



# 김태윤

Backend Engineer · AI·SW 마에스트로 제17기

github.com/xodbs1021

xodbs9810@gmail.com · 010-8757-0371

사용자가 겪는 문제에서 출발해 제 코드 밖까지 원인을 추적합니다.  
대안의 비용과 한계를 따지고, 측정과 검증으로 설계를 바꿉니다.  
팀이 같은 기준으로 움직이도록 서비스 간 계약을 조율하는  
백엔드 개발자 김태윤입니다.

## 사용자가 겪는 문제에서 출발합니다

좋아하는 스트리머의 방송이 멈춘 순간, 화면 뒤에서 영상이 어떻게 전달되는지 궁금해했습니다. 그 관심으로 시작한 PokeClip에서 3인 팀의 팀장을 맡아 전체 아키텍처와 미디어·인프라 영역의 설계 판단을 이끌고 있습니다. 제가 만들고 싶은 것은 기능이 연결된 서비스입니다. 요청이 성공했는지뿐 아니라 사용자가 원하는 장면을 실제로 볼 수 있는 지까지 확인하려 합니다.

## 전부 만들기보다, 필요한 곳을 깊게 만듭니다

처음에는 미디어 서버 전체를 Go로 만들려 했습니다. 그러나 3인 팀이 완성할 범위와 구현 위험을 따져 수신·기본 녹화는 MediaMTX에 맡기고, 조각 색인·업로드 복구·되감기 시간축에 집중했습니다. 사용 중 드러난 문제는 상류 수정으로 연결했습니다. 제가 작성자로 제안한 PR 4건이 병합되어 v1.21.0~v1.21.1 정식 릴리스에 반영되었습니다.

## 그럴듯한 설명보다 실제 동작을 판단 기준으로 삼습니다

되감기 재생목록은 141.97초였지만 브라우저가 인식한 구간은 80.03초였습니다. 오류 로그 대신 실제 재생을 기준으로, 태그 조정에서 파일 내부 시간축을 연결하는 재포장으로 방향을 바꿨습니다. 파일마다 FFmpeg를 시작하던 경로를 상주 Go 호출로 옮겼지만, 첫 비교에는 Go의 I/O가 빠져 있었습니다. 저는 양쪽의 읽기·쓰기를 포함해 **각 200회 다시 비교**했고 p50 **23.43→3.05ms**를 대표 결과로 삼았습니다. 전체 서비스 지연과는 구분합니다.

## 협업에서는 변경 비용과 전환 순서까지 책임집니다

방송 식별자를 바꿀 때 기존 조각 조회가 깨지는 영향을 먼저 공유하고, 소비자를 담당한 팀원에게 대응 방식을 확인했습니다. 제 제안을 전 서비스에 일괄 적용하는 대신, 팀원의 기존 조회 경로를 유지할 ingestStreamId를 추가하고 소비자 배포가 끝난 뒤 새 값을 발행하기로 합의했습니다. 바꾸는 쪽의 편의보다 쓰는 쪽의 전환 순서를 먼저 맞추는 것이 제가 생각하는 협업입니다.

## 프로젝트와 기여

미디어와 배포 자동화에서 실제 실행·실패·복구의 경계를 다뤘습니다.

---

### PokeClip | 2026.07-현재 · 3인 팀 · 팀장

생방송 중 하이라이트를 찾아 BGM을 제외한 클립으로 만드는 서비스입니다. 팀장으로 전체 아키텍처·미디어·인프라의 설계 판단과 서비스 계약 조율을 맡았습니다. Go 색인·업로드 복구·재생용 파일 생산의 구현과 검증을 진행했으며 발행 계층과 개발 환경 연동은 진행 중입니다. Spring 코어·렌더 워커, 웹·OBS의 주 구현은 팀원 담당입니다.

### MediaMTX | 개인 기여 4건 병합 · 정식 릴리스 2개 버전 반영

GitHub xodbs1021로 제안한 HLS 설정 검증(#6154), 종료 코드 보고(#6155), 대체 화면(슬레이트) 녹화 제어(#6182)는 v1.21.0에, 기본 인증 설정 정규화·깊은 복사(#6205)는 v1.21.1에 반영되었습니다. #6182는 원저자 커밋을 보존하고 static source·설정 재적용 경계를 보완한 기여입니다. 아직 열린 제안과 미병합 종결 건은 릴리스 성과와 구분했습니다.

### Shakedown | SoftBank Hackathon · 2026.10.08-10 구현·검증

저장소 분석을 로컬·AWS 배포로 연결하는 서비스에서 엔진 통합·이미지 생성·선택 구성 적용·DB 복구를 맡았습니다. AI의 생성 범위를 규칙과 카탈로그로 제한하고 AWS 배포·TLS 연결·거부 시험·수동 DB 복원을 검증했습니다. 재부팅 뒤 데이터만 남고 앱이 멈춘 결과를 보고, **자동 기동과 삭제 상태 보존**을 복구 기준에 넣었습니다. **두 번의 실제 EC2 재부팅**에서 데이터 2건의 복구와 DELETE 뒤 미부활을 확인했습니다. 제 PR 7건이 팀 저장소에 반영됐고, 초기 AWS 어댑터·시운전·보안 게이트는 팀원 기반을 확장했습니다.

---

설계 초안·조사·구현·실측 실행·리뷰에 AI를 활용합니다. 저는 요구와 대안을 비교하고, 검증 결과에 따른 선택·확정과 팀의 계약 조율을 맡았습니다.

PokeClip 코드 ↗

MediaMTX 기여 ↗

Shakedown ↗

MediaMTX v1.21.0 공식 릴리스 ↗

MediaMTX v1.21.1 공식 릴리스 ↗

# 기술과 활동

기술 이름보다 직접 설명할 수 있는 역할과 검증 범위를 기준으로 정리했습니다.

| 영역                | 경험을 설명할 수 있는 범위                                                |
|-------------------|----------------------------------------------------------------|
| Go · PostgreSQL   | 파일 색인, 실패 작업 재수거, 상태 조건부 갱신, 테스트 DB 격리                         |
| 미디어               | SRT·LL-HLS 수신/재생 경로, fMP4 트랙·시간축, Go 재포장                       |
| AWS · Docker · CI | EC2·S3·CloudFront, ECR/ECS·ALB·RDS 배포 연결, Docker/systemd 복구·CI |
| DB · 실행 계약        | PostgreSQL·MySQL·MongoDB 연결·TLS·수동 복원, 명시적 HTTP runtime        |
| 서비스 간 계약          | Spring 코어의 조회 조건·이벤트 필드 조율. 소비자 구현은 팀원, 본인 구현 범위는 Go 계층        |

| 학력·활동              | 기간 / 내용                                   |
|--------------------|-------------------------------------------|
| 세종대학교 인공지능학과       | 2026.02 졸업                                |
| AI·SW 마에스트로 제17기   | 2026.05-현재 · 3K팀 PokeClip                 |
| 소프트웨어융합대학 학술제      | 2024.12 · 우수상                             |
| SoftBank Hackathon | 2026.10.08-10 구현·검증 · Emerald 팀 Shakedown |

PokeClip의 Spring 코어·렌더 워커는 팀원 담당입니다. Shakedown의 실제 재부팅은 Node/PostgreSQL, MySQL의 TLS는 mysql2 경로에서 검증했습니다. 여러 환경의 결과를 모든 언어·운영 조건으로 일반화하지 않습니다.

포트폴리오 · 대표 근거 ↗  
Shakedown 기여 ↗

# 김태윤 · 백엔드 포트폴리오

문제의 원인을 좁히고,  
검증 근거로 설계를 바꿉니다.

|                                                                                                                                                                      |                                                                                                                                                                 |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>MediaMTX 공식 릴리스 · 상태와 기여 ↗</p> <p><b>4건 · 정식 릴리스</b></p> <p>판단 · 제품에서 드러난 결함을 공통 경계에서 고쳐 상류 PR로 제안했습니다.</p> <p>v1.21.0 3건 + v1.21.1 1건<br/>#6182는 원 기여를 승계·보완</p> | <p>파일·업로드 복구 · 4조건 검증 ↗</p> <p><b>276파일 · 누락 0</b></p> <p>판단 · 혹 알림을 진실로 보지 않고 디스크 재탐색과 먹등 업로드를 조합했습니다.</p> <p>정상·알림 유실 4조건 · 중복 0<br/>복구 범위와 실패 조건은 본문에 명시</p> |
| <p>재포장 p50 · 같은 I/O 범위 ↗</p> <p><b>23.43 → 3.05 ms</b></p> <p>판단 · 재인코딩이 불필요해, 매번 FFmpeg를 실행하던 경로를 Go 내부 호출로 바꿨습니다.</p> <p>각 200회 · 10회 예열<br/>전체 서비스 지연과 구분</p>     | <p>CDN 원본 미디어 요청 ↗</p> <p><b>1,254 → 28건</b></p> <p>판단 · 시청자별로 나뉜 캐시를 공유하도록 캐시 키와 원본 인증 전달을 분리했습니다.</p> <p>50 VU · 25초 · 각 조건 1회<br/>실험용 비교군과 순차 비교</p>         |

## 전체 구조를 설계하고, 실패가 남는 경계를 확인합니다

PokeClip의 전체 구조·미디어·인프라 판단과 계약 조율을 맡았습니다. SoftBank 해커톤 Shakedown에서는 저장소 분석을 실제 배포·복구까지 연결했습니다.

| 먼저 볼 경험         | 판단을 뒷받침하는 근거                         | 쪽  |
|-----------------|--------------------------------------|----|
| 오픈소스 릴리스        | 4건 병합 · v1.21.0 / v1.21.1 정식 릴리스     | 44 |
| 파일·업로드 복구       | 정상·알림 유실 4조건 · 276파일 회수 · 누락/중복 0    | 8  |
| 재생 원인과 재포장      | 같은 I/O에서 p50 23.43 → 3.05ms          | 13 |
| 클립 구간 조회 계약     | 두 실제 SQL · 8,020개 기대 결과 일치           | 17 |
| CDN 캐시·접근 통제    | 50 VU 원본 미디어 요청 1,254 → 28 · 무서명 403 | 20 |
| Shakedown 배포·복구 | 10.08-10 구현·검증 · AWS 적용·DB 복원·재부팅    | 46 |

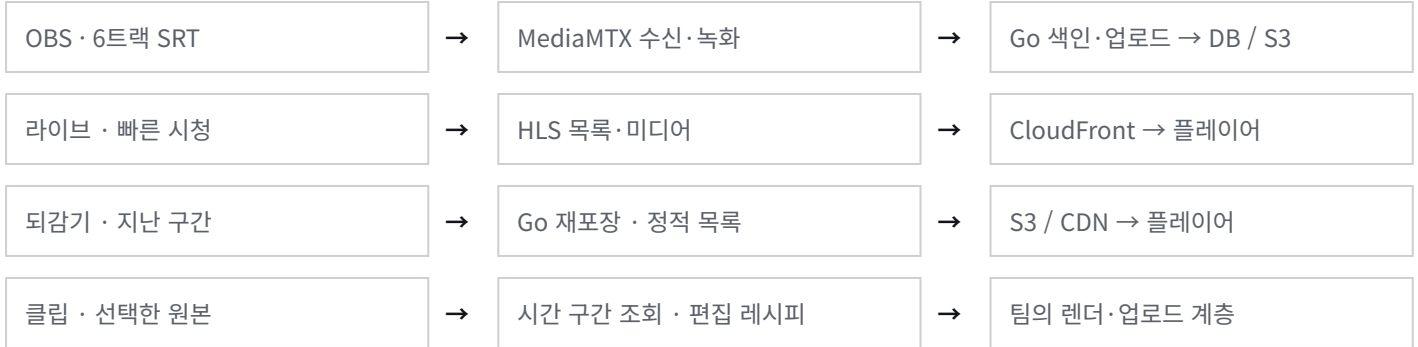
수치는 각 장의 실험 조건에 해당합니다. 본문은 구현·실측과 진행 중인 설계를 구분하며, 상세 원본은 각 사례 링크로 연결했습니다.

