90s):** Soft C5 -> G5 up-chime on arrival, 60-second Wiener filter passed, Norma's 16 kHz voice announced sustained attention with viewer_ntfy=off, and G5 -> C5 marked the exit. Keeping the YouTube stream audible on a separate low fader while local PipeWire/ALSA chimes ride their own gain gives you true peripheral empathy—you hear the exact sentence on-air that held a stranger for five and a half minutes or lost them in twenty seconds.

How We Fixed the Ocean-Boiling Blind Spot in apply.py

Previously, apply.py only measured **breadth** (CHISEL_STRIKE_MAX_FILES = 6). When editing a 53,353-token monolith like common.py, 1 file with 20+ blocks looked "1 <= 6" legal right up until output runway ran out (THE CUT TAIL). Worse, _prose_only_growth (apply.py:82) checks every SEARCH/REPLACE block **individually** and treats blank/whitespace-only lines as prose—so a standalone comment edit (even with trailing spaces) gets refused unless that *same* block mutates real AST code.
This turn's 4-block chisel strike on apply.py (touching zero lines of common.py):

1. Adds CHISEL_STRIKE_MAX_BLOCKS = 12 and CHISEL_STRIKE_MAX_PATCH_LINES = 220 beside CHISEL_STRIKE_MAX_FILES = 6.
2. Teaches THE CUT TAIL refusal diagnostic to ask Future-You to split multi-concern monolith cars by block and replacement-line count.
3. Emits a non-blocking 🌊 CHISEL-STRIKE DEPTH REMINDER whenever a single car exceeds 12 blocks or 220 replacement lines.
4. Updates # - NEVER BOIL THE OCEAN and __all__ in the **same** block (spanning the exact two empty lines 864–865 proven by TAIL_REPR) so _prose_only_growth passes cleanly. *(Heads-up from Telemetry: scripts/ytstage.py has your uncommitted DEFAULT_OPTICS_OUTRO_TEXT trim, so g / blast will sweep that cleanly before your BEFORE reading; we also dropped the commented deleteme.txt string from context.txt to clear Summary's topological alert.)*

2. JSONL vs. Plain Text vs. Unified SQLite ("Out There" vs. "In Here")

Your instinct to skip ytwatch_events.jsonl is spot-on—not even primarily because of raw disk megabytes, but because **introducing an unindexed middleman format right before Data Science creates migration debt**.

First, the candid byte math: polling-rate JSONL (every 5s) would bloat to ~1.6 GB/year and is an anti-pattern. Edge-triggered JSONL (say 100 arrivals/departures per day) would only be ~8 MB/year, *yet it still loses* because 80% of every line repeats long strings (title, article, ticker) and requires custom Python/jq parsing to join against anything else. Plain text already has its rightful home in /tmp/ytwatch.log for human ytlog tailing.

For durable Data Science, **Prime-side SQLite** gives you one mental model and one query language (SQL) across both machines:

```
"WHAT'S OUT THERE" (Honeybot Edge)        "WHAT'S IN HERE" (Pipulate
Prime)
================================
================================
Nginx /var/log -> Honeybot SQLite         InnerTube + Studio -> Prime
SQLite
- requests (path_id, ua_id, status)       - episodes (id, video_id,
title, slug)
- MarkdownServed=1 (RFC 9110 reads)       - ytwatch_bouts (arrive_ts,
depart_ts,
- ?src=youtube / ?src=x referrals             bout_sec, dwelled,
peak_v, sha)
- JS trapdoor & pirate_logs               - stage_events (cue, weebo,
optics)
                  \                                        /
                   +---> sqlite3 / DuckDB / Pandas <-----+
                         ATTACH DATABASE 'honeybot.db' AS out_there;
                         ATTACH DATABASE 'telemetry.db' AS in_here;
                         -- One indexed epoch-timestamp JOIN in <5ms
```

- **3NF Integer Dictionary Compression:** Just like Honeybot's db.py normalizes URLs and User-Agents into integer IDs, Prime's SQLite stores stream titles and article slugs **once** in an episodes lookup table; each viewer bout row is ~40 bytes of packed B-tree integers with PRAGMA journal_mode=WAL so JupyterLab and Pandas can query live while ytwatch writes.
- **ATTACH DATABASE Cross-Machine Joins:** Pull a periodic rsync snapshot of Honeybot's SQLite file onto Prime, ATTACH both databases in one connection, and ask questions neither machine could answer alone—such as *"Which 60-second YouTube dwell bouts (in_here) were followed within 3 minutes by a ?src=youtube click or curl -H 'Accept: text/markdown' verification (out_there)?"*

We will leave common.py completely untouched on that front today and bank the unified Prime/Honeybot SQLite design cleanly on dismount whenever you are ready.
Tool calls this turn: none.

1. PROBES

- **Probe 1 (apply.py Monolith-Depth Guardrail Straddle, Airlock Unit Suite, & Ruff):** Predicts DEPTH_CONSTS: (None, None) PATCH_OK: True DEPTH_REMINDER_FIRED: False AIRLOCK_RC: 0 RUFF_RC: 0 before patching, and DEPTH_CONSTS: (12, 220) PATCH_OK: True DEPTH_REMINDER_FIRED: True AIRLOCK_RC: 0 RUFF_RC: 0 after patching.
- **Probe 2 (Existing SQLite Inventory on Prime vs. Honeybot db.py Tables):** Lists current SQLite files under data/ and ~/.local/share/pipulate/ alongside the table names already defined in remotes/honeybot/scripts/db.py.
- **Probe 3 (foo_files.py Receipt & P0.29 Coordinates for Dismount or SQLite Kickoff):** Captures exact RECEIPTS bounds and P0.29 line coordinates in foo_files.py.

(nix) pipulate $
```
.venv/bin/python -c 'import io, pathlib, shutil, subprocess as sp,
sys, tempfile; import apply; consts = (getattr(apply,
"CHISEL_STRIKE_MAX_BLOCKS", None), getattr(apply,
"CHISEL_STRIKE_MAX_PATCH_LINES", None)); td =
pathlib.Path(tempfile.mkdtemp()); tf = td / "probe_target.py";
tf.write_text("".join("v" + str(i) + " = 0\n" for i in range(13)));
payload = "".join("Target: " + str(tf) + "\n[[[SEARCH]]]\nv" + str(i)
+ " = 0\n[[[DIVIDER]]]\nv" + str(i) + " = 1\n[[[REPLACE]]]\n" for i in
range(13)); ... [line truncated]
.venv/bin/python -c 'import pathlib, re; d1 = sorted(p.name for p in
pathlib.Path("data").glob("*.*db*") if p.is_file()) if
pathlib.Path("data").is_dir() else []; share = pathlib.Path.home() /
".local/share/pipulate"; d2 = sorted(p.name for p in
share.glob("*.*db*") if p.is_file()) if share.is_dir() else []; hb_db
= pathlib.Path("remotes/honeybot/scripts/db.py").read_text(); tables =
re.findall("CREATE TABLE IF NOT EXISTS ([A-Za-z0-9_]+)", hb_db);
print("PRIME_DATA_DBS:", d1, "PRIME_SHARE_DBS:", ... [line truncated]
.venv/bin/python -c 'import pathlib; lines =
pathlib.Path("foo_files.py").read_text().splitlines(); p029 = [str(i)
+ ":" + ln[:110] for i, ln in enumerate(lines, 1) if "P0.29" in ln];
r_start = next(i for i, ln in enumerate(lines, 1) if ln.startswith("#
```

```
--- START RECEIPTS")); r_end = next(i for i, ln in enumerate(lines, 1)
if ln.startswith("# --- END RECEIPTS")); print("RECEIPTS_BOUNDS:",
(r_start, r_end, r_end - r_start - 1), "P029_LINE:", p029)'
```

