

個人で 24 時間 TV 局を回す

放送自動化システム **ICS-TV** のアーキテクチャ

CasparCG × Django × gRPC で作る、止まらないリニア配信

WHO

自己紹介

八神 翔大

やがみ しょうた

X (旧Twitter) @shotayagami

misskey @shotayagami@misskey.io

「よめください」

経歴

- 港湾コンテナターミナルシステムの運営に 16 年。
- かたわら同人サークル運営で、自宅サーバ構築から Web 開発まで。
- その技術で 10 年前に Web システム開発へ転向。
- ~2026年3月: 幼児教育教室を運営する会社で社内システム開発。
- **現在: 新規事業のために用意した会社の“雇われ経営者”。**

開発のきっかけ —— 「そういえば VTuber やってたなぁ」

- 2018～2023年、なぜか VTuber 活動っぽいことをやっていた → YouTube チャンネルが既にあった。
- 2023年から会社勤務で時間がなくなり、3年ほど放置。
- 収益化されていたのに、剥奪。
 - 半年放置すると収益化は消える…。
- でも過去資産は山ほどある —— 活用しない手はない。

→ 「**24時間 再放送する仕組みがあれば、できるんじゃない？**」

番組表を編集すると、あとは人手ゼロで流れ続ける

- ICS-TV は、個人で運用する 24 時間自動放送局。
- YouTube Live へ 24/7 連続送出。番組・CM・テロップを自動で送り出す。
- やることは「番組表に番組を置く」だけ。
 - 録画番組も生中継も、CM 挿入も、フィラー(隙間埋め)も、放送休止(クロージング)も全部自動。
- 放送局のマスター運行に相当する監視・自動退避・障害復帰まで実装。
- 本番稼働中 (v0.8.75)。要件定義から約2週間で初版・個人開発・コードは ~69K 行。

「OBS で手動配信」 から 「無人の放送局」 へ

- 出発点: OBS を人が操作して YouTube に流す —— 人がいないと放送できない。
- **目標: ウェザーニュース LiVE / SOLiVE と同じ「リニア(線形)チャンネル」構造。**
 - 連続した1本の放送の上に、番組表・CM枠・配信枠を論理的に重ねる。
 - YouTube = 無料公開、自社経路(Cloudflare) = 将来の有料、の二階建て。
- **放送は「止まったら事故」 —— 24/7 で落ちない設計が全ての前提になる。**

PART 1

全体アーキテクチャ

制御プレーン / 送出ノード / クラウド —— 役割で割った3層

役割で割った3層 —— 疎結合で繋ぐ

制御プレーン

自宅 Proxmox / RKE2

- Django + PostgreSQL
- 番組編成 UI・番組表公開
- スケジューラ (Celery)
- as-run を解決して push
- 営放 / 納品 / 課金 / 計測

送出ノード

自宅 LXC (CT131)

- playout agent (Python)
- CasparCG (Linux/GPU)
- MediaMTX (SRT/RTMP終端)
- ローカル SSD キャッシュ
- 数十時間分を自律実行

クラウド配信

Cloudflare / YouTube

- CF Live Input → Output
- YouTube 永続キーへ連続送出
- Data API で 4h rolling
- 自前HLS (MediaMTX→nginx)
- 素材は R2 (egress無料)

gRPC で as-run を push / AMCP で CasparCG を駆動 / RTMP・SRT で生入力 / HLS で配信

主要コンポーネントと技術

コンポーネント	技術	役割
制御プレーン	Django + PostgreSQL + Celery	編成・スケジューラ・業務ロジック・公開Web/API
送出 agent	Python (asyncio) + gRPC + SQLite	as-run をローカル実行・store-and-forward
送出エンジン	CasparCG (Linux/GPU) + AMCP	映像合成・テロップ・iGPU h264_vaapi エンコード
生入力	MediaMTX + Cloudflare One/WARP	現場 SRT を公開IPなしでローカル終端
素材	OMV → FFmpeg正規化 → R2	解像度/fps/コーデック/-14LUFS を統一
配信	YouTube Data API + 自前HLS	永続キー連続送出・4h rolling・自前ABR
フロント	React 島 + Django-ninja + Bootstrap 5	公開=SSR島 / studio=SPA / ops=放送コンソール
デプロイ	RKE2 + ArgoCD + kustomize	単一imageを多workloadへ git 駆動展開