GitHub · xodbs1021 ↗  
이메일 · xodbs9810@gmail.com ↗  
전체 목차 · 55-56쪽 ↗

# PokeClip의 구조와 나의 역할

방송 중 지난 장면을 찾고, 원하는 오디오로 클립을 만드는 서비스입니다.

같은 입력에서 사용 목적에 따라 경로를 나눴습니다



## 내가 정한 경계

저장할 때 첫 오디오만 남기면 음악을 제외할 편집 재료가 사라지고, 시청에 6오디오를 모두 쓰면 플레이어가 불필요한 트랙까지 처리해야 합니다. 저는 편집 원본의 6오디오를 보존하고 되감기용은 영상 1 · 오디오 1로 나눴습니다. 클립은 원본 대신 구간 · 크롭 · 트랙 · 자막 레시피를 저장하며, 픽셀을 바꾸는 최종 렌더와 재포장을 구분했습니다.

## 검토 생략을 발견한 뒤 절차를 바꿨습니다

POK-30 설계에서 교차 검토가 생략된 것을 제가 확인하고, 검토 전에는 작업 종료를 막는 게이트를 요구했습니다. 6라운드에서 차단 지적 32건을 코드 작성 전에 해소했습니다. 구현 · 리뷰는 AI 협업이며, 제가 맡은 것은 누락 발견과 검증 절차의 변경입니다.

| 역할    | 담당과 기여                                                        |
|-------|---------------------------------------------------------------|
| 김태윤   | 전체 구조, 미디어 · 인프라의 선택 · 검토 · 확정, 서비스 계약 조율                     |
| 팀원    | Spring 코어 · 렌더 워커, 웹, OBS 플러그인의 주 구현                          |
| AI 협업 | 설계 초안 · 조사 · 구현 · 실측 · 리뷰에 활용. 저는 누락된 재검토를 요구하고 대안 채택과 계약을 확정 |

개별 부품의 병합 · 실측은 완료됐지만, 되감기 발행 계층과 dev-media 연동은 진행 중입니다. 전체 서비스의 상용 운영이나 모든 기능의 단독 구현을 뜻하지 않습니다.

CASE 01 · 실험 조건과 상세 근거 ↗

CASE 03 · 실험 조건과 상세 근거 ↗

CASE 04 · 실험 조건과 상세 근거 ↗

CASE 30 · 실험 조건과 상세 근거 ↗

## 직접 만들 범위를 실험 뒤 다시 정했습니다

기술 이름보다 제품 요구와 팀이 검토할 수 있는 책임을 기준으로 선택했습니다.

### 처음 선택을 바꾼 이유

미디어를 전부 직접 만들던 원안은 3인 팀이 전송·코덱·녹화를 모두 검토해야 했습니다. OBS→SRT→녹화·재생 스파이크가 성립하자, 저는 MediaMTX를 기반으로 채택하고 파일 색인·복구·시간축·회차 관리에 자체 구현을 집중하기로 했습니다.

| 선택               | 비교한 대안과 감수한 비용                                                           |
|------------------|--------------------------------------------------------------------------|
| MediaMTX + Go 계층 | C/C++ 전체 구현의 검토 부담을 줄이는 대신 외부 버전과 동작을 이해·유지                              |
| 서비스별 언어          | Spring은 팀의 검토 역량, Go는 미디어 장기 작업, C++는 libobs 경계. 담당 변경 뒤 채팅도 Spring으로 전환 |
| 미디어 EC2 분리       | UDP 연결·로컬 녹화 상태를 가진 서버를 코어 API와 분리. 교체에는 drain 또는 명시적 중단이 필요             |
| 불필요한 포크 보류       | 6초 목록 창과 실제 시청 지연을 구분. 7조각·GOP 2초의 약 14초 창을 허용하고 녹화 4초는 독립 유지            |

### 내 판단이 바뀐 기준

요구를 만족하는지 확인할 수 있으면 초기 구조도 바꿨습니다. 라이브러리 최소 조각 수를 줄이는 포크보다 창 길이 요구를 재검토했고, 실제 공통 결합은 별도 상류 수정으로 다뤘습니다.

배포·역할 배치의 설계 사례입니다. 언어 선택을 성능 개선으로, dev 구성과 목표 HA 구성을 동일하게 설명하지 않습니다.

CASE 02 · 실험 조건과 상세 근거 ↗

CASE 03 · 실험 조건과 상세 근거 ↗

CASE 04 · 실험 조건과 상세 근거 ↗

CASE 29 · 실험 조건과 상세 근거 ↗

# 완료 알림은 계기, 파일과 DB가 복구 기준

알림이 사라져도 디스크의 파일을 다시 찾아 장부에 등록하도록 설계했습니다.

## 문제

편집기는 특정 방송 시각의 파일을 DB에서 찾습니다. 녹화 완료 혹은 실패하면 파일이 디스크에 있어도 검색 대상에서 빠집니다. 완료 전 강제 종료는 혹 자체를 남기지 않았습니다.

## 내가 선택한 복구 기준

정상 혹은 29/29 최종 크기와 일치하고 p50 2.6ms·p95 4.9ms로 빨랐습니다. 그럼에도 실행 파일 없음·비정상 종료·쓰기 실패는 재시도되지 않았습니다. 저는 빠른 알림과 복구의 기준을 분리하고, 디스크와 DB를 다시 대조하는 구조를 선택했습니다.

| 입력 경로  | 하는 일           | 경계                     |
|--------|----------------|------------------------|
| 완료 혹   | 완성 파일을 빠르게 알림  | 실행 실패·프로세스 종료로 유실 가능   |
| 파일 감시  | 추가·변경된 파일 재발견  | 마지막 파일은 유휴 판정까지 대기     |
| 주기 재스캔 | 디스크와 DB 대조     | 기본 300초 · 탐색 비용과 복구 지연 |
| DB 확정  | 파일 위치·순서·길이 기록 | 파일 중복과 순번 충돌을 구분       |

## 선택한 구조

한 작성자가 순번과 방송 시각을 확정하고, 길이는 파일명이 아니라 fMP4 내부 정보에서 읽습니다. 같은 파일의 재발견은 DB의 고유 제약으로 중복 등록을 막습니다. 파일 크기 안정만으로 완성을 단정하지 않고 후속 파일·유휴 종료·재스캔 등 완성 조건을 나눴습니다.

## 왜 정리 코드보다 복구 경로인가

혹을 더 자주 보내거나 메모리에서 중복을 걸러도 재시작하면 상태가 사라집니다. 파일과 장부를 다시 대조할 수 있어야 실패 뒤에도 수렴합니다. 원본은 읽기 전용으로 붙여 인덱싱 과정이 녹화 파일을 수정하지 않도록 했습니다.

본인 역할: 완료 신호의 신뢰 범위와 복구 구조 선택. 구현·검증은 AI 협업. 파일 회수와 짧은 재접속 경계 복원은 다른 문제이며, 재접속 표시는 혹 유실의 영향을 받을 수 있습니다.

CASE 15 · 실험 조건과 상세 근거 ↗

## 정상·알림 유실 4조건에서 276파일을 회수

빠른 평균값과 마지막 파일의 복구 지연을 나누어 검증했습니다.

정상·알림 유실을 포함한 4가지 조건의 합계

## 276파일 · 누락 0 · 중복 0

흑·파일 감시는 빠른 계기로, 디스크 재스캔은 복구 경로로 분리

| 조건               | 파일 수 | p95      | 최대       | 누락 / 중복 |
|------------------|------|----------|----------|---------|
| 흑 + 파일 감시        | 90   | 0.067초   | 0.123초   | 0 / 0   |
| 흑 유실 · 감시만       | 90   | 0.073초   | 13.526초  | 0 / 0   |
| 둘 다 유실 · 5초 재스캔  | 90   | 4.973초   | 5.019초   | 0 / 0   |
| 둘 다 유실 · 기본 300초 | 6    | 300.014초 | 300.024초 | 0 / 0   |

### 어떻게 확인했나

2026.10.06, 실제 main 타이머 → StartCollect → ApplyCollect → PostgreSQL commit 경로를 실행했습니다. 완료 fixture 파일을 50ms 간격으로 주입하고 mtime을 60초 이전으로 설정했습니다. 앞의 세 조건은 각 3회, 기본 300초는 1회입니다. 합계 276파일이 등록됐고 누락과 중복 commit은 0이었습니다.

### 빠른 평균이 가린 마지막 파일

흑이 없을 때 감시 경로는 대부분 빠르게 찾았지만, 각 실행의 마지막 파일은 유틸 판정까지 기다려 최대 13.526초가 걸렸습니다. 두 신호를 모두 제공하지 않으면 재스캔 주기가 복구 지연을 결정했습니다. 5초는 가속 시험이고 제품 기본값은 300초입니다.

### 설계 판단에 남긴 것

흑의 속도와 복구의 완전성을 별개로 봤습니다. 재스캔 간격을 줄이면 복구는 빨라지지만 디렉터리 탐색 비용이 늘어 납니다. 현재 근거는 파일 등록의 회복이며, 스캔 규모별 비용이나 운영 복구 SLO까지 증명한 수치는 아닙니다.

Apple M5 · Go 1.26.5 · PostgreSQL 17. 실제 OS 알림 장애가 아니라 알림 채널을 제공하지 않은 결함 주입입니다. 실제 방송·S3 업로드·커서 이전 구멍 복구는 이 시험에 포함하지 않았습니다.

### 짧은 재접속은 시간차만으로 찾지 않았습니다

흑 ON/OFF 대조에서 DB 시간차는 574/583ms로 모두 기존 1.5초 기준 미만이었지만, ON에서만 다음 조각 seq 5에 불연속 표식이 남았습니다. 서버의 offline→online 공백 0.886초는 DB 시간차와 별개입니다. 정상 120초는 30파일=30행, 흑 OFF 롤백도 10파일=10행으로 수집을 유지했습니다.

CASE 15 · 실험 조건과 상세 근거 ↗

## S3 성공과 DB 완료 사이에서 죽는다면

업로드의 성공 응답 대신 DB 상태를 재시작 기준으로 삼았습니다.

### 문제

S3에 파일을 보낸 직후 프로세스가 종료되면 객체는 존재하지만 DB는 pending일 수 있습니다. 메모리 큐를 완료 기록으로 사용하면 재시작 뒤 무엇을 다시 해야 하는지 알 수 없습니다. 오래된 실패 작업이 새 방송의 업로드를 막는 상황도 구분해야 했습니다.

### 내가 나눈 실패와 대가

완료의 기준은 큐 접수가 아니라 DB 상태로 두었습니다. 같은 파일은 같은 객체 키로 재시도하고, 예상 상태가 맞을 때만 갱신하는 CAS를 선택했습니다. 제한된 큐와 순회 커서로 오래된 실패 작업이 새 파일을 독점하지 않게 했으며, 그 대가로 DB 조회·중복 시도 비용을 감수했습니다.

| 실패 지점                | 다시 시작하는 기준                      |
|----------------------|---------------------------------|
| 파일 등록 후, 업로드 전 종료    | DB의 pending 행을 스윙퍼가 재수거         |
| PUT 성공 후, DB 확정 전 종료 | 같은 S3 키로 재시도하고 예상 상태·크기 일치 시 확정 |
| 오래된 미완료 작업 누적        | 커서를 이어 읽고 제한된 큐로 접수             |
| 허용 루트 밖의 파일 경로       | 업로드 전 경로 경계에서 차단                |

| 근거               | 확인한 결과                                     |
|------------------|--------------------------------------------|
| 기존 60초 송출        | 15파일 = uploaded 15행 = S3 key 15개           |
| 2026.10.05 접수 검증 | 128 / 1,100 / 4,096건 → 1 / 9 / 32회차 · 각 5회 |
| PostgreSQL 통합 검증 | 6개 시나리오 × 5회 = 30회 통과 · skip 0             |

10/5 접수 시험은 가짜 DB 목록·큐 4,096 조건입니다. 통합 시험은 실제 PostgreSQL·재포장과 가짜 PUT을 사용했습니다. 9회차를 9초로 읽거나 실제 AWS 장애의 복구 시간으로 일반화하지 않습니다.