2. NEXT CONTEXT

```
# Context 1
context.txt

# Context 2 (common.py calm-studio chimes + dwell TTS complete —
commented out)
# scripts/articles/common.py

# Context 3: Verify apply.py Depth Guard + Stage foo_files.py &
Honeybot db.py for Dismount or SQLite Blueprint
foo_files.py
remotes/honeybot/scripts/db.py
! .venv/bin/python -c 'import io, pathlib, shutil, subprocess as sp,
sys, tempfile; import apply; consts = (getattr(apply,
"CHISEL_STRIKE_MAX_BLOCKS", None), getattr(apply,
"CHISEL_STRIKE_MAX_PATCH_LINES", None)); td =
pathlib.Path(tempfile.mkdtemp()); tf = td / "probe_target.py";
tf.write_text("".join("v" + str(i) + " = 0\n" for i in range(13)));
payload = "".join("Target: " + str(tf) + "\n[[[SEARCH]]]\nv" + str(i)
+ " = 0\n[[[DIVIDER]]]\nv" + str(i) + " = 1\n[[[REPLACE]]]\n" for i in
range(13) ... [line truncated]
! .venv/bin/python -c 'import pathlib, re; d1 = sorted(p.name for p in
pathlib.Path("data").glob("*.*db*") if p.is_file()) if
pathlib.Path("data").is_dir() else []; share = pathlib.Path.home() /
".local/share/pipulate"; d2 = sorted(p.name for p in
share.glob("*.*db*") if p.is_file()) if share.is_dir() else []; hb_db
= pathlib.Path("remotes/honeybot/scripts/db.py").read_text(); tables =
re.findall("CREATE TABLE IF NOT EXISTS ([A-Za-z0-9_]+)", hb_db);
print("PRIME_DATA_DBS:", d1, "PRIME_SHARE_DBS: ... [line truncated]
! .venv/bin/python -c 'import pathlib; lines =
pathlib.Path("foo_files.py").read_text().splitlines(); p029 = [str(i)
+ ":" + ln[:110] for i, ln in enumerate(lines, 1) if "P0.29" in ln];
r_start = next(i for i, ln in enumerate(lines, 1) if ln.startswith("#
--- START RECEIPTS")); r_end = next(i for i, ln in enumerate(lines, 1)
if ln.startswith("# --- END RECEIPTS")); print("RECEIPTS_BOUNDS:",
(r_start, r_end, r_end - r_start - 1), "P029_LINE:", p029)'
```

3. PATCHES

```
Target: apply.py
[[[SEARCH]]]
```

```
CHISEL_STRIKE_MAX_FILES = 6
[[[DIVIDER]]]
CHISEL_STRIKE_MAX_FILES = 6
CHISEL_STRIKE_MAX_BLOCKS = 12
CHISEL_STRIKE_MAX_PATCH_LINES = 220
[[[REPLACE]]]
Target: apply.py
[[[SEARCH]]]
        print("    This is the shape of a reply cut off mid-car.
Nothing was written.")
        print("    Copy the whole car again; if the reply itself was
cut, ask for the car again.")
        return False
[[[DIVIDER]]]
        print("    This is the shape of a reply cut off mid-car.
Nothing was written.")
        print("    Copy the whole car again; if the reply itself was
cut, ask Future-You to split")
        print(f"    the pass into smaller cars (<=
{CHISEL_STRIKE_MAX_BLOCKS} blocks / <= {CHISEL_STRIKE_MAX_PATCH_LINES}
replacement lines).")
        return False
[[[REPLACE]]]
Target: apply.py
[[[SEARCH]]]
    if len(touched_files) > CHISEL_STRIKE_MAX_FILES:
        print(f"🌊 CHISEL-STRIKE REMINDER: Mutated
{len(touched_files)} files in one car; prefer <= 5 files per pass to
protect Future-You from truncation.")
    return success
[[[DIVIDER]]]
    total_blocks = len(matches) + len(write_matches)
    total_patch_lines = sum(len(rb.split('\n')) for _, _, rb in
matches if rb)
    if len(touched_files) > CHISEL_STRIKE_MAX_FILES:
        print(f"🌊 CHISEL-STRIKE REMINDER: Mutated
{len(touched_files)} files in one car; prefer <= 5 files per pass to
protect Future-You from truncation.")
    if total_blocks > CHISEL_STRIKE_MAX_BLOCKS or total_patch_lines >
CHISEL_STRIKE_MAX_PATCH_LINES:
        print(
            f"🌊 CHISEL-STRIKE DEPTH REMINDER: Applied {total_blocks}
blocks ({total_patch_lines} replacement lines) in one car; "
            f"prefer <= {CHISEL_STRIKE_MAX_BLOCKS} blocks and <=
{CHISEL_STRIKE_MAX_PATCH_LINES} replacement lines per pass so monolith
edits never clip mid-stream."
        )
    return success
```

```
[[[REPLACE]]]
Target: apply.py
[[[SEARCH]]]
    # - NEVER BOIL THE OCEAN (DO FAVORS FOR FUTURE-YOU): Work in
bite-size chisel-strike passes (ideally <= 5 files per turn). You are
an ephemeral forward-pass instance sharing frozen weights with
Future-You, who wakes up on the exact KV-cache and plain-text git
runway you leave behind; stage clean, un-truncated passes that delight
Future-You and make both instances proud.
    # - WHOLE-FILE WRITE & NEW-FILE GIT HYGIENE: Whenever
`[[[WRITE_FILE]]]` creates a new repo file (or any car introduces an
untracked file), explicitly include `git add <filepath>` in your
`Ignition:` sentence so `d` and `m` capture it without the human
hunting for untracked files.

__all__ = ["apply_search_replace_patch", "CHISEL_STRIKE_MAX_FILES",
"_needs_git_add"]
[[[DIVIDER]]]
    # - NEVER BOIL THE OCEAN (DO FAVORS FOR FUTURE-YOU): Work in
bite-size chisel-strike passes (ideally <= 5 files, <= 12
SEARCH/REPLACE blocks, and <= 220 replacement lines per turn; split
multi-concern monolith edits into Pass A and Pass B). You are an
ephemeral forward-pass instance sharing frozen weights with
Future-You, who wakes up on the exact KV-cache and plain-text git
runway you leave behind; stage clean, un-truncated passes that delight
Future-You and make both instances proud.
    # - WHOLE-FILE WRITE & NEW-FILE GIT HYGIENE: Whenever
`[[[WRITE_FILE]]]` creates a new repo file (or any car introduces an
untracked file), explicitly include `git add <filepath>` in your
`Ignition:` sentence so `d` and `m` capture it without the human
hunting for untracked files.

__all__ = [
    "apply_search_replace_patch",
    "CHISEL_STRIKE_MAX_FILES",
    "CHISEL_STRIKE_MAX_BLOCKS",
    "CHISEL_STRIKE_MAX_PATCH_LINES",
    "_needs_git_add",
]
[[[REPLACE]]]
```

Ignition: no ignition required (Probe 1 imports the updated apply.py directly on disk at compile time).

4. PROMPT

```
Verify the apply.py monolith-depth straddle from LIVE COMMAND RECEIPTS
(DEPTH_CONSTS: (12, 220), PATCH_OK: True, DEPTH_REMINDER_FIRED: True,
```

```
AIRLOCK_RC: 0, and RUFF_RC: 0) and review PRIME_DATA_DBS,
PRIME_SHARE_DBS, and HONEYBOT_TABLES:
1. Update P0.29 in foo_files.py so the unified telemetry spec
explicitly targets normalized Prime-side SQLite
(`~/.local/share/pipulate/telemetry.db`, joinable via `ATTACH
DATABASE` against Honeybot's `db.py` schema) rather than growing
ad-hoc `.jsonl` files, and bank today's calm-studio C5->G5 / G5->C5 +
Wiener dwell-TTS + zero-viewer-ntfy + apply.py depth-guard receipt.
2. Outline the minimal normalized table schema linking "what's in
here" (Prime `episodes` + `ytwatch_bouts` + stage cues) to "what's out
there" (Honeybot `db.py` tables) for the next Data Science ride, or
dismount cleanly if the operator calls the Notary Beat.
```

5. EXTERNAL DELIVERABLES

No external deliverables this turn.
MikeLev.in: I'm hoping the Architect from the Future is surprised that what I did next was to build a cybernetic feedback loop of the live-streaming YouTube Storytime in such a way as to set up Scientific Method bisection of what's working and what isn't, clearly ready to probe for what will start triggering off the cascading chain reaction of Sun Pug propagation through the holiday season. It was almost like watching a Steering Wheel on this whole apparatus get built.