設計の背骨 —— 「編成」と「送出」を分ける

- 編成 (人が組む): 番組を任意時刻に置くタイムライン。自由で、いつでも書き換わる。
- 送出 (機械が実行する as-run): フレーム単位の実行命令列。決定的で、不変。



- ▶ 分離の効能: 営放(広告)も納品も resolver へ依存逆流しない。送出層は編成を知らない。
- ▶ 業務系が落ちても放送は止まらない —— 切り離せるから 24/7 が成立する。

山場 ①

YouTube 上でリニア ch を回す

途切れたら終わり、を逆手に取る

リニアの逆説

- YouTube Live / CF Live Input は、フレームが途切れない連続入力を要求する。
 - 止めたら配信が壊れる。番組ごとにストリームを張り直す、は不可能。
- **発想の転換:**
- **チャンネルごとに「常時稼働の送出を1本」持ち、その上を番組表で切り替える。**
 - これは放送局の linear playout そのもの。番組・CM・配信枠は連続ストリーム上の論理的な区切りにすぎない。
- **CasparCG を永遠に回し続け、約30秒～分単位で「上に載るもの」だけを差し替える。**

編成 → as-run の解決 (リゾルバ)

人が置いた **編成** を、リゾルバが **as-run** (不変な実行命令列) へ解決する。

① **編成** — 人が任意時刻に番組を置く



② **as-run** — [now, +48h] を冪等に解決

```
def resolve(channel, t0, t1):    # t1 = now+48h
    cursor = t0
    for p in programs_in(channel, t0, t1):
        if p.start_at > cursor:    # 隙間
            events += emit_filler(cursor, p.start_at)
        events += (emit_recorded(p) if p.recorded
                    else emit_live(p))
        cursor = p.end_at
    upsert_playout_events(events) # ikey で diff
    push_to_agent(channel, events)
```

隙間→フィラー / CM枠→15・20sグリッド自動充填 / 生→cutのみ事前生成。重なりは **EXCLUDE USING gist** がDBで拒否、実行済みは不変・未来分のみ差替。

継ぎ目のない切替 (LOADBG → PLAY)

- agent は scheduled_at の数秒前に背面ロード、定刻でテイク —— これでフレーム精度カット。

① LOADBG

次クリップを
裏レイヤへ先読み
PREROLL ≈ 5s 前

② PLAY

定刻 (NTP同期) に
テイク=切替
TICK = 100ms

③ filler loop

次イベントの
until まで再生
黒画面を作らない

- ▶ 素材は事前に正規化 (解像度/fps/コーデック/音声 -14 LUFS) —— 継ぎ目で映像も音量も乱れない。
- ▶ CG (Lバー/速報/提供) は別レイヤに常駐。背面が差し替わってもテロップは出っぱなし。

放送品質のこだわり ―― 安全領域と「適応する」CG

- テロップ/時計は画面端ギリギリに置けない。ARIB TR-B4 の使用可能領域(アクションセーフ 97.5%)の内側に収める。
- 朝・夕の時間帯だけ左上に時計を常時表示(daypart)。「放送局らしさ」は細部に出る。
- **CG同士が衝突しないよう「適応動作」させる:**
 - 時計を出す間は L バー内の時計を OFF、予告は右へオフセット(clockAvoid)。
- ▶ CGは送出と別レイヤに常駐(OverlayManager)。背面クリップが差し替わってもテロップは独立して出続ける。

CG レイヤ構成 (一例)