CASE 16 · 실험 조건과 상세 근거 ↗

## 검은 화면에 겹친 세 원인을 단계별로 나눴습니다

한 곳을 고친 뒤에도 증상이 남으면, 다음 단계의 증거로 조사 범위를 옮겼습니다.

### 내 판단 · 한 단계의 성공으로 끝내지 않습니다

송출 인증과 일부 HTTP 요청은 성공했지만 화면은 검었습니다. 저는 서버의 영상 해석, 재생목록의 실제 객체, 브라우저 초기화를 별도로 확인하는 순서로 접근했습니다. 서버 오류가 사라졌다는 사실을 재생 성공으로 간주하지 않았습니다.

| 관측한 증거                         | 조치와 다음 확인                                                       |
|--------------------------------|-----------------------------------------------------------------|
| too many reordered frames (11) | 당시 OBS를 baseline·키프레임 2초로 조정 → 반복 오류 소멸                         |
| 목록 앞 gap.mp4 7개 · HTTP 401     | 창을 900→30으로 잠시 줄여 오래된 GAP을 밀어낸 뒤 0개 확인<br>· 900 복원              |
| variant·조각 200인데 정지            | hls.js 로드 실패와 옵션 오류 확인 → 같은 오리진 제공<br>·liveSyncDurationCount: 3 |

### 내가 확인한 완료 조건

인증 성공 → muxer 동작 → 목록·객체 응답 → JS 초기화 → 실제 재생·되감기로 확인 범위를 넓혔습니다. 조각 길이 4.000000초와 브라우저 조작을 확인했고, 서버에서 수정한 플레이어·프록시·마운트 설정의 저장소 차이도 남겼습니다.

2026.08.18 · MediaMTX 1.19.3·당시 OBS/브라우저 조건의 진단 기록입니다. 설정별 효과를 독립 A/B로 모두 검증한 것은 아닙니다. 다음 사례의 fMP4 시간축 문제와는 다른 장애입니다.

CASE 14 · 원본 판단·실측 ↗

## 오류 0건인데 61.94초가 보이지 않았습니다

목록 길이·브라우저 버퍼·실제 탐색 위치를 서로 다른 성공 조건으로 봤습니다.

| 관측               | 초기 재생 결과                |
|------------------|-------------------------|
| 목록에 선언된 길이       | 141.97초                 |
| 브라우저가 인식한 구간     | 80.03초                  |
| 선언 길이와 인식 구간의 차이 | 61.94초 (141.97 - 80.03) |
| 60초 탐색           | 약 87초로 이동               |

### 문제를 좁힌 기준

각 녹화 파일의 디코딩 시작 시각 tfdt가 파일마다 다시 시작했습니다. 저는 HTTP 성공·오류 로그 0을 재생 성공으로 보지 않고, 파일 내부 시각과 플레이어가 실제로 읽은 구간을 대조하는 방향을 택했습니다.

| 시도한 대안                  | 남은 문제                     |
|-------------------------|---------------------------|
| 절대 시각 PROGRAM-DATE-TIME | 파일 내부의 상대 시간 중첩이 남음       |
| 초기화 정보 EXT-X-MAP        | 코덱 안내를 보완해도 시간축은 이어지지 않음  |
| DISCONTINUITY 표시        | 일부 구간은 살아나지만 경계 간극·멈춤이 남음 |

### 내가 바꾼 방향

원본 녹화는 보존하고 CDN에 전달할 재생용 파일에서 트랙과 시간축을 구성하기로 했습니다. 재생목록의 태그만 고치는 접근을 멈추고 파일 내부 변환의 필요성을 다음 장의 단일 변수 대조로 검증했습니다.

이 초기 141.97초 입력과 뒤의 39.87초·10분 실험은 서로 다른 데이터입니다. 한 입력의 동일 조건 전후 성과처럼 합치지 않았습니다.

CASE 22 · 실험 조건과 상세 근거 ↗

## 시간값만 바꾸자 39.87초 전체가 잡혔습니다

공통 init 분리와 시간축 변환을 따로 비교해, 어떤 변경이 재생을 바꿨는지 확인했습니다.

| 같은 녹화 파일 10개 | 변경한 것             | 인식 길이  |
|--------------|-------------------|--------|
| A · 원본 나열    | 변환 없음             | 8.10초  |
| B · 공통 init  | init만 분리, tfdt 유지 | 8.10초  |
| C · 연속 시간축   | B에서 tfdt만 변경      | 39.87초 |

### 설계의 전제부터 구분했습니다

MediaMTX의 파일별 상대 시간은 독립 녹화를 위한 의도된 동작이고, 공식 playback 서버가 방송 위치를 다시 배치합니다. PokeClip은 S3·CDN 직접 전달을 선택했으므로 그 서버를 거치지 않는 경로에서 시간축 변환을 말아야 했습니다.

### 내가 채택한 변환 규칙

파일의 mtxi.DTS에서 첫 파일의 값을 빼 방송 내 시작 간격을 구하고, 같은 간격을 각 트랙의 단위로 바꿔 기존 tfdt에 더했습니다. 영상과 오디오 각각의 길이를 누적하는 대신 한 기준을 적용해 파일 내부 A/V 차이를 보존했습니다.

| 통제·검증 항목     | 결과                        |
|--------------|---------------------------|
| B와 C의 바이트 차이 | tfdt 페이로드 안 193바이트 위치만 다름 |
| 목록 / 오디오 길이  | 39.873초 / 39.872초         |
| 브라우저         | 전체 버퍼 · 20초 탐색 성공         |

2026.08.30 · MediaMTX 1.20.1 · hls.js 1.5.15. 상류 버그 수정이나 속도 개선 실험이 아니라 직접 전달 경로의 변환 조건 검증입니다.

CASE 23 · 실험 조건과 상세 근거 [↗](#)

# Go 재포장: 같은 I/O 범위에서 p50 7.69배

파일마다 프로세스를 띄우던 작업을 상주 Go 프로세스의 재포장 호출로 바꿨습니다.

읽기·재포장·쓰기 p50

## 23.43 → 3.05 ms

동일 입력 · 각 200회 · 같은 I/O 범위 · 전체 서비스 지연과 구분

### 내가 바꾼 실행 단위

필요한 작업은 재인코딩이 아니라 압축 샘플을 보존한 트랙·시간 정보 변경이었습니다. 저는 매회 FFmpeg 실행을 상주 Go 호출로 바꿨습니다. 첫 비교는 Go I/O가 빠져 있어 읽기·쓰기를 포함해 다시 측정하도록 기준을 맞췄습니다. 프로세스 시작을 없애는 대신 init·시간축 규칙과 메모리 비용을 서비스 안에서 관리하는 선택입니다.

읽기·재포장·쓰기 p50 · 같은 입력 각 200회



| 측정                  | FFmpeg p50 | Go p50 | 비율    |
|---------------------|------------|--------|-------|
| 과거 150조각            | 18.70ms    | 1.72ms | 10.9배 |
| 동일 I/O 재측정 · 각 200회 | 23.43ms    | 3.05ms | 7.69배 |

### 재측정 조건을 맞추다

2026.10.04 · macOS arm64 · Go 1.26.5 · FFmpeg 8.1.2. 동일한 5,155,160B 입력(1080p·30fps·영상 1개 · AAC 오디오 6개)을 10회 예열 뒤 각 200회 처리했습니다. 두 방식 모두 읽기·재포장·로컬 쓰기를 포함합니다. Go는 상주, FFmpeg는 매번 시작하는 실제 구조 차이는 유지했습니다.

| 추가 확인          | 결과                             |
|----------------|--------------------------------|
| 동일 I/O 재측정 p95 | FFmpeg 24.50ms → Go 3.29ms     |
| 양쪽 산출물         | 영상 1개·오디오 1개 · 길이 4.066667초    |
| 과거 150조각 검증    | 초기화 파일 해시 1종 · 영상/오디오 경계 별도 대조 |

7.69배는 반올림 전  $p50\ 23.4330625 \div 3.0479995 (=7.688\cdots)$ 로 계산했습니다. 과거 Go 측정은 I/O 제외로 조건이 다릅니다. 재측정은 warm file cache이며 fsync·S3·네트워크·동시 방송을 제외했습니다. 전체 서비스 지연 개선율은 측정하지 않았습니다.

CASE 24 · 실험 조건과 상세 근거 ↗

## 동시성을 두 배로 늘려도 처리량은 비슷했습니다

속도 개선 뒤에는 작업 수 상한과 입력 제한의 근거를 따로 확인했습니다.

| 동시 작업 수 | 최대 RSS 중앙값 | 호출 p95 중앙값 |
|---------|------------|------------|
| 1       | 85MiB      | 5.83ms     |
| 2       | 107MiB     | 10.57ms    |
| 4       | 172MiB     | 11.15ms    |
| 8       | 303MiB     | 45.21ms    |

### 내 판단 · 빠른 호출만으로 동시성을 늘리지 않습니다

6.78MB 입력에서 동시 작업을 4→8로 늘려도 처리량은 약 293→294회/초였습니다. 반면 RSS는 172→303MiB, 호출 p95는 11.15→45.21ms로 늘었습니다. 저는 단일 호출의 속도만으로 동시성을 늘리지 않고, 작업 수 제한이 필요한 근거로 읽었습니다. 이 조건에서는 처리량 이득보다 메모리와 긴 응답의 비용이 컸습니다.

| 입력 상한 검사   | 산출·읽기 결과                           |
|------------|------------------------------------|
| 16MiB      | 성공 · 16MiB 읽음                      |
| 16MiB + 1B | ErrInputTooLarge · Read 0회 · 산출물 0 |

### 검증을 분리한 이유

동시성은 실제 녹화 입력 3종 × 4조건 × 독립 프로세스 5회, 프로세스당 800회를 측정했고 48,000회 모두 기대 출력 SHA256과 일치했습니다. 크기 상한은 유효한 free box로 경계를 만든 별도 입력에서 조건당 1회 검사했습니다.

2026.10.06 · Apple M5 · Go 1.26.5 · GOMAXPROCS=4. 동시성 실험은 디스크·S3 제외. p95는 실행별 p95의 중앙값이며 누적 할당은 최대 점유량과 다릅니다.

CASE 21 · 실험 조건과 상세 근거 ↗

CASE 24 · 실험 조건과 상세 근거 ↗

# 재접속의 0.2초 빈틈은 태그만의 문제가 아니었습니다

앞 조각의 트랙 끝과 플레이어의 재배치를 함께 비교했습니다.

## 원인과 내가 선택한 기준

끊김 직전 오디오는 영상보다 0.19~0.22초 먼저 끝났습니다. 늦게 끝난 영상에 다음 조각을 맞추면 오디오가 비었습니다. 저는 같은 회차·동일 init 조건에서 이른 트랙의 끝으로 보정하고, 플레이어가 불연속 표시를 어떻게 해석하는지 함께 비교했습니다.

| 변형 | 배치와 표시           | 재생 버퍼  |
|----|------------------|--------|
| A  | 늦은 트랙 끝 + 불연속 표시 | 3개로 분리 |
| B  | 이른 트랙 끝 + 불연속 표시 | 3개로 분리 |
| C  | 이른 트랙 끝 + 표시 생략  | 1개로 연결 |

| 후속 검증 C'     | 결과                                |
|--------------|-----------------------------------|
| 입력           | 10분 · 150조각 · 재접속 2회 · init 해시 1종 |
| hls.js 1.7.3 | 오류 0 · stall 0 · 탐색 12/12 성공      |
| EXTINF 유지    | 원래 영상 길이로 두어도 경계 통과               |
| AVFoundation | 두 재접속 경계 통과                       |

## 감수한 비용과 적용 한계

이른 트랙에 맞추면 영상 타임스탬프가 겹칠 수 있습니다. 표시 생략은 동일 회차·동일 init에 한정했고 코덱 설정이 달라진 입력에는 적용하지 않습니다. 보정값이 메모리에 있는 단계의 재기동 복구도 별도 과제로 남겼습니다.