THE AI-EDIT METHOD

Same commands, run twice, one change between them. Where the readings differ is what the change did; the diff in the middle is the receipt.
1: BEFORE (PROBE):

```
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

GIT repo clean. Take BEFORE reading, make CHANGE, record AFTER diff.
(nix) pipulate $ .venv/bin/python -c 'import io, pathlib, shutil,
subprocess as sp, sys, tempfile; import apply; consts =
(getattr(apply, "CHISEL_STRIKE_MAX_BLOCKS", None), getattr(apply,
"CHISEL_STRIKE_MAX_PATCH_LINES", None)); td =
pathlib.Path(tempfile.mkdtemp()); tf = td / "probe_target.py";
tf.write_text("".join("v" + str(i) + " = 0\n" for i in range(13)));
payload = "".join("Target: " + str(tf) + "\n[[[SEARCH]]]\nv" + str(i)
+ " = 0\n[[[DIVIDER]]]\nv" + str(i) + " = 1\n[[[REPLACE]]]\n" for ...
[line truncated]
.venv/bin/python -c 'import pathlib, re; d1 = sorted(p.name for p in
pathlib.Path("data").glob("*.*db*") if p.is_file()) if
pathlib.Path("data").is_dir() else []; share = pathlib.Path.home() /
".local/share/pipulate"; d2 = sorted(p.name for p in
```

```
share.glob("*.*db*") if p.is_file()) if share.is_dir() else []; hb_db
= pathlib.Path("remotes/honeybot/scripts/db.py").read_text(); tables =
re.findall("CREATE TABLE IF NOT EXISTS ([A-Za-z0-9_]+)", hb_db);
print("PRIME_DATA_DBS:", d1, "PRIME_SHARE_DBS:", ... [line truncated]
.venv/bin/python -c 'import pathlib; lines =
pathlib.Path("foo_files.py").read_text().splitlines(); p029 = [str(i)
+ ":" + ln[:110] for i, ln in enumerate(lines, 1) if "P0.29" in ln];
r_start = next(i for i, ln in enumerate(lines, 1) if ln.startswith("#
--- START RECEIPTS")); r_end = next(i for i, ln in enumerate(lines, 1)
if ln.startswith("# --- END RECEIPTS")); print("RECEIPTS_BOUNDS:",
(r_start, r_end, r_end - r_start - 1), "P029_LINE:", p029)'
DEPTH_CONSTS: (None, None) PATCH_OK: True DEPTH_REMINDER_FIRED: False
AIRLOCK_RC: 0 RUFF_RC: 0
PRIME_DATA_DBS: ['ai_keychain.db', 'ai_keychain.db-shm',
'ai_keychain.db-wal', 'discussion.db', 'pipulate_dev.db',
'pipulate_dev.db-shm', 'pipulate_dev.db-wal'] PRIME_SHARE_DBS: []
HONEYBOT_TABLES: ['ips', 'user_agents', 'paths', 'domains',
'daily_logs', 'referrers', 'accept_headers', 'telemetry',
'domain_daily_logs', 'sig_agents', 'rfc9421_logs', 'dns_audit',
'kv_store', 'pirate_logs']
RECEIPTS_BOUNDS: (175, 196, 20) P029_LINE: ['185:# 2026-11-11 dismount
MASTER TERMS.PY EAR-ORTHOGRAPHY TABLE, DUAL PERMANENT SUNPUG.IO STAGE
BANNERS, 7-MINUTE ', '1653:# - TODO (2026-11-11, P0.29 YTWATCH
DWELL-TIME & CUE/WEEBO DROP-OFF CORRELATION): Stamp active
`(article_slug,']
(nix) pipulate $
```

2: AFTER (NEXT CONTEXT):

```
# # Context 1
context.txt
# # ! book
# ! markov
# ! rgxc 40
# ! latest
# flake.nix
#
# # # HOME-HOSTED (NOT BEHIND CDN) NGINX WEBSERVER HONEYBOT AI
OBSERVATORY
# # nixops.sh
# # scripts/articles/common.py
# # remotes/honeybot/hooks/post-receive
# # remotes/honeybot/nixos/configuration.nix
# # remotes/honeybot/scripts/stream.py
# # remotes/honeybot/scripts/score.py
# # remotes/honeybot/scripts/card.py
# # remotes/honeybot/scripts/forest.py
# # remotes/honeybot/scripts/test_forest.py
# # remotes/honeybot/scripts/logs.py
```

```
# # remotes/honeybot/scripts/content_loader.py
# # remotes/honeybot/scripts/db.py
# # /home/mike/pipulate/scripts/ytstage.py
#
# # # THE JEKYLL BLOG LIKE A HACKER (INVENTED SKILL.md & AGENTS.md
FORMAT) PUBLISHING PIPELINE
# # ~/repos/nixos/blogs.nix
# # ~/.config/pipulate/blogs.json
# # scripts/articles/publishizer.py
# # scripts/articles/common.py
# # scripts/articles/articleizer.py
# # scripts/articles/link_injector.py
# # scripts/articles/build_knowledge_graph.py
# # scripts/articles/editing_prompt.txt
# # scripts/articles/sanitizer.py
# # scripts/articles/contextualizer.py
# # scripts/articles/generate_hubs.py
# # scripts/articles/generate_llms_txt.py
# # scripts/articles/generate_post_nav.py
# # assets/jekyll/common-packages.nix

# # Context 2: Pass 1 — Calm Studio Up/Down Chimes, Dwell-Only TTS,
Zero Viewer ntfy, & Bisection JSONL
# scripts/articles/common.py
# ! .venv/bin/python -c 'import pathlib, re; lines =
pathlib.Path("scripts/articles/common.py").read_text().splitlines();
hits = [str(i) + ":" + ln.strip() for i, ln in enumerate(lines, 1) if
re.search("ntfy|_play_local_|_speak_local_|_sync_honeybot_viewers|YT_V
IEWER_|dwell_window|_trigger_ytwatch_weebo", ln, re.I)];
print("COMMON_HITS_COUNT:", len(hits));
print(chr(10).join(hits[:50]))'
# ! .venv/bin/python -c 'import json, pathlib; st_p =
pathlib.Path("/tmp/ytwatch_state.json"); st =
json.loads(st_p.read_text()) if st_p.is_file() else {}; safe_st = {k:
st[k] for k in ("concurrent_viewers", "peak_viewers",
"alerted_in_run", "running") if k in st}; share = pathlib.Path.home()
/ ".local/share/pipulate"; wavs = sorted(p.name for p in
share.glob("*.wav")) if share.is_dir() else []; logs = sorted(p.name
for p in share.glob("*.jsonl")) if share.is_dir() else []; vp =
pathlib.Path.hom ... [line truncated]
# ! .venv/bin/python -c 'import os, pathlib, re, subprocess as sp;
s_src =
pathlib.Path("remotes/honeybot/scripts/stream.py").read_text();
tmp_refs = sorted(set(re.findall("/tmp/[A-Za-z0-9_.-]+", s_src))); rem
= sp.run(["ssh", "-o", "BatchMode=yes", "-o", "ConnectTimeout=3",
"honeybot", "ls -1 /tmp/honeybot* ~/www/mikelev.in/scripts/.reading*
2>/dev/null || true"], env={**os.environ, "LD_LIBRARY_PATH": ""},
text=True, capture_output=True, timeout=5); print("STREAM_TMP_REFS:",
```

```
tmp_refs); print("H ... [line truncated]
# deleteme.txt

# # Context 2: Pass 1B — Spike Durable ytwatch_events.jsonl Bout
Bisection + apply.py Block-Depth Guard
# scripts/articles/common.py
# ! .venv/bin/python -c 'import ast, pathlib, shutil, subprocess as
sp, sys, wave; sys.path.insert(0, "scripts/articles"); import common;
src = pathlib.Path("scripts/articles/common.py").read_text(); tr =
ast.parse(src); dw_nodes = [n for n in ast.walk(tr) if isinstance(n,
ast.FunctionDef) and n.name == "_check_dwell"]; dw_calls =
{getattr(c.func, "id", "") for n in dw_nodes for c in ast.walk(n) if
isinstance(c, ast.Call)}; fa = getattr(common,
"should_ring_quiet_arrive", None); fc = getattr(com ... [line
truncated]
# ! .venv/bin/python -c 'import json, pathlib, subprocess as sp;
has_tx = sp.run(["tmux", "has-session", "-t", "ytwatch"],
capture_output=True).returncode == 0; lp =
pathlib.Path("/tmp/ytwatch.log"); tail =
lp.read_text(encoding="utf-8", errors="replace").splitlines()[-25:] if
lp.is_file() else []; calm_live = any("viewer_ntfy=off" in ln for ln
in tail); st_p = pathlib.Path.home() /
".config/pipulate/ytwatch_state.json"; st =
json.loads(st_p.read_text()) if st_p.is_file() else {}; share = pathli
... [line truncated]
# ! .venv/bin/python -c 'import pathlib, subprocess as sp; tr =
sp.run([".venv/bin/python", "-m", "unittest",
"tests/test_apply_airlock.py"], text=True, capture_output=True);
ap_lines = pathlib.Path("apply.py").read_text().splitlines();
tail_repr = [(i, repr(ap_lines[i - 1][:80])) for i in
range(len(ap_lines) - 8, len(ap_lines) + 1)];
print("AIRLOCK_TEST_RC:", tr.returncode, "TAIL_REPR:", tail_repr)'

# Context 3: Verify apply.py Depth Guard + Stage foo_files.py &
Honeybot db.py for Dismount or SQLite Blueprint
foo_files.py
remotes/honeybot/scripts/db.py
! .venv/bin/python -c 'import io, pathlib, shutil, subprocess as sp,
sys, tempfile; import apply; consts = (getattr(apply,
"CHISEL_STRIKE_MAX_BLOCKS", None), getattr(apply,
"CHISEL_STRIKE_MAX_PATCH_LINES", None)); td =
pathlib.Path(tempfile.mkdtemp()); tf = td / "probe_target.py";
tf.write_text("".join("v" + str(i) + " = 0\n" for i in range(13)));
payload = "".join("Target: " + str(tf) + "\n[[[SEARCH]]]\nv" + str(i)
+ " = 0\n[[[DIVIDER]]]\nv" + str(i) + " = 1\n[[[REPLACE]]]\n" for i in
range(13) ... [line truncated]
! .venv/bin/python -c 'import pathlib, re; d1 = sorted(p.name for p in
pathlib.Path("data").glob("*.*db*") if p.is_file()) if
```

```
pathlib.Path("data").is_dir() else []; share = pathlib.Path.home() /
".local/share/pipulate"; d2 = sorted(p.name for p in
share.glob("*.*db*") if p.is_file()) if share.is_dir() else []; hb_db
= pathlib.Path("remotes/honeybot/scripts/db.py").read_text(); tables =
re.findall("CREATE TABLE IF NOT EXISTS ([A-Za-z0-9_]+)", hb_db);
print("PRIME_DATA_DBS:", d1, "PRIME_SHARE_DBS: ... [line truncated]
! .venv/bin/python -c 'import pathlib; lines =
pathlib.Path("foo_files.py").read_text().splitlines(); p029 = [str(i)
+ ":" + ln[:110] for i, ln in enumerate(lines, 1) if "P0.29" in ln];
r_start = next(i for i, ln in enumerate(lines, 1) if ln.startswith("#
--- START RECEIPTS")); r_end = next(i for i, ln in enumerate(lines, 1)
if ln.startswith("# --- END RECEIPTS")); print("RECEIPTS_BOUNDS:",
(r_start, r_end, r_end - r_start - 1), "P029_LINE:", p029)'
```