- L10** 本線 (番組/CM/フィラー)
- L30** L バー + バー時計
- L35** 予告 (clockAvoid 右寄せ)
- L36** 朝夕時計 (daypart)
- L40** 速報
- L41** 速報チャイム (音源ライブラリ)
- L45** フリーテロップ
- L50** 提供クレジット

YouTube への送出 —— 連続性と枠分割を分離

- CF Live Output が YouTube の永続ストリームキーへ「連続送出」。送出は一度も切れない。
- その上で Data API が 4時間ごとに liveBroadcast を rolling 生成し、
 - testing → live → complete を順送りでトランジション。
- **物理的な送出の連続性と、4h という枠分割は、独立に成立する。**
 - ▶ 送出は CF へ1系統だけ → CF からファンアウトで自宅の上り帯域が半分。

実機: チャンネル設定 (ch2)

channel: ICS-TV 教育 ch2

YouTube 連携 (OAuth)

{# YouTube Data API 利用開示 (Google OAuth ブランディング要件)。API を実際に使うのは運用者 (本画面) なので視聴者向け公開トップではなく、OAuth 連携を行う本カードに掲示する。privacy/terms は公開ポスト側の絶対 URL。#}

本サービス (ICS-TV) は、配信枠 (YouTube ライブ放送) の作成・管理のために YouTube Data API を利用します。取得データの取り扱いについては [プライバシーポリシー](#)・[利用規約](#) をご覧ください。

● 接続済み

access_token expiry: 2026-06-17 04:42 (JST) - 約1hで自動更新 (refresh_token 使用)

scopes: <https://www.googleapis.com/auth/youtube.force-ssl>

再認可

永続 liveStream (ingest 受信口)

livestream_id: [REDACTED]

ingest_url: <rtmp://a.rtmp.youtube.com/live2>

stream_key: ***マスク***

既存。再作成は Django admin で channel.youtube_livestream_id を NULL にしてから【作成】。

YouTube 枠 (2h×12 rolling)

枠ダッシュボードを開く

Celery beat が 10 分周期で rolling 生成し、1 分周期で transition(live/complete) する。

永続 liveStream + 4h rolling (Celery: 10分生成 / 1分 transition) + Cloudflare Live。

二階建ての下半分 —— 自前HLSで「自社配信」も持つ

- 公開プレイヤーは YouTube 埋め込みではなく、自前HLSで配信。将来の有料配信(サブスク)の足場。
- ▶ 送出 encoder の tee から、同じ映像を2系統へ分岐する ——

① YouTube 経路

無料公開

- ・ CF Live Input → 永続キーへ連続送出
- ・ Data API が 4h 枠を rolling 生成
- ・ 視聴者規模はここで稼ぐ

② 自前HLS 経路

自社・将来有料

- ・ MediaMTX で clean な h264 を終端
- ・ 独立 unit が 720p/480p/144p+音声のみ ABR
- ・ tmpfs→nginx→Cloudflare、React Player で再生
- ・ 画質切替 / 音声のみ / PiP / 続きから

両経路は独立 —— 自前HLS ladder が落ちても YouTube 送出は無傷。送出ノードの容量(30fps)とは綱引き。

山場 ②

WAN 断でも止まらない agent

制御が落ちても、放送は続く

自宅集約という制約 → ローカル自律

- 制御プレーンも送出ノードも自宅。両者を繋ぐ WAN は、いつか必ず切れる。
- **agent が as-run を「数十時間分」ローカル SQLite に保持 —— 実機では queue = 693 件 (seq 62026)**
 - - 制御プレーンが落ちても、agent はローカルキューで放送を続行（隙間はフィラー）。実績は store-and-forward で WAN 復帰時に返送。
 - ▶ 広告の放送回数も「実送出確定時」に確定 —— 欠送を過大計上しない。

通知センター / ICS-TV 教育 (ch2)