2026.09.19 최초 / 09.23 후속 실험. 10분 실험과 앞 장의 단일 호출 속도는 서로 다른 검증입니다.

CASE 25 · 실험 조건과 상세 근거 ↗

# 재생용 파일의 생성과 저장 완료를 분리했습니다

원본·재생 조각·init을 각각 복구해야 한 축의 성공이 다른 실패를 가리지 않습니다.

## 내가 정한 책임

Producer는 입력과 방송 위치를 받아 init·재생 조각·init 해시를 반환하고, DB와 저장소 접근은 밖에 두기로 했습니다. 잘린 파일·트랙 누락·큰 입력에서는 부분 산출을 남기지 않고, 결정적인 변환 실패와 일시적인 업로드 실패를 다르게 다뤘습니다.

| 처리 축        | 완료·복구 기준                     |
|-------------|------------------------------|
| 편집용 원본      | 원본 객체와 DB의 업로드 상태            |
| 되감기 조각      | 재생 객체의 독립 상태·CAS·수거 커서       |
| 초기화 정보 init | init 해시와 업로드 확정 · 대기 중 조각 해제 |

| 실제 입력·저장 검증               | 결과                                |
|---------------------------|-----------------------------------|
| RTMP → 사이드카 → HTTPS MinIO | 재생 객체 24개 + init 1개 · 모두 첫 시도 업로드 |
| 두 번째 송출 seq 12-23         | 원본 12개 / 재생용 12개 모두 첫 시도          |
| 재생용 PUT                   | 중앙값 12ms · 최대 14ms                |
| 원본 완료 → 재생 완료 로그 간격       | 중앙값 22.65ms · 최대 29.72ms          |

PR #192 병합. 2026.09.23 MinIO 실측으로 AWS S3 성능은 아닙니다. 같은 109행 로그에 negative\_drift -1,980ms 오류 1건도 있어 업로드 성공을 시간축 문제 전체 해결로 확대하지 않습니다. 완료 로그 간격은 순수 재포장 CPU 시간이 아닙니다.

- 생산 계층 PR #192 ↗
- CASE 21 · 실험 조건과 상세 근거 ↗
- CASE 16 · 실험 조건과 상세 근거 ↗

# 첫 조각을 놓치는 조회 계약을 수정했습니다

파일의 시작점이 요청 안에 있는지 대신, 두 구간이 겹치는지를 물어야 했습니다.

두 실제 SQL과 독립 기대 결과의 대조

## 8,020개 기대 결과 일치

내 판단 · 시작점 포함 → 구간 겹침으로 계약 수정 / 소비자 구현은 팀원

### 내 기여 · 계약 수정과 소비자 구현을 구분

AI가 작성한 저장 구조 조사 4편에 교차 검토가 빠진 것을 확인하고 전부 재검토하도록 요구했습니다. 그 과정에서 클립 시작을 걸치는 조각 누락이 드러나, 저는 구간 겹침 조건으로 계약을 수정·확정하고 데이터 담당과 조율했습니다. 계약 수정은 08.14, 조회·회귀 테스트 구현은 팀원의 PR #120으로 08.24 병합됐습니다.

| 요청 (3.9초, 8초) | 필요한 조각   | 이전 BETWEEN | 겹침 조건 |
|---------------|----------|------------|-------|
| A · [0,4)     | 마지막 0.1초 | 누락         | 포함    |
| B · [4,8)     | 전체       | 포함         | 포함    |
| C · [8,12)    | 없음       | 잘못 포함      | 제외    |

확정한 두 조건 · 실제 duration 사용

```
segment_start < clip_end
AND segment_start + duration > clip_start
ORDER BY seq
```

### 고정 4초를 빼는 우회를 기각한 이유

녹화 길이는 키프레임 위치에 따라 달라질 수 있습니다. 검색 시작을 무조건 4초 앞당기면 실제 구간의 의미를 숨깁니다. 각 파일의 실제 길이로 겹침을 계산하고 정확히 끝점만 닿은 파일은 제외하도록 했습니다.

제 기여는 문제를 검토하고 계약을 수정·확정한 범위입니다. Spring Reader와 조립기의 팀원 구현을 단독 구현으로 기재하지 않습니다.

[소비자 구현·테스트 PR #120](#) ↗

[CASE 28 · 실험 조건과 상세 근거](#) ↗

## 두 시간축에서 8,020개 기대 결과와 일치했습니다

정확한 seq 집합을 독립 계산과 대조하고, 조회 뒤 가용성 판정은 따로 확인했습니다.

| 실제 SQL      | 검사 입력                 | 결과          |
|-------------|-----------------------|-------------|
| 상대 PTS      | 4,000 시작 오프셋 + 경계 10개 | 4,010/4,010 |
| playback 시간 | 4,000 시작 오프셋 + 경계 10개 | 4,010/4,010 |

### 현재 호출 경로도 검사했습니다

PostgreSQL 17에서 실제 SQL을 추출해 별도 교집합 계산과 대조했습니다. 렌더 경로의 COALESCE(playback\_pdt, start\_wall\_utc)는 playback 값이 있으면 우선하고 NULL이면 대체되는지 확인했습니다. 다른 스트림·pending/failed·정확한 접점도 포함했습니다.

| 요청과 파일 상태                       | 조립 결과                               |
|---------------------------------|-------------------------------------|
| [3.9,8) · A/B uploaded          | A·B 선택 · 요청 양 끝을 덮으므로 complete=true |
| [3.9,8) · A pending, B uploaded | B만 가용 · 요청 머리가 없어 complete=false    |
| [0,12) · A 완료, B 대기, C 완료       | A에서 멈춤 · 중간 미완료를 건너뛰지 않음            |

### 회귀 검증의 강도

기존 BETWEEN 대조식은 각 축 4,000개 오프셋 중 3,999개에서 앞부분을 놓쳤습니다. 별도 8개 경계 SQL 재현과 저장소 테스트는 순서·상태 보존·스트림 격리까지 확인합니다. 항상 빈 목록을 반환해도 통과하지 않도록 기대 seq를 명시했습니다.

2026.10.06 정확성 시험. 3,999/4,000은 합성 입력의 반례 비율이며 과거 운영 장애율이 아닙니다. Java HTTP→FFmpeg→최종 영상 전체 성공률이나 성능 측정도 아닙니다.

CASE 28 · 실험 조건과 상세 근거 ↗

## 식별자를 바꿀 때 소비자의 조회를 지켰다

회차 식별과 물리 경로를 분리하고, 소비자 배포 뒤 새 값을 발행하기로 합의했습니다.

### 문제와 첫 제안

송출 키는 재사용되므로 같은 계정의 월요일 방송과 화요일 방송을 구분할 수 없습니다. 이벤트 streamId를 방송 회차인 session\_id로 바꾸려 했지만, 기존 소비자는 물리 경로로 조각을 조회했습니다. 값만 바꾸면 기존 조회가 0행이 되는 영향이 있었습니다.

| 2026.09.03 협의    | 확인과 반영                                        |
|------------------|-----------------------------------------------|
| 15:10 · 변경 영향 공유 | 조각 조회가 깨지는 영향을 적고 소비자 측 대응 방식 확인 요청           |
| 16:06 · 팀원 답변    | 물리 경로 + 시간 범위를 유지. 회차 앞부분·기존 NULL 행과 배포 순서 고려 |
| 16:19 · 계약 확정    | ingestStreamId 추가 수용. 소비자 배포 완료 뒤 새 값 발행      |

| 필드                 | 한 가지 의미              |
|--------------------|----------------------|
| streamId           | 방송 회차(session_id)    |
| ingestStreamId     | 실제 녹화·조각 조회·재생 경로    |
| streamerId         | 사용자 ID               |
| eventId / sequence | 이벤트 중복 식별 / 회차 안의 순서 |

### 협업에서 한 일

제 제안을 전 서비스에 일괄 적용하는 대신 팀원의 조회 기준을 유지할 필드를 추가했습니다. 두 필드의 용도와 전환 순서를 계약에 남겨 미디어 변경과 소비자 배포가 어긋나는 문제를 줄이려 했습니다. 위키 PR #113으로 계약을 반영했습니다.

이 사례의 근거는 계약 원문과 확인 요청·답변 기록입니다. 이벤트 계약의 합의와 전체 생산자·소비자의 운영 배포 완료는 별개입니다. Spring 소비자의 주 구현은 팀원 담당입니다.

### 내가 조율한 호환성

회차 ID로 바꾸는 생산자 쪽의 편의보다 기존 소비자의 조회 의미를 먼저 맞췄습니다. eventId는 중복 식별, sequence는 회차 내 순서, ingestStreamId는 파일 조회 좌표로 나뉘었습니다. 각 서비스가 종료를 제각각 판정하지 않도록 미디어가 확정된 생명주기를 소비하게 했습니다.

CASE 33 · 실험 조건과 상세 근거 ↗

# 캐시 키와 원본 인증 전달을 분리했습니다

같은 미디어를 공유하되, 접근 권한과 원본 세션은 각 경계에서 확인해야 했습니다.

공유 캐시 구성에서 관측한 원본 미디어 요청

## 1,254 → 28건

50 VU · 25초 · 실험용 비교군과 순차 비교 · 상세 측정 21쪽

### 비교 실험의 문제

같은 URI도 세션 쿠키·쿼리마다 캐시가 갈라지는 비교군에서 시작했습니다. 원본 전달에서도 세션을 지우자 init·segment가 401을 반환했습니다. 저는 재사용을 나누는 캐시 키와 원본 인증 전달을 분리하고, 수정한 구성에서 원본 요청·전송량·응답 지연을 각각 비교했습니다. 비교군은 실험용 구성입니다.

| 정책                  | 선택과 이유                                    |
|---------------------|-------------------------------------------|
| CachePolicy         | 동일 URI가 동일 바이트인 미디어 조각에서 쿠키·쿼리를 캐시 키에서 제외 |
| OriginRequestPolicy | 캐시 미스 때 MediaMTX 세션 쿠키·쿼리를 원본으로 전달        |
| 목록 / 미디어            | 변하는 목록은 TTL 0, 내용이 고정된 조각은 TTL 120초       |
| 뷰어 권한               | CloudFront 서명 쿠키를 엣지에서 검사. Hit에도 적용       |

### 실패한 첫 시도에서 바꾼 점

캐시 키뿐 아니라 원본 전달에서도 세션 정보를 제거하자 init·segment 요청이 401을 반환했습니다. 재사용을 구분하는 키와 원본 인증에 필요한 전달값을 분리해야 했습니다. 초기 실패를 pilot 기록으로 남기고 수정 구성으로 정식 부하를 비교했습니다.

2026.10.05 임시 AWS 실험. 쿼리가 사용자별 다른 콘텐츠를 만드는 서비스에는 이 정책을 그대로 적용할 수 없습니다. 라이브 실험은 뷰어 HTTPS / CloudFront→EC2 HTTP입니다.

CASE 12 · 실험 조건과 상세 근거 ↗

# 원본 미디어 요청 1,254건이 28건으로 줄었습니다

공유 캐시의 바이트 전송 회피율은 97.95%였고, 응답 지연은 비슷했습니다.

원본 미디어 요청 · 시청자별 캐시 → 공유 캐시

## 1,254 → 28건

50 VU · 25초 · 각 조건 1회 · 응답 지연의 개선율과 구분

내 판단 · 변경 · 같은 미디어의 세션별 캐시 분산을 막도록 캐시 키를 바꾸고, 원본 인증 전달은 유지했습니다.