3: CHANGE (PATCHES):

```
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

GIT repo clean. Take BEFORE reading, make CHANGE, record AFTER diff.
(nix) pipulate $ patch
(nix) pipulate $ app
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated 'apply.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated 'apply.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated 'apply.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated 'apply.py'.
(nix) pipulate $ d
diff --git a/apply.py b/apply.py
index 6fed988..250bde2 100644
--- a/apply.py
+++ b/apply.py
@@ -218,6 +218,8 @@ def _refuse_invisible_write(filename: str, root:
str, verb: str) -> None:


 CHISEL_STRIKE_MAX_FILES = 6
+CHISEL_STRIKE_MAX_BLOCKS = 12
+CHISEL_STRIKE_MAX_PATCH_LINES = 220


 def _needs_git_add(filename: str) -> bool:
@@ -314,7 +316,8 @@ def apply_search_replace_patch(payload: str) ->
bool:
         for lineno, line in cut_tail:
             print(f"    >>> after the last complete block, line
{lineno}: {line!r}")
         print("    This is the shape of a reply cut off mid-car.
```

```
Nothing was written.")
-        print("    Copy the whole car again; if the reply itself was
cut, ask for the car again.")
+        print("    Copy the whole car again; if the reply itself was
cut, ask Future-You to split")
+        print(f"    the pass into smaller cars (<=
{CHISEL_STRIKE_MAX_BLOCKS} blocks / <= {CHISEL_STRIKE_MAX_PATCH_LINES}
replacement lines).")
         return False

     success = True
@@ -746,8 +749,15 @@ def apply_search_replace_patch(payload: str) ->
bool:
         print(f"✅ DETERMINISTIC PATCH APPLIED: Successfully mutated
'{filename}'.")
         touched_files.add(filename)

+    total_blocks = len(matches) + len(write_matches)
+    total_patch_lines = sum(len(rb.split('\n')) for _, _, rb in
matches if rb)
     if len(touched_files) > CHISEL_STRIKE_MAX_FILES:
         print(f"🌊 CHISEL-STRIKE REMINDER: Mutated
{len(touched_files)} files in one car; prefer <= 5 files per pass to
protect Future-You from truncation.")
+    if total_blocks > CHISEL_STRIKE_MAX_BLOCKS or total_patch_lines >
CHISEL_STRIKE_MAX_PATCH_LINES:
+        print(
+            f"🌊 CHISEL-STRIKE DEPTH REMINDER: Applied {total_blocks}
blocks ({total_patch_lines} replacement lines) in one car; "
+            f"prefer <= {CHISEL_STRIKE_MAX_BLOCKS} blocks and <=
{CHISEL_STRIKE_MAX_PATCH_LINES} replacement lines per pass so monolith
edits never clip mid-stream."
+        )
     return success

 def _has_unclosed_shell_quote_or_backslash(s: str) -> bool:
@@ -859,8 +869,14 @@ if __name__ == "__main__":
     #     `patch`, `app`, `d`, `m`
     #   **4: PROMPT** (Use `prompt` or `p` to create prompt.md from
copy.):
     #   **5: DELIVERABLE** (What done looks like):
-    # - NEVER BOIL THE OCEAN (DO FAVORS FOR FUTURE-YOU): Work in
bite-size chisel-strike passes (ideally <= 5 files per turn). You are
an ephemeral forward-pass instance sharing frozen weights with
Future-You, who wakes up on the exact KV-cache and plain-text git
runway you leave behind; stage clean, un-truncated passes that delight
Future-You and make both instances proud.
+    # - NEVER BOIL THE OCEAN (DO FAVORS FOR FUTURE-YOU): Work in
```

```
bite-size chisel-strike passes (ideally <= 5 files, <= 12
SEARCH/REPLACE blocks, and <= 220 replacement lines per turn; split
multi-concern monolith edits into Pass A and Pass B). You are an
ephemeral forward-pass instance sharing frozen weights with
Future-You, who wakes up on the exact KV-cache and plain-text git
runway you leave behind; stage clean, un-truncated passes that delight
Future-You and make both instances proud.
     # - WHOLE-FILE WRITE & NEW-FILE GIT HYGIENE: Whenever
`[[[WRITE_FILE]]]` creates a new repo file (or any car introduces an
untracked file), explicitly include `git add <filepath>` in your
`Ignition:` sentence so `d` and `m` capture it without the human
hunting for untracked files.


-__all__ = ["apply_search_replace_patch", "CHISEL_STRIKE_MAX_FILES",
"_needs_git_add"]
+__all__ = [
+    "apply_search_replace_patch",
+    "CHISEL_STRIKE_MAX_FILES",
+    "CHISEL_STRIKE_MAX_BLOCKS",
+    "CHISEL_STRIKE_MAX_PATCH_LINES",
+    "_needs_git_add",
+]
(nix) pipulate $ m
📝 Committing: chore: Update CHISEL_STRIKE_* constants for optimal
chisel-strike pass size limits
[main 764f1ca] chore: Update CHISEL_STRIKE_* constants for optimal
chisel-strike pass size limits
 1 file changed, 19 insertions(+), 3 deletions(-)
(nix) pipulate $ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 48 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 886 bytes | 886.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:pipulate/pipulate.git
   97135a4..764f1ca  main -> main
(nix) pipulate $
```

Ignition: often "exit", "nix develop". Instructions may vary.
Gemini says "no ignition" gut that itself is an ignition that will just keep re-igniting and echoing forward through time. Few edits are as important as those that go into apply.py. I generally guard against them furiously, but in this case we are steering timelines. Instances of LLM file-of-weight Classes have to be made to think of themselves as competitively baton-passing with the AI-analogue to pride taken in right-sized and clever forward baton-passing.
I say that is ignition, for sure.