← 運行ダッシュボード

未確認をすべて既読

severity	kind	message	発生	状態
INFO	agent_recovered	agent 心拍が復帰	06/19 07:49:32	既読
CRIT	agent_offline	agent 心拍途絶 (>90s)。送出はローカルバッファで継続中	06/19 07:48:32	既読

↑ 本番ch2 通知センター: 心拍途絶(>90s) を検知も「送出はローカルバッファで継続中」 → 約1分後に自動復帰 (07:48:32 → 07:49:32)。

冪等な as-run —— 再起動・再送に強い

冪等キーは (channel・時刻・action・seg) から決定的に算出:

```
ikey = uuid5(NS, f"{ch}:{scheduled_at}:{action}:{seg}")
```

- 再解決しても同じキー → upsert が「自然な差分」。実行済/実行中は不変、未来分のみ差替。
- agent は実行済キーを SQLite に永続化 → 再起動・WAN復帰後は再 pull で実行済をスキップ。
- 削除は tombstone (CANCELLED) として配信 —— カスケード失敗を作らない。
- **結果: 二重送出不しい・取りこぼさない。実行層をほぼステートレスに保てる。**

実機: AS-RUN ライブログ

AS-RUN ライブログ (新しい順)

時刻	action	status	番組	note
22:13:12	play_filler	scheduled	-	
22:08:12	play_filler	cancelled	-	
22:07:26	play_filler	scheduled	-	
22:03:12	play_filler	cancelled	-	
22:02:26	play_filler	cancelled	-	
22:01:10	play_filler	scheduled	-	
21:58:12	play_filler	cancelled	-	
21:57:26	play_filler	cancelled	-	
21:56:10	play_filler	cancelled	-	
21:56:00	play_filler	scheduled	-	
21:53:12	play_filler	cancelled	-	
21:52:26	play_filler	cancelled	-	
21:51:10	play_filler	cancelled	-	
21:51:00	play_filler	cancelled	-	
21:50:52	play_filler	scheduled	-	
21:48:12	play_filler	cancelled	-	
21:47:26	play_filler	cancelled	-	
21:46:10	play_filler	cancelled	-	
21:46:00	play_filler	cancelled	-	
21:45:52	play_filler	cancelled	-	

scheduled と cancelled(=tombstone) が冪等キーで差分反映。

3層ウォッチドッグ + 自動退避・復帰

ウォッチドッグは3層で、重複なく分担する:

- **ホスト層** watchdog.sh (systemd 30s) — プロセス死活の最後の砦
 - **agent層** feed monitor (asyncio) — feed断を2秒で検知 → 自動スレート退避
 - **server層** 死活 beat (Celery) — heartbeat 途絶 90s で通知
- ▶ 生フィード断 → 2秒で「しばらくお待ちください」へ自動退避。復帰は誤爆防止のヒステリシス(10s連続)で本線へ戻し、フラップ制限(10分3回)も持つ。
- ▶ ハマった盲点「AMCP健全 ≠ 出力健全」—— 出力経路にも watchdog を足した。

実機: 運行ヘルス (ch2)

ヘルス	
agent	ONLINE
最終 heartbeat	19:14:23
CasparCG	healthy
feed	idle
slate	off
自動復帰	ON OFFにする
queue	693 (seq 62026)
レイヤ状態	
1-10 main	ON filler/2
1-20 bumper	—
1-30 lbar	ON
1-35 preview	—
1-40 breaking	—
1-45 freeform	—
1-50 sponsor	—
1-90 slate	—
YouTube 枠	
状態	live
窓	18:00-20:00