50 VU · 25초 · 원본 미디어 전송량



| 동일 EC2 · 순차 비교 | 시청자별 캐시       | 공유 캐시      |
|----------------|---------------|------------|
| 클라이언트 미디어 수신   | 253.00MB      | 263.243MB  |
| 원본 미디어 요청      | <b>1,254건</b> | <b>28건</b> |
| 요청 기준 캐시 적중률   | 0.00%         | 97.81%     |
| 미디어 HTTP p95   | 1,442.14ms    | 1,389.47ms |

## 분모를 고정해 읽었습니다

원본 요청은 1,254→28건으로 줄었지만 p95는 1,442.14→1,389.47ms로 비슷했습니다. 공유 캐시의 바이트 전송 회피율은  $1 - 5.387 / 263.243 = 97.95\%$ 입니다. 전후 수신 총량이 달라 원본 바이트의 전후 감소율과 구분합니다. 목록 1,366요청·1.921MB는 계속 원본으로 가며 미디어와 합친 원본 응답은 7.308MB입니다.

t3.small · MediaMTX 1.20.1 · CPU 1 / 384MiB · 720p30 합성 방송. 3구성 × 1/10/25/50 VU, 각 조건 1회·25초. 완료 HTTP 16,214건 실패·순번 누락 0, 마지막 반복 1개 중단. 반복 신뢰구간은 없으며 CPU·재생 지연 98% 개선으로 해석하지 않습니다.

CASE 12 · 실험 조건과 상세 근거 ↗

# 이미 캐시에 있는 영상도 권한을 확인했습니다

서명된 Miss·Hit와 서명 없는 요청을 같은 URI에서 대조했습니다.

Authorized: first fetch

HTTP/2 200

content-type:  
video/mp4

content-length:  
373502

x-cache:  
Miss from cloudfront

x-amz-cf-pop:  
ICN53-P1

Authorized: same segment

HTTP/2 200

content-type:  
video/mp4

content-length:  
373502

x-cache:  
Hit from cloudfront

x-amz-cf-pop:  
ICN53-P1

Unsigned: same segment

HTTP/2 403

content-type:  
text/xml

content-length:  
146

x-cache:  
Error from cloudfront

x-amz-cf-pop:  
ICN53-P1

Actual curl response headers · CloudFront ICN53-P1 · 2026-10-05 · identical media URI

2026.10.05 실제 응답 헤더 발췌 · 최초 / 반복 / 서명 없음

| 같은 미디어 URI      | HTTP / 캐시  | 본문 검증        |
|-----------------|------------|--------------|
| 유효 서명 · 최초      | 200 / Miss | 373,502B     |
| 유효 서명 · 반복      | 200 / Hit  | 길이·SHA256 동일 |
| 서명 없음 · 캐시 예열 후 | 403        | 접근 차단        |

## 내가 확인 대상으로 삼은 실패

원본에서만 인증하면 캐시 적중 요청은 그 검사를 거치지 않을 수 있습니다. 저는 캐시 공유의 이득과 권한 검증을 한 변경의 조건으로 보고, 옛지 서명 검사와 미스 시 원본 세션 전달을 함께 검증하는 구성을 선택했습니다.

## 검증이 증명하는 범위

같은 URI가 같은 미디어 바이트를 가리키는 통제된 스트림에서 캐시 재사용과 무서명 거부를 확인했습니다. 인증 정책을 바꾸지 않은 로컬 부하 실험과도, 다음 장의 시청자 인증 없는 S3 객체 실험과도 구분합니다.

CASE 12 · 실험 조건과 상세 근거 ↗

## 두 번째 요청을 곧바로 캐시 적중으로 세지 않았습니다

실제 X-Cache 상태와 본문을 확인한 뒤 Miss·Hit와 부분 읽기를 비교했습니다.

### 2026.10.06 S3 원본

| 관측 결과                         |
|-------------------------------|
| 즉시 두 번째 전체 GET                |
| 20/20 Miss                    |
| 같은 IP · 1초 뒤 두 번째 GET         |
| 20/20 Miss                    |
| 후속 세 번째 전체 GET                |
| 20/20 Hit · 같은 POP · 해시 일치    |
| 첫 256KiB Range                |
| 10/10 · HTTP 206 · 범위 · 본문 일치 |

| 연결 준비 뒤 첫 바이트 | 최초 Miss | 후속 3차 Hit |
|---------------|---------|-----------|
| 256KiB 객체     | 54.0ms  | 14.2ms    |
| 4MiB 객체       | 52.9ms  | 15.0ms    |

### 내가 지킨 계측 기준

재요청 횟수를 캐시 상태의 대리값으로 쓰지 않고 실제 헤더로 분류했습니다. DNS·TCP·TLS 영향을 나누기 위해  $\text{time\_starttransfer} - \text{time\_pretransfer}$ 를 비교했습니다. 두 번째가 Miss인 내부 원인은 이 자료만으로 확정하지 않았습니다.

같은 IP·POP · 새 curl HTTP/1.1 · 최초/3차 20쌍 · 총 110요청. 뷰어→CloudFront와 OAC(always)→비공개 S3 모두 HTTPS. 시험 객체에 시청자 인증은 적용하지 않았습니다. 영상 재생·운영 CPU·S3 전송 절감률의 근거로 확대하지 않습니다.

CASE 12 · 실험 조건과 상세 근거 ↗

## 새 저장 계층보다 현재 작업 단위를 먼저 검토했습니다

4초 객체와 시간 인덱스가 이미 있다면, 큰 파일의 부분 읽기 문제와 조건이 다릅니다.

### 내가 보류한 도입

PokeClip의 클립 생성은 선택 구간과 겹치는 몇 개 원본을 받는 작업입니다. 저는 별도 KV·FUSE·라이브 저장 계층을 추가하기보다 현재 시간 인덱스와 제한된 병렬 다운로드를 유지하는 방향을 택했습니다. 별도 계층은 인덱스·복구 책임도 늘리기 때문입니다.

### 초기 조사 결론도 다시 검토했습니다

S3 쓰기가 비핵심이라는 초기 설명은 원본 가용성과 복구 영향을 빠뜨렸습니다. DVR의 서로 다른 객체 탐색과 렌더·AI의 직접 GET도 다시 반영했습니다. 재검토는 기술 도입 여부를 넘어 첫 조각이 빠지는 조회 계약의 정정으로 이어졌습니다.

| dev EC2 S3 PUT · 예열 제외 | p50     | p95     |
|------------------------|---------|---------|
| 1,206,272B · 29회       | 40.52ms | 49.02ms |
| 849,920B · 29회         | 36.07ms | 39.82ms |

### PUT 결과로 정한 다음 질문

각 30회 총 60회가 HTTP 200이었고 첫 요청은 각각 71.62/61.43ms였습니다. 이 조건의 PUT은 수십 ms였으므로 파일 마감·혹·재포장·DB 확정과 다운로드를 따로 봐야 했습니다. 다음 장은 실제 S3 GET 배치 완료 실험입니다.

2026.08.30-31 t3.small의 단기·순차 PUT 원시 기록 재집계. 개별 p95를 합쳐 종단 p95로 쓰지 않으며, 이 결과만으로 저장소 병목이 없다고 결론내리지 않았습니다.

CASE 19 · 실험 조건과 상세 근거 ↗

## 8개 객체의 다운로드 완료 시간을 비교했습니다

병렬도를 늘린 효과는 파일 크기와 실행 위치에 따라 달랐습니다.

| 서울 Lambda · 배치 중앙값 | 순차      | 2병렬     | 4병렬     | 8병렬     |
|--------------------|---------|---------|---------|---------|
| 1MiB × 8개          | 246.2ms | 154.4ms | 136.1ms | 130.5ms |
| 4MiB × 8개          | 411ms   | 372ms   | 391ms   | 359ms   |

1MiB × 8개 · 서울 Lambda · 수신과 해시 검증 완료



### 내 판단 · 동시성을 무제한으로 늘릴 근거는 아닙니다

1MiB 배치는 약 47%, 4MiB 배치는 약 13% 짧아졌습니다. 4MiB에서 2병렬이 4병렬보다 짧은 결과도 남겼습니다. 현재 객체 단위를 유지하되, 실제 워커의 파일 크기·메모리·동시 작업 수에 맞춰 상한을 정해야 합니다.

2026.10.06 · 서울 리전 · Python 3.13 ARM64 Lambda 512MB · 각 조건 7회 무작위 순서 · 연결 풀 16 · 최대 시도 1 · 예열 포함 456객체 크기/SHA 일치. 임시 다운로드 실험으로 렌더 서비스 중단 성과가 아닙니다. 같은 코드의 Mac 1MiB 배치는 884.9→257.0ms였지만 다른 네트워크·자원 조건이라 합치지 않았습니다.

CASE 19 · 실험 조건과 상세 근거 ↗

# 오디오를 많이 나누는 것보다 편집 목적을 먼저 정했습니다

본방 믹스와 소스별 트랙을 보존하면서, 방송자의 기존 설정을 보호하는 입력 계약입니다.

## 내가 줄인 구현 범위

초기 약 10개 소스를 직접 추적하는 우회 대신, 저는 본방 믹스와 편집용 소리를 OBS 네이티브 6트랙으로 전달하도록 요구를 줄였습니다. 소스 생성·종료·무음까지 별도 엔진이 관리하지 않도록 하고 서버 트랙명은 audio1~audio6으로 고정했습니다.

| 제품 요구        | 선택한 경계                                                       |
|--------------|--------------------------------------------------------------|
| 본방 인코딩 부담    | 같은 압축 영상을 SRT 출력에도 재사용. 본방 시작 전 GOP 조건을 맞추고 비호환 실행 중 인코더는 거절 |
| 기존 오디오 설정 보호 | 본방·VOD 트랙을 보호하고 장면 모음별 백업·복원. 자동 배정은 기본 꺼짐                   |
| 첫 오디오의 의미    | 고정 믹서 0이 아니라 실제 본방 믹서를 사용. 본방 트랙 2도 같은 규칙                    |
| 순간 표시 재시도    | 같은 eventId로 재시도. UI 구현과 서버 저장 선행 조건을 구분                      |

## 기여와 실물

제 기여는 전체 송출 구조·서버 입력·통합 계약의 선택과 조율입니다. PR #179·#195의 OBS 플러그인 구현·실측은 팀원 jaehwan-space의 작업입니다. 다음 장은 이 계약이 파일까지 전달됐는지 보여주는 팀 검증입니다.

로컬 macOS 검증을 실제 치지직 송출이나 Windows 실기 검증까지 완료한 것으로 확대하지 않습니다.

팀 OBS 구현 PR #179 ↗

팀 멀티오디오 구현 PR #195 ↗

CASE 05 · 실험 조건과 상세 근거 ↗

CASE 06 · 실험 조건과 상세 근거 ↗

CASE 07 · 실험 조건과 상세 근거 ↗

CASE 08 · 실험 조건과 상세 근거 ↗

# 트랙 수와 실제 소리의 보존을 따로 확인했습니다

입력 개수만 맞는 것과, 사용자가 선택할 소리가 분리돼 남는 것은 다른 조건입니다.

| 초기 6트랙 경로 검증       | 결과                             |
|--------------------|--------------------------------|
| SRT → fMP4 녹화      | 오디오 6개 보존                      |
| LL-HLS 목록          | EXT-X-MEDIA 6개                 |
| 브라우저 트랙 선택         | 요청 200 · 오류 0                  |
| 여섯 번째 오디오 추출       | -map 0:a:5 → 출력 오디오 1개         |
| OBS 녹화·인덱스         | 61파일 = DB 61행                  |
| PR #195 팀 테스트 톤 실험 | 관측                             |
| 입력 톤               | 440 / 1,000Hz, 방송 중 2,500Hz 추가 |
| ffprobe / HLS      | 영상 1 + AAC 6 / 대체 오디오 6        |
| 해당 스템 목표 대역        | -24.1dB                        |
| 빈 트랙               | -91.0dB                        |
| 본방 트랙 2 / 컬렉션 A·B  | 본방 440Hz 보존 / 각각 원래 체크 복원      |

## 내가 구분한 검증 범위

초기 무음 입력은 존재·순서·선택 추출만 증명합니다. 후속 팀 실측은 테스트 톤의 분리까지 확인합니다. 필터의 다른 대역 -47~-57dB를 원음 혼입률로 단정하지 않았고, 실제 음악 제거의 청감 품질과도 구분했습니다.