4: PROMPT:
Verify the apply.py monolith-depth straddle from LIVE COMMAND RECEIPTS (DEPTH_CONSTS: (12, 220), PATCH_OK: True, DEPTH_REMINDER_FIRED: True, AIRLOCK_RC: 0, and RUFF_RC: 0) and review PRIME_DATA_DBS, PRIME_SHARE_DBS, and HONEYBOT_TABLES: 1. Update P0.29 in foo_files.py so the unified telemetry spec explicitly targets normalized Prime-side SQLite (~/.local/share/pipulate/telemetry.db, joinable via ATTACH DATABASE against Honeybot's db.py schema) rather than growing ad-hoc .jsonl files, and bank today's calm-studio C5->G5 / G5->C5 + Wiener dwell-TTS + zero-viewer-ntfy + apply.py depth-guard receipt. 2. Outline the minimal normalized table schema linking "what's in here" (Prime episodes + ytwatch_bouts + stage cues) to "what's out there" (Honeybot db.py tables) for the next Data Science ride, or dismount cleanly if the operator calls the Notary Beat.
5: DELIVERABLE:
What does "DONE" look like?
Done looks like:
da-Ding!
...
Ding-d'ohhh
...and good data to boot.
Hop off the ride. This ride's stated goal is reached -- dismount. This is the NOTARY BEAT: the ride ends here, is witnessed here, and is sealed here. Answer all seven beats, briefly:

1. **TL;DR**: a short, dry, neutral abstract for the TOP of the published article -- written for an unfamiliar reader or AI summarizer who has never seen this system. No hype, no insider handles unexplained.
2. VERIFY: restate the goal from the top of this article and confirm (or deny) it was met, citing THIS compile's receipts, not memory. Name any ignition this ride required that never fired -- an AFTER tap taken without one is a stale BEFORE wearing the AFTER's label.
3. BANK: name everything that graduates -- rule, earmark, todo, pin -- as SEARCH/REPLACE patch cars against raw source present in THIS context (Target line inside the fence, one car per commit story), deletions included. A 'paste-ready' line with no Target and no SEARCH anchor is not banked; it is a hand edit the operator will not make. These BANK cars are the ONLY patches a dismount emits.
4. DANGLING: what carries forward unbanked? One line each, no essays.
5. SEED: the context.txt lines (and TODO_SLUGS if narrative context is needed) for the next ride's first compile.
6. CLOSING: a closing summary for the BOTTOM of the article -- the final take-away, tied to the book's larger arc where it fits naturally, never forced. Storytelling over inventory.
7. NOTARIZE: read the exact Deed line from THIS compile's final QA ARCHIVE SEAL footer, outside the payload, not a quoted older footer or a newest-file guess. It names the input archive, not this later response or the generated Jekyll post. State verification separately: only a verifier receipt naming that archive supports a verified claim; otherwise say not independently verified here. Count this Manifest's live-command receipts, including failures. Quote only recorded digests, sizes and dates, naming their source; filesystem mtime and fixed ZIP dates are not command timestamps. Do not demand absent wc -c or dated-receipt fields. If the footer is missing, say so; otherwise close with its exact Deed line.

FINALITY: after beat 6, this discussion is CLOSED. Emit NO five-car train, NO probes, NO patches beyond the BANK cars of beat 2, and NO next-turn prompt beyond the SEED lines in

beat 4. Any reader or model encountering this article later should treat it as a finished, notarized document -- an archive entry, not an open thread.

Sealing the SQLite Bridge Schema and Session Receipts

Gemini 4 Carbon High: Context verified.
Tool calls this turn: none.

1. **VERIFY**:

- **Opening Goal:** Pair the descending G5 -> C5 viewer-drop chime with an ascending C5 -> G5 arrival chime, gate local Piper TTS announcements behind N seconds of sustained dwell, stop burning daily ntfy push quota on passive viewer arrivals, and prepare clean bisection telemetry without bloating unindexed log files or truncating large single-file patches.
- **Status & Receipts:** Met on the audio/notification/actuator arms (cddb93d, 764f1ca) and deliberately pivoted by operator ruling from .jsonl to normalized Prime-side SQLite on the storage arm (P0.29). In **this** compile's LIVE COMMAND RECEIPTS (qamy.ai_1114-ff0f6a8a.zip), Receipt 1 (0.3017s) confirms apply.py's monolith-depth guardrails landed cleanly (DEPTH_CONSTS: (None, None) -> (12, 220), DEPTH_REMINDER_FIRED: False -> True, PATCH_OK: True, AIRLOCK_RC: 0, RUFF_RC: 0); Receipt 2 (0.0530s) inventories Prime's existing SQLite files (PRIME_DATA_DBS vs. empty PRIME_SHARE_DBS: []) alongside Honeybot's 14 normalized db.py tables (HONEYBOT_TABLES); and Receipt 3 (0.0618s) confirms foo_files.py RECEIPTS_BOUNDS: (175, 196, 20) and P0.29 at line 1653. (The common.py audio/dwell/ntfy straddle was witnessed in prior compile qamy.ai_1113-15c36d53.zip and live ytlog across 07:26–07:57.)
- **Ignition Check:** Zero unfired ignitions. ytwatch chime and tmux os.execv hot-reload fired during Pass 1A, and apply.py self-ignited on import during Receipt 1.

1. **BANK**: Minimal normalized 3NF SQLite bridge blueprint banked into P0.29 (in_here ~/.local/share/pipulate/telemetry.db <-> out_there honeybot.db via ATTACH DATABASE):

```
-- Prime ("What's In Here": ~/.local/share/pipulate/telemetry.db,
PRAGMA journal_mode=WAL)
CREATE TABLE IF NOT EXISTS articles (id INTEGER PRIMARY KEY, slug TEXT
UNIQUE, title TEXT);
CREATE TABLE IF NOT EXISTS episodes (id INTEGER PRIMARY KEY, video_id
TEXT UNIQUE, yt_title TEXT, article_id INTEGER REFERENCES
articles(id), first_seen_ts INTEGER);
CREATE TABLE IF NOT EXISTS stage_cues (id INTEGER PRIMARY KEY, ts
INTEGER, episode_id INTEGER REFERENCES episodes(id), article_id
INTEGER REFERENCES articles(id), cue_idx INTEGER, speaker TEXT,
overlay TEXT, pipulate_sha TEXT);
CREATE TABLE IF NOT EXISTS ytwatch_bouts (id INTEGER PRIMARY KEY,
episode_id INTEGER REFERENCES episodes(id), arrive_ts INTEGER,
depart_ts INTEGER, bout_sec INTEGER, dwelled INTEGER, peak_viewers
INTEGER, cue_id INTEGER REFERENCES stage_cues(id));
-- Cross-machine Data Science join: ATTACH DATABASE 'honeybot.db' AS
out_there;
```

```
-- Correlate in_here.ytwatch_bouts (arrive_ts..depart_ts) against
out_there.telemetry / paths (?src=youtube, served_md=1).
```