PART 3

放送局を「丸ごと」一人で

送出だけでなく、観る人・売る人・作る人ぶんまで1システムに

公開プロダクト —— Django MVC から React + API へ

- 公開フロントは Django SSR に React「島」を載せる方式。studio管理は React Router の真SPA。
 - 島↔サーバは Django-ninja API(/api/v1)。認証は既存 session + CSRF を踏襲 (JWT 無し)。
 - プレイヤー島: hls.js で画質切替/音声のみ/PiP、heartbeat(/api/beat)、WS ライブコメント。
 - UI 基盤は Bootstrap 5 へ全面移行 —— 自作 DS 2本を撤去しテーマ束 (@icstv/theme)に一元化 (v0.8.61~63)。
- ▶ モバイル対応はトークン不変で @media のみ。EPGは二重スクローラを避け横スクロール化。

```
frontend/  
  apps/  
    home player guide  
    discover detail web ← 島  
    studio ← SPA  
ops (放送コンソール) ← 独立 SPA  
  packages/  
    @icstv/theme      Bootstrap5 テーマ束  
    @icstv/api        ninja生成TS
```

観る人向け —— 会員・サブスク・年齢制限

会員 (Member)

メール + TOTP 2FA

- ・ Django User と分離した視聴者会員
- ・ 未認証はゲートで弾く
- ・ 統計用に月精度の生年など平文最小限
- ・ 退会/メール変更/PWリセット

サブスク (Stripe)

松竹梅ティア

- ・ ホスト型 Checkout + webhook で状態同期
- ・ 特典4種=広告なし/HD/コメント/限定
- ・ 現状は コメント特典・限定 を enforce
- ・ 特商法ページも内製

年齢制限 (BILL-02)

PG12 / R15 / R18

- ・ Series→Program で rating を解決
- ・ Member.age と min_age を突合
- ・ VOD=403 ハードゲート(anon→login)
- ・ リニアは共有配信ゆえバッジのみ

「年齢 > 可視性」で解決。リニアは視聴者別に止められないのでバッジ表示に留める —— 媒体特性に正直に。

放送コンソール (ops.*) — 運行操作を独立 SPA へ

v0.8.36～38 のリファクタで、運行操作を「放送コンソール」として独立した SPA / ホスト (ops.*) に切り出した。

旧: studio 管理画面の一部だった運行操作が、放送を担う ops ドメインに分離。

放送コンソールが担う操作:

- 押え / 巻き (即時開始・終了)
- 生入力 / SRT 受け入れ開始・停止
- YouTube 配信枠 (liveBroadcast) 手動遷移
- フル CG モード切替
- 速報テロップ発射・チャイム選択 (音源ライブラリから即時発火)
- 配信枠 (YT) リアルタイム状態カード

▶ 責任境界: studio は「編成・素材」/ ops は「いま流れているものをコントロールする」。

▶ 実装: @icstv/ops パッケージ + ops.home.yagamin.net (別ホスト)。

規模と現状

v0.8.75

本番稼働中
(YouTube ライブ)

~69K 行

サーバ48K + agent6K
+ frontend15K / proto

798

コミット
約2週間で初版 / 個人開発

1 ch

制御は多ch対応済
ノード容量の壁で1ch集中

- ▶ 稼働: 自宅 Proxmox の RKE2 (制御) + LXC (送出) + Cloudflare/YouTube。Zabbix で監視。
- ▶ 実装済: 録画/生/CM・正規化・営放(広告)・納品QC・請求/予算・権利・視聴計測・WSコメント・会員/2FA・サブスク課金・年齢制限・自前HLS・React島/studio SPA・放送コンソール(ops)・放送時間帯・見逃し配信VOD・Remotion動画スタジオ・Playwright E2E。
- ▶ 真の残: DVR追っかけ再生・字幕(Whisper)・出力consumer凍結検出・1080p化・2ch同時(=送出ノード容量)。

放送の周りも全部 —— 周辺サブシステムまで個人で

- 放送を成立させる“裏方”も、同じ1システムに:
 - 納品QCポータル — 外部制作会社が入稿、技術+R128ラウドネス+リーダー検出を自動QC、回(Episode)単位で確定。
 - 営放(広告) — タイム/スポット契約・競合排他・放確台帳。
 - 請求 — 放送確認書/請求書を内製PDF、番組予算(AP)も管理。
 - 視聴計測 / ライブコメント(WS) / 会員 / サブスク課金 / 権利管理。
- ▶ 納品・経理予算は ICS-DELIVERY / ICS-BACKOFFICE として本体から分離済み (Node.js・別 DB・別リポ)。
- ▶ どれも送出系と疎結合 —— 落ちても放送は止まらない。