초기 스파이크 2026.08 · 후속 OBS 32.2.2 macOS→SRT→로컬 MediaMTX. 팀원 구현·측정, 김태윤 전체 구조·미디어 통합 기여.

CASE 06 · 실험 조건과 상세 근거 ↗

CASE 08 · 실험 조건과 상세 근거 ↗

# 마이크를 고르는 문제에 상시 VAD가 필요할까

모델을 추가하기 전에, 판단이 필요한 시점과 사람이 이미 하는 일을 살폈습니다.

## 내가 기각한 접근

VAD는 음성이 있는지를 찾지만 게임 NPC의 대사와 방송자 마이크의 의미를 구분하지 못합니다. 저는 이미 편집자가 클립을 고르는 단계에 상시 전 트랙 분류가 필요한지 되물었습니다. 기능 추가보다 잘못 정의한 자동화 문제를 줄이는 쪽을 택했습니다.

| 사용 흐름         | 채택한 설계                                                 |
|---------------|--------------------------------------------------------|
| 사람이 편집하는 경로   | 선택 구간만 디코드 → 파형·RMS로 무음 후보 표시 → 솔로 미리듣기 → 선택 트랙만 자막 입력 |
| 사람이 없는 원클릭 경로 | 템플릿의 지정 트랙 사용 → 렌더 전 무음 검사                             |
| 폐기한 의존        | 서버의 소스 라벨 자동 분류. 이후 고정 트랙명 계약에 맞춰 필수 조건에서 제거           |

## 줄인 책임과 남긴 대가

계속 돌아가는 분류 시스템과 그 오판을 관리하는 책임을 줄였습니다. 대신 편집 경로에는 사용자의 선택이 남고, 원클릭 경로에는 템플릿 설정이 필요합니다. RMS도 소리 에너지이므로 마이크나 사람의 말을 자동 식별했다고 쓰지 않습니다.

ADR-008 채택 및 ADR-017 후속 계약에 근거한 설계 판단입니다. CPU 절감률·분류 정확도·에디터 전체 구현 완료를 입증하는 사례는 아닙니다. 수치가 없어도 요구 재정의와 대안 기각의 근거가 분명해 본문으로 복구했습니다.

CASE 31 · 원본 판단·실측 ↗

## 일회용 연결과 장기 권한을 나눴습니다

OBS에 OAuth 전체를 다시 넣지 않고, 웹 로그인과 짧은 페어링 코드를 연결했습니다.

### 내가 정한 연결 흐름

방송자가 웹 로그인 뒤 8자 코드를 OBS에 한 번 입력해 송출 설정을 받는 흐름을 선택했습니다. 플러그인 OAuth 로그인·콜백·토큰 갱신의 별도 구현 대신 기존 웹 인증을 활용했습니다. 온보딩 입력 1회와 매 방송 반복 설정 0회는 서로 다른 요구로 구분했습니다.

| 계약         | 선택·검증                                         |
|------------|-----------------------------------------------|
| 코드 수명      | Crockford Base32 8자 · 10분 만료 · 일회용 교환         |
| 시도 제한      | 발급은 계정, 교환은 IP 기준. 발급 계정의 자격증명에만 연결           |
| 실제 Auth 연동 | PR #179에서 교환 성공 · 동일 코드 재사용 409 · 설정 파일 권한 확인 |
| 비밀 전달      | SRT URL 로그에 passphrase를 섞지 않는 별도 연결 필드        |

### 같이 설계한 권한 경계

편집 가능 여부와 외부 채널 업로드 권한을 분리하고, 권한 없는 업로드 요청은 방송자의 승인 대기함으로 보내도록 설계했습니다. 편집자가 방송자의 OAuth 토큰을 소유하지 않고 서버가 실행하도록 했습니다.

페어링의 실제 연동은 팀 구현 검증입니다. CASE 09는 passphrase 원문을 Secrets Manager로 옮기고 DB에는 참조를 남기는 방식으로 변경할 예정이며, 현재 전환을 준비 중입니다. CASE 11의 전체 승인·토큰 모델도 설계 범위입니다.

CASE 09 · 실험 조건과 상세 근거 ↗

CASE 10 · 실험 조건과 상세 근거 ↗

CASE 11 · 실험 조건과 상세 근거 ↗

## DB를 정본으로 두는 발행 설계

오래된 작성자가 최신 재생목록을 덮지 않도록 설계했습니다. 발행 계층 연결은 진행 중입니다.

### 막으려는 실패

작업 A가 10번 조각까지 목록을 만들고 멈춘 사이 B가 11번까지 공개할 수 있습니다. 뒤늦게 A가 덮어쓰면 시청자는 이미 보던 11번 조각을 잃습니다. DB와 S3에 나눠 쓰는 두 작업은 하나의 트랜잭션이 아닙니다.

| 기준              | 설계한 역할                            |
|-----------------|-----------------------------------|
| DB 장부           | 무엇을 발행할지와 복구 시작점을 결정              |
| 작성 권한 번호(fence) | 이전 작성자의 발행 권한 구분                  |
| 세대 번호 + 조건부 갱신  | 현재 상태와 일치할 때만 다음 세대 예약            |
| S3 조건부 PUT      | 예상한 객체 버전에만 쓰기                    |
| 종료·확정 GAP       | 종료한 방송을 다시 열거나 실패 조각을 조용히 삭제하지 않음 |

### 내가 감수한 복잡도

시청 요청마다 DB를 읽는 대신 파일 변경 시 정적 목록을 발행하도록 선택했습니다. DB를 복구 기준으로 두는 대가로 세대 예약·확정·작성 권한 관리가 늘어납니다. 09.29 설계 산식의 상한 약 2,820 DB 왕복/시간/스트림은 운영 실측 이 아닙니다. 종료된 ENDLIST는 되돌리지 않고 새 회차 URL을 사용합니다.

| 확인된 상태                   | 남은 검증                                  |
|--------------------------|----------------------------------------|
| 생산자 #192 · 목록 렌더 #196 병합 | S3 발행 루프와 종료 전이의 통합                    |
| 발행 계층 구현 진행 기록           | 오래된 writer·불명확한 DB commit·종료 경쟁의 통합 검증 |

이 장은 운영에서 경합을 막아낸 성과가 아니라 설계 판단 사례입니다. 2026.10.12까지의 프로젝트 기록을 기준으로 하며, 운영 충돌 발생 횟수나 복구 시간은 제시하지 않습니다.

CASE 20 · 실험 조건과 상세 근거 ↗

CASE 26 · 실험 조건과 상세 근거 ↗

# 미완료 조각 뒤의 완료 파일을 먼저 공개하지 않습니다

발행 설계와 별개로, 목록의 문법과 공개 경계를 순수 부품으로 구현·검증했습니다.

## 내가 선택한 분리

DB·S3·HTTP·파일 접근 없이 상태 목록에서 재생 가능 범위와 HLS 문장을 계산하도록 나눴습니다. 10번 완료·11번 대기·12번 완료에서 최신 seq만 보고 12번까지 공개하면 11번의 404가 생깁니다. 연속으로 확정된 앞부분만 공개해야 합니다.

| 상태·경계                | 규칙                                      |
|----------------------|-----------------------------------------|
| uploaded / 확정 GAP    | 공개 가능한 상태로 확정                           |
| pending / 아직 미확정인 구멍 | 그 뒤의 완료 파일까지 넘어가지 않음                    |
| 최근 1시간 창             | 머리에서 거꾸로 창을 채울 꼬리를 계산                   |
| 불연속 누적값              | 렌더 태그와 축출 시 누계가 같은 행 판정 함수를 사용          |
| 회차 계승                | 300초 이내·init 호환 등 조건을 확인. 종료된 기존 목록은 불변 |

## 구현이 있는 근거

PR #196은 경계 계산, 렌더·검사·봉인, 캐시와 부팅 장부 읽기를 분리해 병합했습니다. 900조각 기대 목록을 독립 Python 생성기와 바이트 대조하고 hls.js 1.5.15·1.6.0·1.7.3 파서의 구문 오류 0을 확인했습니다.

골든·경계·결합 주입·race·PG 읽기 검증을 포함합니다. 이 PR만으로 발행 루프가 켜지는 것은 아니며 호출·S3 조건부 PUT·종료 전이 연결은 후속 계층입니다.

렌더·경계 PR #196 ↗

CASE 27 · 실험 조건과 상세 근거 ↗

CASE 26 · 실험 조건과 상세 근거 ↗

# 긴 원장과 최근 1시간 창을 나뉘 봤습니다

같은 구현의 첫 계산·증분 갱신·전체 렌더를 서로 다른 작업으로 측정했습니다.

| 최근 900조각 · 10만 행 원장 |           |         | 관측        |
|---------------------|-----------|---------|-----------|
| 전체 목록 렌더            |           |         | 0.280ms   |
| 전체 렌더 할당            |           |         | 약 507KB/회 |
| 원장 행 수              | 최초 계산     | tail 갱신 | 보유 heap   |
| 900                 | 3.933μs   | 4.162μs | 0.106MiB  |
| 5,400               | 21.324μs  | 3.646μs | 0.680MiB  |
| 21,600              | 71.739μs  | 2.855μs | 2.393MiB  |
| 100,000             | 320.410μs | 2.952μs | 11.58MiB  |

## 내 판단 · 계산량과 메모리는 다른 문제입니다

보여줄 창은 일정해도 과거 원장을 모두 들고 있으면 보유 메모리는 증가합니다. 초기 전체 탐색과 정상 갱신을 분리해 보되, 경계 계산만 0할당이라는 결과를 전체 렌더가 0할당인 것처럼 설명하지 않았습니다.

## 반례도 함께 검사했습니다

900·5,400·21,600·100,000행에서 최근 1시간 창을 고정하고 조건별 1초 × 10회 측정했습니다. 미해결 구멍, 공개 GAP, 목록 복사의 독립성도 확인했습니다. 출력 규칙과 비용을 발행 연결 전에 독립 검증하는 목적입니다.

2026.10.06 합성 원장 구성요소 벤치마크. 초기 계산 320.410μs와 tail 갱신 2.952μs는 수정 전후가 아니므로 개선 배수로 쓰지 않습니다. DB·네트워크 처리량과 실제 서비스 수용량도 아닙니다.

CASE 27 · 실험 조건과 상세 근거 ↗

## 정리 코드보다 테스트 DB의 소유권을 먼저 확인했습니다

테스트가 남긴 가짜 실패 행을 실제 복구 작업자가 처리하는 오염을 막았습니다.

### 원인과 내 선택

통합 테스트가 연결만 닫고 데이터를 남겨 공용 개발 DB에 가짜 실패 행 1,121개가 쌓였습니다. 종료 때 삭제하는 처방은 강제 종료를 건디지 못합니다. 저는 전용 PG\_TEST\_DB를 강제하고, 지울 수 있는 DB의 소유권부터 확인하도록 선택했습니다.

| 입력·상태                | 처리                               |
|----------------------|----------------------------------|
| 공용 DB와 같은 이름         | 시작 거부                            |
| 기준 DSN의 DB 이름이 비어 있음 | 시작 거부                            |
| 이미 있는 DB에 소유 표식 없음   | 사용·정리 거부                         |
| 새 테스트 DB             | 생성 시에만 pokeclip_testdb_marker 생성 |
| 동시 실행                | 실행별 고유 DB 이름 필요                  |

### 기각한 판정 · 테이블 모양은 소유권이 아닙니다

처음의 테이블 개수 판정은 공용 DB도 같은 모양일 수 있었습니다. 미끼 DB의 데이터가 삭제되는 반례가 확인돼, 업무 스키마가 아닌 명시적 소유 표식으로 바꿨습니다. store·upload의 부트스트랩도 한 곳으로 모았습니다.