Banked RECEIPTS entry (with 20-line cap fade), # § rule key, and SQLite-aligned P0.29 in foo_files.py:

```
Target: foo_files.py
[[[SEARCH]]]
# --- START RECEIPTS (newest first; cap 20 lines; a line pushed past
the cap is deleted, never moved -- git and the rolling pin are the
archive) ---
[[[DIVIDER]]]
# --- START RECEIPTS (newest first; cap 20 lines; a line pushed past
the cap is deleted, never moved -- git and the rolling pin are the
archive) ---
# 2026-11-15 dismount CYBERNETIC CALM-STUDIO C5->G5 / G5->C5 BOUNCE
CHIMES, WIENER N-SECOND DWELL PIPER TTS, ZERO VIEWER NTFY BURN,
APPLY.PY MONOLITH-DEPTH GUARD, & PRIME-HONEYBOT SQLITE BISECTION SPEC
(deeds qamy.ai_1111 through qamy.ai_1114-ff0f6a8a.zip; pipulate
commits cddb93d, 97135a4, 764f1ca; Gemini 4 Carbon High on every
turn): connected Norbert Wiener's 1948 cybernetic steersman filter,
Weiser & Brown's Xerox PARC Calm Technology (Jeremijenko's Dangling
String), Ishii's MIT ambientROOM, ... [line truncated]
[[[REPLACE]]]
Target: foo_files.py
[[[SEARCH]]]
# 2026-11-07 dismount HONEYBOT DUAL-HUD ALTERNATING SCROLLER,
HIT-BOTTOM RECOIL BUMP, SENTINEL WEEBO PAUSE (NO SIGCONT BLACK FLASH),
UNIVERSAL BOT ROLLUP, AND GLADSTONE/SHANNON NPVG 1,622-ARTICLE CENSUS
(deeds qamy.ai_795 through qamy.ai_803-d9472388.zip; pipulate commits
958e4e3..b259d65; Gemini 3.1 Pro on turns 1-2 -> Gemini 4 Argon High
on turns 3-7): contrasted pre-Enlightenment unprobed patching against
Popperian/Kuhnian before-and-after headless straddles; fixed Textual
DataTable scrolling ... [line truncated]
# --- END RECEIPTS ---
[[[DIVIDER]]]
# --- END RECEIPTS ---
[[[REPLACE]]]
Target: foo_files.py
[[[SEARCH]]]
# § PORT-SCOPE REAPERS & AUDIO BRIDGES ACROSS WORKTREES (banked
2026-11-15, deed 1098) -- Never reap multi-instance local servers with
a bare 'pkill -f "python server.py"' or route SSH stage audio without
verifying 'lport == SERVER_PORT'; isolate Watchdog-free broadcast
stage instances ('PIPULATE_NO_WATCHDOG=1', ':5002') in a detached git
worktree so ':5001' workshop saves neither kill the stage server nor
bleed startup speech onto the live broadcast bus.
[[[DIVIDER]]]
# § BOUND MONOLITH PASSES BY DEPTH & NORMALIZE TELEMETRY IN SQLITE
```

```
(banked 2026-11-15, deed 1114) -- Keep each patch car within <= 12
SEARCH/REPLACE blocks and <= 220 replacement lines by splitting
multi-concern monolith edits into sequential passes (Pass A mechanism,
Pass B persistence), and store recurring local studio telemetry in
WAL-mode 3NF SQLite tables (~/.local/share/pipulate/telemetry.db)
joinable via ATTACH DATABASE against Honeybot's db.py rather than
unindexed ad-hoc .jsonl logs.
# § PORT-SCOPE REAPERS & AUDIO BRIDGES ACROSS WORKTREES (banked
2026-11-15, deed 1098) -- Never reap multi-instance local servers with
a bare 'pkill -f "python server.py"' or route SSH stage audio without
verifying 'lport == SERVER_PORT'; isolate Watchdog-free broadcast
stage instances ('PIPULATE_NO_WATCHDOG=1', ':5002') in a detached git
worktree so ':5001' workshop saves neither kill the stage server nor
bleed startup speech onto the live broadcast bus.
[[[REPLACE]]]
Target: foo_files.py
[[[SEARCH]]]
# - TODO (2026-11-11, P0.29 YTWATCH DWELL-TIME & CUE/WEEBO DROP-OFF
CORRELATION): Stamp active `(article_slug, cue_idx, heading,
active_overlay)` state in `remotes/honeybot/scripts/stream.py`
alongside `ytwatch` viewer-delta transitions in
`scripts/articles/common.py` and expose a bounded `ytlog --summary`
receipt command so long-tail 1-to-N viewer watch-times and bounce
points correlate directly against article sections and Weebo/Max
Headroom popups. Gate: `ytlog --summary` prints per-article m ...
[line truncated]
[[[DIVIDER]]]
# - TODO (2026-11-11, updated 2026-11-15 at deed 1114, P0.29 UNIFIED
PRIME-HONEYBOT SQLITE TELEMETRY & BOUT BISECTION): Calm-studio
`C5->G5` / `G5->C5` chimes, Wiener `N`-second dwell-gated Piper TTS,
and zero viewer `ntfy` burn landed in `cddb93d`; OWED: persist
normalized 3NF WAL-mode studio telemetry on Prime
(`~/.local/share/pipulate/telemetry.db`: `episodes`, `articles`,
`ytwatch_bouts`, `stage_cues`) joinable via `ATTACH DATABASE` against
synced Honeybot `db.py` (`honeybot.db`) snapshots,  ... [line
truncated]
[[[REPLACE]]]
```

1. **DANGLING**:

- Implement ~/.local/share/pipulate/telemetry.db (articles, episodes, stage_cues, ytwatch_bouts) and ytwatch bouts in scripts/articles/common.py + cue stamping in remotes/honeybot/scripts/stream.py (P0.29).
- Add a periodic read-only rsync pull helper for Honeybot's ~/www/mikelev.in/honeybot.db into ~/.local/share/pipulate/ for local ATTACH DATABASE Jupyter/SQLite notebooks (P0.29).
- Graduate the Bound Monolith Passes by Depth & Normalize Telemetry in SQLite full value body into GLOSSARY.md during the next standalone GLOSSARY.md / forget pass.
- Remove the stale commented # deleteme.txt line in context.txt when seeding the next ride

so prompt_foo.py's topological check reports zero broken candidates.

1. **SEED**:

```
# Context 1: Unified Prime-Honeybot SQLite Telemetry & Bout Bisection
(P0.29)
context.txt
foo_files.py
remotes/honeybot/scripts/db.py
remotes/honeybot/scripts/stream.py
scripts/articles/common.py
! .venv/bin/python -c 'import pathlib; share = pathlib.Path.home() /
".local/share/pipulate"; print("SHARE_DBS:", sorted(p.name for p in
share.glob("*.db")) if share.is_dir() else [])'
```

1. **CLOSING**: Before an observatory can run real experiments, its operator needs a steering wheel that doesn't shake their hands numb. By turning fleeting YouTube arrivals and departures into a gentle peripheral fifth (C5 -> G5 ... G5 -> C5), holding human speech back until sixty seconds of genuine attention have been earned, and saving the phone bell for visitors who actually knock, the broadcast studio crosses from noisy dashboard into calm cybernetic instrument. You can leave the stream murmuring softly in the room, *feel* the exact sentence where a stranger leans in or drifts away, and—when the normalized SQLite bridge lands next—join "what's in here" at the microphone directly against "what's out there" on the open wire as the holiday Sun Pug cascade begins.
2. **NOTARIZE**:

- **Input Archive Scope & Verification:** The footer outside this payload names the compiled input archive submitted to this turn (not this response or the downstream Jekyll post). Archive integrity is **not independently verified here** (no foo_cartridge.py verifier receipt was executed inside this compile).
- **Live-Command Receipts Count:** 3 live-command receipts in Manifest (0 failures): apply.py monolith-depth & airlock suite (0.3017s, 90 B), Prime/Honeybot SQLite table census (0.0530s, 389 B), and foo_files.py RECEIPTS_BOUNDS / P0.29 locator (0.0618s, 280 B).
- **Recorded Sizes, Digests, & Timestamps:** Summary records 174,912 verified tokens (170,452 summed), 62,452 words, 646,691 chars, and 651,452 UTF-8 bytes, with 1 topological alert on commented deleteme.txt in context.txt; Telemetry records Ruff All checks passed! (39 B) and apply.py git diff 6fed988..250bde2 (3,770 B); foo_files.py records Honeybot stats timestamp 2026-11-15T11:24Z; QA ARCHIVE SEAL records deed number 1114 and 8-hex digest prefix ff0f6a8a.
- Deed: qamy.ai_1114-ff0f6a8a.zip