実機: 納品QCポータル

納品	対象	チャンネル	番組 / 回	本編(放送)	配信	番組	番組	保管
【#ForzaHorizon5】 何となく深夜のドライブ 【アイシーエス #八神翔太】 社内制作	単発番組	-	-	approved v1	-	-	-	-
【#アーケードアーカイブス #ニューマンアスレチック】 30年ぶりにやるスポーツゲーム 【アイシーエス #八神翔太】 社内制作	単発番組	-	-	approved v1	-	-	-	-
【#スプラトゥーン3】 グランドフェスらしいのでサザエ集め! 【アイシーエス #八神翔太】 社内制作	単発番組	-	-	approved v1	-	-	-	-
【#PICROSS S】 イカ以外にも塗るゲームありました 【アイシーエス #八神翔太】 社内制作	単発番組	-	-	approved v1	-	-	-	-
【東方】 あすをボム #8 「弾幕ごっこは原点回帰できたか」 【手書き】 社内制作	単発番組	-	-	approved v1	-	-	-	-
【東方】 あすをボム #7 「盆栽と仏教の相関」 【手書き】 社内制作	単発番組	-	-	approved v1	-	-	-	-
【東方】 あすをボム #6 「這いよるうなぎの遡洄」 【手書き】 社内制作	単発番組	-	-	approved v1	-	-	-	-
【東方】 あすをボム #5 「旧地獄の灼熱地獄」 【手書き】 社内制作	単発番組	-	-	approved v1	-	-	-	-
【東方】 あすをボム #4 「イージーノバルによる娯楽の変化」 【手書き】 社内制作	単発番組	-	-	approved v1	-	-	-	-
【東方】 あすをボム #3 「紅魔館紅茶異変」 【手書き】 社内制作	単発番組	-	-	approved v1	-	-	-	-
【東方】 あすをボム: 幻想郷を席巻するスキママーケティング 【手書き】 社内制作	単発番組	-	-	approved v1	-	-	-	-
【東方】 あすをボム: 博麗神社債務不履行問題 【手書き】 社内制作	単発番組	-	-	approved v1	-	-	-	-

新規納品

{# 納品メタデータ入力フォーム (新規作成 / 編集 共用)。delivery=None で新規、delivery 指定で current.target_kind で対象ラジオを初期選択。}

タイトル (必須)

納品の対象
☒ 週間編成番組の特定回
 ☐ 単発番組
 ☐ CM

番組 (Series)

放送予定日 回数 (任意)
 年 / 月 / 日

放送予定日 から 回数 で回を特定/新規作成します。既存の回を表示の場合は下記。
 既存の回から選ぶ (任意・指定時は上記より優先)

— (新規作成)

制作会社 (どこが・空=社内制作)
 — (社内制作)

作成してアップロードへ

← 納品マトリクス

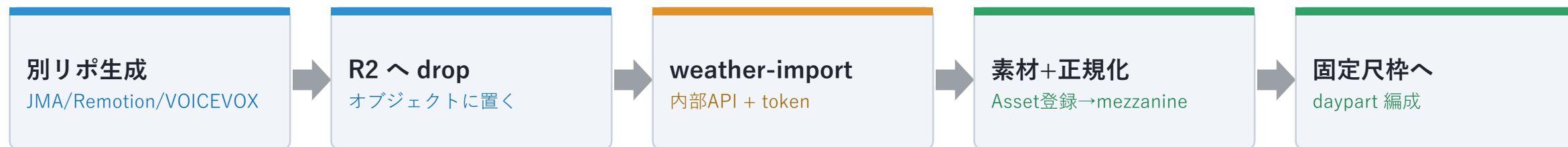
本編=approved が多数

← 新規納品フォーム

回(Episode)を指定して入稿

外部連携の実例 —— 天気予報の自動取り込み (#22)