### 결과와 비용

PR #61이 병합됐고 동일 이름·빈 이름·표식 없는 DB의 실패 경계와 정상 생성 경로를 확인했습니다. CREATEDB 권한과 기존 테스트 DB의 일회성 이행은 필요하며, 동일 DB 이름을 공유하는 동시 실행까지 안전해진 것은 아닙니다.

테스트 DB 격리 PR #61 ↗

CASE 17 · 실험 조건과 상세 근거 ↗

# root 없이 녹화하려면 새 볼륨의 소유권도 맞아야 했습니다

실행 사용자만 바꾸는 대신 이미지와 공유 파일의 권한을 함께 설계했습니다.

## 내가 선택한 경계

scratch에는 시작 시 chown을 실행할 셸이 없습니다. 저는 prep 단계에서 디렉터리를 만들고 COPY --chown으로 최종 이미지의 소유권을 정하는 방식을 선택했습니다. 빈 named volume으로 처음 복사되는 디렉터리까지 포함한 변경입니다.

| 대상         | 정한 권한·검증                            |
|------------|-------------------------------------|
| 미디어 / 사이드카 | UID 10002가 녹화, 10001이 읽기            |
| 디렉터리 / 파일  | 0755 / 0644로 공유 읽기                  |
| 실행 혹       | root 소유, 일반 사용자의 덮어쓰기 금지            |
| 반례         | root 소유여도 0777이면 보호되지 않음 → 계약 검사 보강 |
| 기존 볼륨      | 이미 가진 소유권은 별도 일회성 이행 필요             |

## 다른 대안을 쓰지 않은 이유

root init은 매번 넓은 권한으로 볼륨에 접근하고, gosu는 root 시작이 남습니다. Compose pre\_start는 팀 도구 버전 조건을 바꿉니다. 이미지 단계에서 소유권을 정해 런타임 root 작업을 없애는 쪽을 택했습니다.

PR #62 병합. USER·COPY·쓰기 모드·NoNewPrivs·녹화·혹을 검사했습니다. macOS bind mount의 소유권 재매핑을 Linux 검증으로 대체하지 않으며 기존 DB 볼륨까지 지우는 일괄 초기화는 이행 방법이 아닙니다.

컨테이너 권한 PR #62 ↗

CASE 18 · 실험 조건과 상세 근거 ↗

## 테스트 성공·커버리지·실행 범위를 따로 다뤘습니다

품질 검사의 판정과 팀의 병합 정책은 같은 장치가 아닙니다.

| 실제 Actions 대조 · PR #77 | index 커버리지     | 테스트 / 게이트 |
|------------------------|----------------|-----------|
| 정상                     | 72/88문 · 81.8% | 성공 / 통과   |
| 테스트 파일 하나 제외           | 46/88문 · 52.3% | 성공 / 실패   |
| 복원                     | 81.8%          | 성공 / 통과   |

### 내가 정한 검사 기준

테스트 개수가 줄어도 남은 테스트는 성공할 수 있습니다. 핵심 패키지별 80% 기준을 두고 실제 테스트 제외·복원 대조로 게이트를 확인했습니다. 반올림된 80.0%가 아니라  $\text{covered} \times 100$ 과  $\text{total} \times 80$ 을 비교해 79.99%를 차단합니다. 합성 프로파일 10종도 기대 판정과 일치했습니다.

### 팀 경계에 맞춘 CI 변경과 대가 · PR #163

미디어와 무관한 PR에 media-ci가 나타나지 않게 paths로 범위를 좁혔습니다. 사라진 체크를 필수로 기다리지 않도록 보호 설정을 먼저 해제한 뒤 워크플로를 바꾸는 순서를 정했습니다. 검사 내용은 유지했지만 실패가 병합을 강제 차단하는 보호는 사라지는 선택이었습니다.

08.18 Actions 원본은 테스트 제외 2줄 외 변화와 다른 패키지 비율을 대조했습니다. 10.06의 10종은 원본 스크립트에 합성 coverprofile을 넣은 별도 검증입니다. 검사 속도 개선이나 제품 커버리지 신규 측정으로 쓰지 않습니다.

커버리지 PR #77 ↗

실행 범위 PR #163 ↗

CASE 34 · 실험 조건과 상세 근거 ↗

CASE 35 · 실험 조건과 상세 근거 ↗

# 릴리스 교체를 9개 의존 계약으로 검증했습니다

상류 수정의 반영 여부부터 파일 포맷까지, 버전 문자열 밖의 계약을 확인했습니다.

## 내 판단 · 사용하던 전제를 업그레이드의 통과 조건으로

MediaMTX를 직접 전부 만들지 않는 선택에는 버전 변경을 검증할 책임이 남습니다. v1.20.1에서 v1.21.1-pokeclip.1로 옮길 때, 저는 설정·혹·녹화 파일에 의존하던 전제 9곳을 다시 확인하고 제품·개발 환경이 같은 버전을 바라보도록 묶는 범위를 확정했습니다.

| 검증한 경계             | 변경과 확인                                     |
|--------------------|--------------------------------------------|
| 설정·혹 의존 전제 9곳      | 버전 변경 뒤 기존 가정의 유효성을 재확인                    |
| Dockerfile과 테스트 상수 | 태그가 다르거나 누락되면 계약 검사에서 실패                   |
| Go 미디어 라이브러리       | mediacommon v2.9.5 · go-mp4 v1.7.3로 정합성 유지 |
| 녹화 fixture         | 영상 1·오디오 6 입력을 새 버전으로 재녹화, 기대값 6곳 갱신       |
| 재포장 회귀             | 기존 golden 결과가 바뀌지 않았는지 별도로 확인              |

## 왜 버전 이름만 바꾸지 않았나

새 서버를 예전 녹화 fixture로 검사하면 실제 배포할 조합을 검증하지 못합니다. 저는 새 입력에 필요한 기대값 6곳의 갱신과 재포장 golden 유지 여부를 나눴습니다. 기대값을 모두 새 결과로 덮어 회귀를 숨기지 않기 위해서입니다. 초기 계약 검사에서는 태그 불일치·Dockerfile 누락·태그 없는 FROM·중복 FROM도 실패하도록 정했습니다.

## 상류 기여를 제품 의존성 관리로 연결

PR #217은 2026.10.06 병합됐습니다. 상류 정식 v1.21.1과 PokeClip 포크 태그를 구분하고, 이미지 태그·digest와 라이브러리·fixture를 함께 맞췄습니다. 이후 포크 .2를 고정한 PR #231도 10.12 병합됐지만 실제 서버 배포·연동 완료를 뜻하지는 않습니다.

본인 기여는 변경 범위·통과 조건의 판단과 검토·확정입니다. 구현·회귀 실행은 AI 협업입니다. 초기 혹 검증의 194개 PASS·변형 오류 7/7 탐지는 별도 시점의 근거이며, #217의 새 실행 결과와 합산하지 않았습니다.

버전·의존성·fixture 갱신 PR #217 ↗

후속 포크 버전 고정 PR #231 ↗

상류 정식 v1.21.1 릴리스 ↗

# 설정 오류가 방송 요청까지 잠복하지 않게

PR #6154 병합 · v1.21.0 반영 — 실제 실행 제약을 공통 설정 검증으로 옮겼습니다.

## 사용한 도구의 한계까지 확인하다

PokeClip의 수신·녹화는 MediaMTX에 맡겼습니다. 다만 LL-HLS의 조각 수를 6으로 설정하면 서버는 기동하고 실제 송출 시 먹서가 실패했습니다. 설정 검증은 통과하지만 하위 라이브러리는 최소 7개를 요구했습니다. 자동 재생 모드에서는 10초마다 같은 실패가 반복됐습니다.

### 수정 전

서버 기동 성공 → 송출 시 먹서 실패  
40초 관측에서 실패 5회  
0 입력은 하위 라이브러리 기본값으로 대체

### 수정 후

Conf.Validate에서 잘못된 설정을 즉시 거부  
허용·거부 경계 6개 테스트  
0도 설정 오류로 거부 · 호환성 변화 명시

## 왜 공통 검증 경계인가

파일·환경변수·API 설정 입력이 같은 검증을 거치게 했습니다. HLS가 켜진 경우 lowLatency는 7 이상, fMP4·MPEG-TS는 3 이상을 요구하고, HLS가 꺼진 경우에는 관련 없는 값으로 기동을 막지 않습니다.

## 검증과 외부 리뷰

최소값 미만 6·2·2는 기동 거부, 경계값 7·3·3은 정상 기동을 확인했습니다. lowLatency 7의 30초 송출에서 먹서 실패 0과 목록 응답을 확인했습니다. PR의 외부 리뷰도 세 유형과 경계 조건을 검증했고, 메인테이너가 2026.08.30 병합했습니다. v1.21.0 릴리스에 포함됐습니다.

PR #6154 작성자: xodbs1021. 본인 역할은 수정 방향·검증 범위 판단과 상류 제안, 구현·실측·리뷰는 AI 협업입니다. 처리속도 개선이 아닌 설정 오류 발견 시점을 앞당긴 변경입니다.

## 내가 선택한 수정 위치와 호환성

먹서를 계속 재시작하는 처방보다 설정을 받아들이는 공통 경계에서 거부하도록 했습니다. HLS를 끈 경우에는 관련 없는 값으로 시작을 막지 않습니다. 종전 0을 기본값으로 바꾸던 입력이 거부되는 호환성 변화도 PR에 명시했습니다.

CASE 36 · 실험 조건과 상세 근거 ↗

v1.21.0 공식 릴리스 ↗

## 종료 신호와 작업 완료는 다르다

인증서 갱신 도중 Close를 호출해, 남은 goroutine의 완료를 기다리도록 수정했습니다.

종료 순서 회귀 검증 · 같은 테스트 -race -count=20

## 20/20 실패 → 20/20 통과

내 변경 · 종료 신호만 보내던 Close가 watcher 작업 완료까지 기다리도록 수정

### 문제

인증서 watcher에 종료 신호를 보냈더라도 이미 시작된 갱신 작업은 계속될 수 있습니다. Close가 먼저 반환하면 그 뒤에도 로거를 사용하거나 인증서 상태를 바꾸는 경로가 남습니다. 채널을 닫는 것만으로 자원 회수의 순서를 보장할 수 없었습니다.

| 순서    | 수정한 동작                            |
|-------|-----------------------------------|
| 시작    | watcher 실행 전에 WaitGroup 등록        |
| 종료 요청 | done 채널을 닫아 종료를 요청                |
| 작업 완료 | Wait로 진행 중인 watcher가 끝나기를 대기      |
| 자원 정리 | watcher 자원 · 인증서 상태 정리 후 Close 반환 |

### 우연을 기다리지 않는 재현

실제 인증서 갱신을 유발하고 테스트의 Parent.Log에서 진행을 멈췄습니다. 이때 Close를 호출해 watcher를 풀기 전에는 반환하지 않아야 한다고 검사했습니다. 같은 테스트를 수정 전과 수정 후에 적용했습니다.

2026.10.04 · -race -count=20

결과

|               |                           |
|---------------|---------------------------|
| base 04825598 | 20/20 실패 · 갱신 중 Close가 반환 |
| head 296e3cea | 20/20 통과 · 갱신 종료까지 대기     |

### 결과와 대가

반환 시점을 실제 작업 완료에 맞춘 대신, 느린 로거가 있으면 Close도 늦어질 수 있습니다. 이 수치는 처리량 향상이 아니라 종료 순서에 대한 회귀 검증입니다. 다른 모든 경합이 없다는 뜻은 아닙니다.

20회 비교는 base 04825598 / 당시 head 296e3cea의 같은 TestCloseDuringReload입니다. 현재 PR head 전체를 이번에 재실행한 결과가 아니며 PR #6261은 2026.10.15에도 열림입니다.