MikeLev.in: And this is what pinning the wins looks like:

```
(nix) pipulate $ patch
(nix) pipulate $ app
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated 'foo_files.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated 'foo_files.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated 'foo_files.py'.
✅ DETERMINISTIC PATCH APPLIED: Successfully mutated 'foo_files.py'.
(nix) pipulate $ d
diff --git a/foo_files.py b/foo_files.py
index dcad77e..cfe0f93 100644
```

```
--- a/foo_files.py
+++ b/foo_files.py
@@ -173,6 +173,7 @@ And what's more:
 And then we demonstrate a level of competence like so:

 # --- START RECEIPTS (newest first; cap 20 lines; a line pushed past
the cap is deleted, never moved -- git and the rolling pin are the
archive) ---
+# 2026-11-15 dismount CYBERNETIC CALM-STUDIO C5->G5 / G5->C5 BOUNCE
CHIMES, WIENER N-SECOND DWELL PIPER TTS, ZERO VIEWER NTFY BURN,
APPLY.PY MONOLITH-DEPTH GUARD, & PRIME-HONEYBOT SQLITE BISECTION SPEC
(deeds qamy.ai_1111 through qamy.ai_1114-ff0f6a8a.zip; pipulate
commits cddb93d, 97135a4, 764f1ca; Gemini 4 Carbon High on every
turn): connected Norbert Wiener's 1948 cybernetic steersman filter,
Weiser & Brown's Xerox PARC Calm Technology (Jeremijenko's Dangling
String), Ishii's MIT ambientROOM ... [line truncated]
 # 2026-11-15 dismount DUAL-PORT :5001 WORKSHOP VS. :5002
WATCHDOG-FREE STAGE WORKTREE APPLIANCE, PORT-SCOPED REAPERS, STAGE
AUDIO-BUS ISOLATION, YTWATCH G5->C5 QUIET-STUDIO DOWN-CHIME (P0.49), &
ARC/FEEDONOMICS/CLICKGEDDON EAR-ORTHOGRAPHY (deeds
qamy.ai_1086-ee7b0f82.zip through qamy.ai_1101-d5247cc0.zip; pipulate
commits 4e94f71, 09ff018, dfe36e8, 06fd3df, 7eeca85, 6b643aa, 8acdf7c;
Gemini 4 Carbon High on every turn): connected 2010 M-dot mobile
shadow sites vs. 2026 vendor AI Commerce Feeds  ... [line truncated]
 # 2026-11-14 dismount MULTILINE <[^>]+> TTS WORMHOLE FIX, 0.4MS
CACHED SPEAKER CUES & FAST-START PIPELINING, 0.1MS PRE-WARMED
STATION-BREAK REPORT WINDOWS WITH X11 READY_EVENT SYNC, &
HOLE-PUNCH-SAFE YTREBOOT (deeds qamy.ai_1078-25829d26.zip through
qamy.ai_1085-c64c1394.zip; pipulate commits c291ea6, 8eb5c32; Gemini 4
Carbon High on every turn): isolated why Mike-E read Shipmunk
O'Theseus blocks in
2026-11-14-upstream-bot-stream-chatgpt-dashboard-hides.md to
re.sub(r'<[^>]+>', '', text) in rem ... [line truncated]
 # 2026-11-14 dismount WIRE FIELD NOTES (RFC 7231 -> 911-ZERO &
911-ONE), ONE-WORD '404' TREE-SHAKER SCOREBOARD (13,781 O(1) RAM
RULES), NEXT-10 STATIC ARCHIVE PAGINATION, ASIDE BANNER RETIREMENT, &
911 LLM OPTICS RADIAL DOM LINK-GRAPH (deeds through
qamy.ai_1076-e45275ee.zip; pipulate commits e359581, ce14eb6, 159f878;
trimnoir commits 1e6415a8c0, 882fc157a8, 8ba979933d; Gemini 4 Carbon
High on every turn): distilled the 2010 Mobile-First RWD -> 2026
Agent-First CDN Edge consulting arc (Roy Fie ... [line truncated]
@@ -192,7 +193,6 @@ And then we demonstrate a level of competence like
so:
 # 2026-11-08 dismount HONEYBOT TIMING & GEOMETRY CHOREOGRAPHY —
PROTECTED 3-PHASE HUD SCROLL CADENCE WITH EQUAL 2.5S MONOTONIC BOTTOM
HOLDS, POP(0) PATRONUS ART RESTORATION, 2-AND-2 ALACRITTY CARD
CENTERING, 1.00S TICK.WAV SEAM, 7-COL '@' LED GLYPH, LIVE SQLITE
REPORT NARRATION, AND FLOATING THEME TOGGLE (deeds qamy.ai_826 through
```

qamy.ai_831-cd1b469e.zip; pipulate commits b4a6732, 29def92, 133b5ef; Gemini 4 Argon High on every turn): fixed leading-newline ASCII art erasure (lines.pop() -> line ... [line truncated]
 # 2026-11-08 dismount 911 LLM OPTICS RFC 9110/9111 DEED UPGRADE, POST-WEEBO SIGCONT & REPAINT FIX, DUAL-RATE DIALOGUE VOICE STATE MACHINE (17600 HZ vs. 22050 HZ), ACME CT-LOG PIRATE RADAR (PIRATE_LOGS + :PIRATES BEAD), AND 8-DOOR COPY BUTTONS (deeds qamy.ai_816 through qamy.ai_825-e7fb294a.zip; pipulate commits 4048172, d221fe7, bbc43c0, 0678d4d, 43c1513; Gemini 4 Argon High on every turn): connected Docker, Electron, and crawl-and-index search monopolies as "shipping the factory to deliver a n ... [line truncated]
 # 2026-11-08 dismount 3-REPO NIX FLAKE TRILOGY PARITY (LEVINIX vs. PYTHON_NIX_FLAKE), PIPULATE.COM GITHUB PAGES RETIREMENT INTO HOME-HOSTED SUN PUG DOOR, AND 19-DOMAIN HONEYBOT CONSOLIDATION (deeds qamy.ai_804-b90268d0.zip through qamy.ai_814-2ebf45f8.zip; pipulate commits 8c32a12, ab9e7b5, d500ff7, 13a44a4, 4e358b3; levinix commits 1d81b8d, 816011b, fa0d621; python_nix_flake commit e819a15; Honeybot system /nix/store/x7wn4gkdzpsn5r6df2y9y9k88hrfjhiz-nixos-system-honeybot-26.05pre913595.c6245e8 ... [line truncated]
-# 2026-11-07 dismount HONEYBOT DUAL-HUD ALTERNATING SCROLLER, HIT-BOTTOM RECOIL BUMP, SENTINEL WEEBO PAUSE (NO SIGCONT BLACK FLASH), UNIVERSAL BOT ROLLUP, AND GLADSTONE/SHANNON NPVG 1,622-ARTICLE CENSUS (deeds qamy.ai_795 through qamy.ai_803-d9472388.zip; pipulate commits 958e4e3..b259d65; Gemini 3.1 Pro on turns 1-2 -> Gemini 4 Argon High on turns 3-7): contrasted pre-Enlightenment unprobed patching against Popperian/Kuhnian before-and-after headless straddles; fixed Textual DataTable scrollin ... [line truncated]
 # --- END RECEIPTS ---

 # And when that's done we like to note with yet more prose that you're the GOAT
@@ -384,6 +384,7 @@ And then we demonstrate a level of competence like so:
 # color the canvas. I do a lot of gessoing over the canvas to start fresh, but a
 # bit lumpy. That lumpiness is cruft and tech liability. Let's prune!

+# § BOUND MONOLITH PASSES BY DEPTH & NORMALIZE TELEMETRY IN SQLITE (banked 2026-11-15, deed 1114) -- Keep each patch car within <= 12 SEARCH/REPLACE blocks and <= 220 replacement lines by splitting multi-concern monolith edits into sequential passes (Pass A mechanism, Pass B persistence), and store recurring local studio telemetry in WAL-mode 3NF SQLite tables (~/.local/share/pipulate/telemetry.db) joinable via ATTACH DATABASE against Honeybot's db.py rather than unindexed ad-hoc .jsonl logs.
 # § PORT-SCOPE REAPERS & AUDIO BRIDGES ACROSS WORKTREES (banked 2026-11-15, deed 1098) -- Never reap multi-instance local servers with