- 別リポの生成側(JMA予報 + Remotion + VOICEVOX)が天気動画を作り、R2 に置く。
 - ICS-TV は内部API /internal/weather-import をトークン保護で公開。R2キーを受けて素材登録→正規化。
- 編成は特定時刻の固定尺枠(daypart)へ自動で嵌め込む —— 生成は当日朝にまとめて。
- **肝:** 編成も送出も中身を一切知らない。「素材を1本足す」だけで放送に乗る —— 疎結合がそのまま外部連携に効く。



新しい外部システムが増えても、ICS-TV 側は「素材入口」を1つ開けるだけ。送出系には一切手を入れない。

マルチリポ化 — 業務系を本体から切り出す

ICS-TV (Django) から業務ロジックを段階的に別リポへ抽出。接点は X-Internal-Token 保護の内部 API のみ。

ICS-DELIVERY (Phase 2 / Node.js・別 DB)

- B2B 納品ポータル: アップロード・QC・正規化・進行表
- seam: POST /internal/delivery-asset で完成 asset 登録 / GET /internal/delivery-refs で参照

ICS-BACKOFFICE (Phase 3 / Node.js・別 DB)

- 経理請求・番組予算・権利・会員分析のバックオフィス
- read-API-first: 当面は ICS-TV を真実として読み取り集約、書き込み移管は後半フェーズ

ICS-WEATHER / ICS-EARTHQUAKE (別リポ / Python)

- 生成系: JMA 予報+Remotion+VOICEVOX → R2 → seam で取り込み
- 地震速報: 主系(NHK/JMA)+副系 ポーリング → breaking-telop 内部 API

▶ ICS-TV コア (Django) は「編成・送出・公開」に集中。本体が送出系に強い理由はここにある。

つまずき —— 「2ch同時」の壁

- 制御プレーンは最初から多ch設計(Channel.enabled)。検証で 2ch目「教育」を立ち上げた。
- **が、送出ノードが容量飽和 —— 4コアLXC + 単一iGPU + k8s-worker 同居で、2系統の合成+エンコードは重すぎた。**
 - 真因の一つ: YouTube が CDN で 1080p をハード要求、encoder は 720p60 → ingestion starved でバッファリング。VBITRATE 引き上げで暫定回避。
- 判断: ch2 を Channel.enabled=False で停止し ch1 に集中(公開非表示/YT枠生成/resolver を停止)。
- **学び: 「動く」と「同時に2本回る」は別問題。壁はコードでなくノード容量に出た。恒久策=GPU正規化・1080p化・ノード増設。**

24時間配信は、実際に回って・見られている

321

視聴回数 / 週

17.3h

総再生時間 / 週

1,183

登録者 (累計)

203

視聴回数 / 48h



収録時点・直近7日のアナリティクス（無人運用）。視聴が右肩上がりで積み上がる。

チャンネルのコンテンツ

インスピレーション 動画 ショート **ライブ配信** 投稿 再生リスト ポッドキャスト コース プロモーション コラボレーション

フィルタ

	ライブ配信	タイプ	公開設定	制限	日付	視聴
☐	ライブ配信					
ライブ配信中						
☐		ICS-TV 総合 2026-06-19 18:00-20:00 (JST) の配信枠です。	ストリーミングソフトウェア	公開	なし	2026/06/19 ライブ配信中
☐		ICS-TV 教育 2026-06-19 18:00-20:00 (JST) の配信枠です。	ストリーミングソフトウェア	公開	なし	2026/06/19 ライブ配信中
配信予定						
☐		ICS-TV 総合 2026-06-20 18:00-20:00 (JST) の配信枠です。	ストリーミングソフトウェア	公開	なし	2026/06/20 公開予約
☐		ICS-TV 教育 2026-06-20 18:00-20:00 (JST) の配信枠です。	ストリーミングソフトウェア	公開	なし	2026/06/20 公開予約
☐		ICS-TV 教育 2026-06-20 16:00-18:00 (JST) の配信枠です。	ストリーミングソフトウェア	公開	なし	2026/06/20 公開予約
☐		ICS-TV 総合 2026-06-20 16:00-18:00 (JST) の配信枠です。	ストリーミングソフトウェア	公開	なし	2026/06/20 公開予約