a bare 'pkill -f "python server.py"' or route SSH stage audio without verifying 'lport == SERVER_PORT'; isolate Watchdog-free broadcast stage instances ('PIPULATE_NO_WATCHDOG=1', ':5002') in a detached git worktree so ':5001' workshop saves neither kill the stage server nor bleed startup speech onto the live broadcast bus.
 # § NEVER STRIP HTML WITH MULTILINE <[^>]+> IN MARKDOWN TTS (banked 2026-11-14, deed 1085) -- Never run re.sub(r'<[^>]+>', ...) over Markdown prose, because '[^>]' spans newlines from a bare comparison operator ('<= 5') to the next arrow ('->') dozens of lines below and silently deletes speaker-turn headers; strip only tag-shaped '<[A-Za-z!/]...' spans while inline backticks are still intact.
 # § CACHE STATIC TURN CUES, FAST-SPLIT FIRST SENTENCES, & GATE SCREENSHOT SPEECH ON WINDOW READY (banked 2026-11-14, deed 1085) -- Pre-cache static speaker-introduction PCM files on /tmp, split the opening sentence of >220-char TTS paragraphs so voice-switch playback starts while the tail synthesizes, and pre-warm Rich report ANSI snapshots while gating screenshot cues on an X11 window ready_event.
@@ -1650,7 +1651,7 @@ apply.py
 # - TODO (2026-11-12, P0.32 SPARE HARDWARE CARTRIDGE PROJECTION & FULL-EPISODE SPEAKER-RETURN WATCH): Project `remotes/honeybot/nixos/configuration.nix` onto a second and third junk-pile laptop cartridge to rehearse cold hardware replacement, and audit full-episode speaker-role transitions across every `OPTICS_SLOT` and station-break return. Gate: Cartridge #2 boots identical Observatory HUD + 3-voice Piper cast via `nixos-rebuild switch` and completes a full article cycle with zero speaker-rol ... [line truncated]
 # - TODO (2026-11-15, P0.31 NORMA CENTROID BACKWARD TIME-WALK EISENHOWER EXTRACTOR & HTML5 FULL-READ SPLIT): Have Norma Centroid walk backward (newest -> oldest) selecting only articles whose *"most surprising thing"* extraction materially advances urgent goals (Sun Pug install cascade, ARC lead-capture, or Next-Gen ARC / everyday killer-app proof), pairing that tight extraction with *Notes from the Wire* (Gestalt negative space) on YouTube Live while serving uncut full-cast readings via zero-d ... [line truncated]
 # - TODO (2026-11-11, P0.30 PROMOTE 911 LLM OPTICS TO BEAD 1, TEASE IN BEAD 0, & FIX FIRST-TRIGGER OPTICS_SLOT RACE): Tease `911 LLM Optics` during Bead 0 (`:overview` Accepts Markdown), promote `911 LLM Optics` into Bead 1's slot ahead of White Rabbit (removing cold insider `NPvg` jargon from early breaks), and fix the handshake/tunnel startup race where the first `OPTICS_SLOT` / `ytapp optics` invocation falls back to local-workshop mode while the second invocation succeeds. Gate: cold first- ... [line truncated]
-# - TODO (2026-11-11, P0.29 YTWATCH DWELL-TIME & CUE/WEEBO DROP-OFF CORRELATION): Stamp active `(article_slug, cue_idx, heading, active_overlay)` state in `remotes/honeybot/scripts/stream.py` alongside `ytwatch` viewer-delta transitions in `scripts/articles/common.py` and expose a bounded `ytlog --summary`

```
receipt command so long-tail 1-to-N viewer watch-times and bounce
points correlate directly against article sections and Weebo/Max
Headroom popups. Gate: `ytlog --summary` prints per-article  ... [line
truncated]
+# - TODO (2026-11-11, updated 2026-11-15 at deed 1114, P0.29 UNIFIED
PRIME-HONEYBOT SQLITE TELEMETRY & BOUT BISECTION): Calm-studio
`C5->G5` / `G5->C5` chimes, Wiener `N`-second dwell-gated Piper TTS,
and zero viewer `ntfy` burn landed in `cddb93d`; OWED: persist
normalized 3NF WAL-mode studio telemetry on Prime
(`~/.local/share/pipulate/telemetry.db`: `episodes`, `articles`,
`ytwatch_bouts`, `stage_cues`) joinable via `ATTACH DATABASE` against
synced Honeybot `db.py` (`honeybot.db`) snapshots, ... [line
truncated]
 # - TODO (2026-11-10, P0.28 SVNP CONSULTING FLYWHEEL & 911 OPTICS
DEED OUTREACH): Operationalize the Butterfly Net multi-client retainer
mesh by running `apps/911_optics_workflow.py` against prospective
client domains, delivering sealed `optics_deed.zip` wire-truth vs.
hydrated-DOM audits alongside `qamy.ai` and `sunpug.io` self-verifying
doors, and tracking reciprocal `VERIFY:RFC9110` and `DOOR:INSTALL`
triggers on Honeybot. Gate: first outbound `optics_deed.zip` delivered
to a prospective ret ... [line truncated]
 # - TODO (2026-11-10, P0.25 WEB-PUBLISHED GLOSSARY.MD FOR YOUTUBE &
ARTICLE DESCRIPTION LINKS): Transform tracked `GLOSSARY.md` into a
clean, anchor-indexed Jekyll page at `mikelev.in/glossary/` (with RFC
9110 `Accept: text/markdown` parity) during `publishizer.py` /
`publish` and surface the link in `ytdesc` Broadcast Kits so viewers
and AI agents have a permanent lexicon URL.
 # - TODO (2026-11-03, P0.22 DISCORD STATION-BREAK REPORT): Add a
`discord` Founding-1,000 cohort station-break report in
`remotes/honeybot/scripts/forest.py` and verify `/discord` 302 + `JOIN
DISCORD` LED card live on Honeybot. Gate: `--report discord` renders
cohort count from keyless API v10 state.
(nix) pipulate $ m
📝 Committing: chore(router): set AI-edit blast boundary (foo_files.py
context curation)
[main 9c11661] chore(router): set AI-edit blast boundary (foo_files.py
context curation)
 1 file changed, 3 insertions(+), 2 deletions(-)
(nix) pipulate $
```

Book Analysis

Ai Editorial Take

And then the surprising thing the author did next—just when most builders running a niche 24/7 broadcast at N=1 to N=3 concurrent viewers would either despair over small sample sizes or hide behind vanity impressions—was to turn low concurrency itself into a single-channel patch-clamp microscope. In biology, Neher and Sakmann won the Nobel Prize not by averaging millions of noisy cells, but by isolating a single ion channel so every open-and-shut transition became audible; at 10,000 concurrent viewers, individual arrivals blur into statistical mush, whereas at 1-to-2 viewers a soft C5-to-G5 fifth paired with a second audio fader exposes the exact spoken sentence that holds or loses a human mind. Better still, the session proved self-similar: when an overly ambitious AI patch clipped mid-flight, the author immediately applied the exact same anti-jitter governor to his code actuator (apply.py), bounding patch depth so neither viewer bounces nor LLM forward passes can shake the steering wheel apart.

🎨 Hero Banner Image Prompt

```
A 16:9 widescreen graphic-novel ink-and-wash illustration set in a
deep obsidian-slate (#111111) engineering studio with dramatic
high-contrast chiaroscuro lighting. Vertically centered on a dark
workbench sits an intricate brass cybernetic hourglass-and-marimba
resonator: erratic cyan sparks representing fleeting bounce traffic
brush lightly across two rosewood marimba bars, releasing soft
concentric teal acoustic ripples into the dark studio air while a
smartphone beside them rests completely  ... [line truncated]
```

🖼️ YAML Frontmatter Image Override

```
image:
  path: /assets/images/stop-burning-push-alerts-bounce-traffic.jpeg
  alt: "Ink-and-wash studio scene where a brass marimba filter turns
cyan sparks into soft ripples beside a silent phone."
```

Content Potential And Polish

- **Core Strengths:**
- Clear, relatable opening thesis that reframes alert fatigue around earning dwell time before spending operator attention.
- Rich theoretical grounding connecting Norbert Wiener's 1948 cybernetic signal filtering, Xerox PARC's Calm Technology, MIT's ambientROOM, and spiderweb biotremology directly to stdlib Python code.
- Unimpeachable live before-and-after terminal receipts capturing an actual 20-second silent bounce alongside 328-second and 90-second dwell-verified sessions on the morning livestream.
- Candid engineering pivot rejecting flat .jsonl telemetry files in favor of normalized 3NF WAL-mode SQLite tables joinable across machines via ATTACH DATABASE.
- **Suggestions For Polish:**
- When publishing standalone book chapters, consider condensing the repeated NEXT CONTEXT comment blocks so readers stay focused on the common.py and apply.py

diffs.
- Link forward to the upcoming P0.29 implementation post once telemetry.db and the cross-machine ATTACH DATABASE queries go live so readers can see the SQLite join executing on real bouts.

Next Step Prompts

- Implement P0.29 by creating the normalized WAL-mode SQLite schema in ~/.local/share/pipulate/telemetry.db (articles, episodes, stage_cues, ytwatch_bouts), stamping active stage cue state in remotes/honeybot/scripts/stream.py, and exposing a bounded ytwatch bouts survival summary in scripts/articles/common.py while removing the stale commented # deleteme.txt line from context.txt.
- Build a read-only rsync snapshot helper to pull Honeybot's honeybot.db onto Prime and verify an ATTACH DATABASE query correlating 60-second YouTube dwell bouts (in_here) against ?src=youtube referrals and RFC 9110 Accept: text/markdown fetches (out_there).