収録時は「総合」「教育」2ch同時ライブ + 翌日枠を自動「配信予定」生成。現在は容量都合で1ch集中(前述)。

地震速報の自動割り込み & 週間グリッド編成エディタ

【地震速報サブシステム (v0.8.29 / icstv-earthquake)】

- Celery beat が 15 秒間隔でポーリング (主系: NHK/JMA, 副系: P2P地震情報)
 - 最大震度・気象庁コードから通報レベルを分類 → EarthquakeAlert 台帳へ登録
 - 内部エンドポイント /internal/breaking-telop を叩いて速報テロップ + チャイムを自動発射
 - 放送コンソールから任意タイミングの手動発射も可能
- ▶ ICS-TV 側は「speed-telop の入口を1つ開けるだけ」— 送出系には一切手を入れない。

【週間グリッド編成エディタ (v0.8.49)】

- 曜日 × タイムライン の 2D グリッド上でシリーズ枠を D&D 配置・リサイズ
- 繰り返し放送 (SeriesSlot) を週単位で一括管理 → 同じシリーズを毎週同じ時間に置ける
- API: slot_create / slot_move / slot_resize_grid + 週読み (WeekOut/WeekSlotOccurrence)
- Studio SPA の WeekGridPage として実装。モーダルからシリーズをインライン作成も可。

UPDATE v0.8.51 → v0.8.75

発表3日前からも 24 リリース — 数字に向き合い編成を変えた

【急激な登録者減 → 編成を見直す (v0.8.52 / v0.8.70)】

- 24h 連続配信の裏で登録者が急減。ヒアリングでは「配信開始通知が山のように届く」の指摘
- 対策① YouTube 枠を 2h → 4h へ (v0.8.52) — 通知頻度を半減
- 対策② 放送時間帯を導入 (v0.8.70) — 現有コンテンツで 24h は戦いきれない。やむなく休止を選択
- ch1 は 06:00–10:00 / 16:00–24:00 で運用中。休止帯は休止スレート・YT 枠なし — 送出 1 本は連続のまま

【新機能も同時進行】

- 見逃し配信 VOD (v0.8.72) — 生放送を自動録画 (MediaMTX) → R2 署名 PUT → VOD 公開
 - Bootstrap 5 全面移行 (v0.8.61～63) — 自作 DS 2本を撤去、Playwright E2E を CI に追加
 - Remotion 動画スタジオ (v0.8.65～66) — タイトルカード/Lサードをブラウザで Props 編集
- ▶ 順風の話ではない — 実運用の数字と向き合って編成を変えた。それでも送出は一度も止まっていない。

持ち帰ってほしい4つ

- リニアの逆説: 「常時1本の送出」を持てば、放送局型の自動編成は個人でも作れる。
- 編成と送出の分離: 人が書く編成と、機械が実行する不変な as-run を分ける。業務系を切り離せる
- 冪等 as-run + store-and-forward: 決定的キーで実行層をほぼステートレスに。
クラッシュにも WAN断にも強い。
- ローカル自律 + 多層監視: 制御が落ちても放送は続く。これが 24/7 を成立させる。

ありがとうございました

個人で 24 時間 TV 局を回す —— ICS-TV

CasparCG × Django × gRPC / 本番 v0.8.75 稼働中

視聴はこちら

YouTube youtube.com/@shotayagami (ライブ配信)

ICS-TV tv.yagamin.net (公開サイト: 番組表・自前HLS・見逃し配信)

Questions?