2026.10.04 원문 발췌 · 같은 회귀 테스트의 전후 로그

이전: Close() returned while a reload was in progress

--- FAIL: TestCloseDuringReload (0.02s)

이후: --- PASS: TestCloseDuringReload (0.52s)

PR #6261 · 재현 테스트와 열린 제안 ↗

# 인증 기본값의 표현과 공유 참조를 함께 고쳤습니다

PR #6205 병합 · v1.21.1 반영 — 정규화와 공유 참조를 함께 고쳤습니다.

| 서로 다른 원인                       | 필요했던 변경                  |
|--------------------------------|--------------------------|
| nil과 []의 의미는 같지만 DeepEqual은 다름 | 기본 IPs 표현을 정규화           |
| 얕은 복사로 중첩 권한 목록이 공유됨           | 기존 deepClone으로 내부 값까지 복제 |

## 내가 한 판단

기본값을 명시한 정상 설정과 이전 인증 필드의 조합이 거부되는 문제를 하나의 비교 오류로만 다루지 않았습니다. 환경변수로 로드된 값을 바꾸면 전역 기본값까지 변하는 별칭 공유도 있어, 정규화와 독립된 소유권을 함께 해결하는 수정 범위를 택했습니다.

| 회귀 테스트                     | 확인한 경계                 |
|----------------------------|------------------------|
| 기본값 정규화                    | 로딩된 값과 비교 기준의 의미 일치    |
| 명시적 기본값 + deprecated auth  | 정상 조합 허용               |
| 기본값 비공유                    | 로딩된 사용자를 바꿔도 전역 기본값 유지 |
| 환경변수 사용자 + deprecated auth | 사용자 지정과 이전 필드의 충돌은 거부  |

PR #6205 · 2026.09.20 병합 · v1.21.1 정식 릴리스 반영. 인증 우회가 입증됐다는 주장이 아니라 설정 비교·공유 상태 수정입니다. 독립된 테스트로 로딩 순서에 따른 오염을 구분했습니다.

- 정규화·깊은 복사 PR #6205 ↗
- CASE 39 · 실험 조건과 상세 근거 ↗
- v1.21.1 공식 릴리스 ↗

# 기존 기여를 이어 실제 소스의 온라인 경계로 옮겼습니다

PR #6182 병합 · v1.21.0 반영 — 대체 화면과 실제 방송의 녹화 경계를 분리했습니다.

## 기여의 출발점

원 작성자 AbdElrahman31의 PR #5767과 커밋을 보존해 #6182를 이어 진행했습니다. 제 기여는 최초 아이디어를 새로 만든 것이 아니라, 누락된 static source·설정 재적용 경계를 보완하고 회귀 테스트를 갖추는 범위를 확정한 것입니다.

| 같은 3개 시나리오         | 원 기여 상태  | 보완 뒤 |
|--------------------|----------|------|
| publisher 온라인      | 정상       | 통과   |
| static source 온라인  | 녹화 시작 누락 | 통과   |
| offline 상태의 reload | 대체 화면 녹화 | 통과   |

## 선택한 수정

publisher 추가 한 곳에서만 녹화를 시작하지 않고 setOnline/setOffline 상태 전환에 연결했습니다. reload는 source 포인터 존재 여부가 아니라 isOnline()으로 판단하도록 바꿨습니다. 시청용 대체 화면은 유지하면서 원본 녹화 여부를 분리하는 경계입니다.

## 호환성도 선택의 일부입니다

alwaysAvailableRecorded 기본값 true로 기존 동작을 유지하고, false를 선택한 경우에만 실제 방송을 녹화하도록 했습니다. 설정 예시와 생성된 OpenAPI도 함께 맞췄습니다.

PR #6182 · 2026.09.05 병합 · v1.21.0 정식 릴리스 반영. 원 기여자 커밋을 보존한 승계·보완 작업입니다.

승계·보완 PR #6182 ↗

CASE 38 · 실험 조건과 상세 근거 ↗

v1.21.0 공식 릴리스 ↗

# 실패한 명령과 시작조차 못 한 명령을 구분했습니다

#6155는 v1.21.0 반영, #6259는 열림 — 종료 코드와 실행 실패의 관측 경계를 나눴습니다.

| 서로 다른 관측 공백           | 수정·검증                                                                         |
|-----------------------|-------------------------------------------------------------------------------|
| #6155 · 실패 종료가 0으로 보임 | ExitError의 ExitCode()를 계산한 뒤 return 누락. 즉시 반환하도록 수정하고 exit 5를 재시작 ON/OFF에서 검증 |
| #6259 · 실행 실패 원인이 없음  | 13개 호출 중 OnExit이 없는 7개에 기존 6개와 같은 실패 콜백 연결                                    |

## 내가 구분한 문제

실행 파일이 없으면 자식 프로그램이 스스로 오류를 남길 수 없습니다. 종료 코드 수정 하나로 모든 혹은 실패가 보이는 것도 아니어서 호출부의 콜백까지 대조했습니다. 관측의 책임을 명령 내부와 부모 서버로 나눠 최소 수정 범위를 정했습니다.

| 7종 프로브 · 통제된 실행 사건 | 수정 전        | 수정 후    |
|--------------------|-------------|---------|
| launched           | 19          | 19      |
| 실패 원인을 포함한 exited  | 0           | 19      |
| 수정 후 일반·race 실행    | DATA RACE 0 | panic 0 |

#6155는 08.30 병합되어 v1.21.0에 반영됐고, #6259는 2026.10.15 열림입니다. 19건은 실험 이벤트 수입입니다. 재시도·종료 정책은 바꾸지 않았으며 정상 종료는 계속 조용합니다.

종료 코드 PR #6155 ↗

실행 실패 보고 PR #6259 ↗

CASE 37 · 실험 조건과 상세 근거 ↗

CASE 41 · 실험 조건과 상세 근거 ↗

v1.21.0 공식 릴리스 ↗

# 로그를 더 모으는 선택은 종료 비용도 바꿉니다

관측 기능 자체가 실행에 미치는 영향을 확인한 로컬 구현·검증 사례입니다.

## 내가 요구한 검증 · “왜 동시 혹은 8개인가”

당시 제안에는 일부 스파이크를 생략하고 한계만 적는 선택지가 있었습니다. 저는 전체 시험을 진행하기로 했고, 동시 혹은 수 8의 근거를 되물었습니다. 그 결과 동시 종료 수와 혹은 종류의 배선 검사를 나눠, 1·4·8·16 동시성과 13종 혹은의 연결 여부를 별도로 다루게 했습니다.

## 선택한 동작과 비용

stdout·stderr를 서버 로거로 보내 파일·syslog에서 실패 이유를 찾게 하되 기본은 꺼진 선택 옵션으로 뒀습니다. 긴 줄·UTF-8 경계·마지막 미완결 줄을 처리했습니다. 느린 로그 목적지는 출력 복사와 cmd.Wait(), 명령 풀 종료에 대기를 전파할 수 있습니다.

| 대안·조건           | 내가 확인해야 했던 차이                         |
|-----------------|---------------------------------------|
| 일정 시간 뒤 출력을 포기  | 종료를 빨리 끝낼 수 있지만 로그 유실 가능              |
| 혹·출력 완료 뒤 로거 교체 | 출력 보존에 유리하지만 느린 목적지·종료하지 않는 혹은 대기를 늘림 |
| WaitDelay 1초    | 느린 로거 호출 전체의 상한이 아님                   |

## 수치가 판단을 바꾸는 지점

기능 ON의 성공만 보여주지 않고 동시 출력의 유실과 가장 늦은 혹은의 대기를 함께 봤습니다. 후속 가드 v2에서 동작이 달라지자 “로거 교체 중 폐기”라는 설명도 “완료까지 대기”로 정정했습니다.

개인 로컬 구현·실험으로 상류 제출·병합 성과가 아닙니다. 선행 #6206은 미병합 종결됐고 유지보수자 #6326이 대체했습니다. 아래 과거 실측이 현재 상류와의 호환성이나 운영 동작을 보장하지 않습니다.

CASE 43 · 원본 판단·실측 ↗

## 유실률과 종료 꼬리를 같은 실행 단위로 봤습니다

대량 출력 스파이크와 후속 리로드 보존 시험은 질문·입력이 다른 실험입니다.

| 동시 혹 수 | 파일 로그 유실률 중앙값 | 최후 혹 exit→로그 지연 |
|--------|---------------|-----------------|
| 1      | 0%            | 315ms           |
| 4      | 1.864%        | 1,296ms         |
| 8      | 29.252%       | 1,371ms         |
| 16     | 32.220%       | 2,832ms         |

### 2026.09.08 · 같은 양의 혹 출력을 늘린 실험

혹마다 50,000줄, 동시성별 10회로 총 40회 실행했습니다. 혹별 290행을 290번의 독립 실행으로 부풀리지 않았습니  
다. 16개 조건에서는 한 혹이 최대 23,877줄을 잃었습니다. 위 지연은 회차에서 가장 늦은 혹을 고른 뒤 그 중앙값입  
니다.

| 2026.09.23 · 가드 v2       | OFF 10회      | ON 10회   |
|--------------------------|--------------|----------|
| PID별 1,000줄 · 합계 10,000줄 | 상속 stdout 전량 | 파일 로그 전량 |
| ON의 stdout 누출 / 유실       | —            | 0 / 0줄   |
| 리스너 부재 중앙값               | 0.610ms      | 0.598ms  |

### 성능 개선으로 합치지 않은 이유

후속 시험은 같은 바이너리에서 옵션만 바꿨지만, 로컬 파일과 SIGINT에 바로 끝나는 혹을 사용했습니다. 표의 반올  
림값 차이 0.012ms는 산포 안에 있어 속도 개선으로 볼 수 없습니다. 이전의 느린 목적지·대량 출력 유실을 해결했다  
고 연결하지 않았습니다.

후속 코드 204195ad · 비-race 빌드. 느린 목적지와 리로드가 겹치거나 혹이 종료하지 않는 경우는 별도 검증이 필요합니다. 현재 상류 호환성 확  
인 전의 역사적 실험 기록입니다.

CASE 43 · 원본 판단·실측 ↗

# 병합된 수정과 채택되지 않은 접근을 구분합니다

2026.10.15 확인 · 제출 8건 중 4건 병합·릴리스 반영 / 열림 3 / 미병합 종결 1

| PR    | 내용                  | 상태           |
|-------|---------------------|--------------|
| #6154 | HLS 설정 검증           | 병합 · v1.21.0 |
| #6155 | 외부 명령 종료 코드         | 병합 · v1.21.0 |
| #6182 | 대체 화면 녹화 제어 · 승계·보완 | 병합 · v1.21.0 |
| #6205 | 기본 인증 설정 정규화·깊은 복사  | 병합 · v1.21.1 |
| #6259 | 7종 혹 실행 실패 보고       | 열림           |
| #6261 | 인증서 watcher 종료 대기   | 열림           |
| #6306 | 사용자별 SRT passphrase | 열림           |
| #6206 | 로거 교체와 혹 종료 순서      | 미병합 종결 10.11 |

## 채택되지 않은 접근 · #6206

로거 교체 전에 혹 종료를 기다리는 접근에서, 종료가 3초 늦는 혹은 리스너 부재 중앙값을 0.248ms→3,014.39ms로 늘렸습니다. 이후 유지보수자는 혹이 로거 재생성을 일으킬 때 교착할 수 있음을 지적하고, 로거를 닫지 않고 다시 실행하는 #6326으로 직접 해결했습니다. 제 PR 병합으로 집계하지 않습니다.

#6306은 기존 43사례와 별도로 보강한 후속 제안입니다. 미병합 종결 #6206과 로컬 실험 CASE 43은 릴리스 성과에 포함하지 않습니다.

[기여 목록 · xodbs1021 ↗](#)

[유지보수자의 대체 수정 #6326 ↗](#)

[v1.21.0 공식 릴리스 ↗](#)

[v1.21.1 공식 릴리스 ↗](#)

[CASE 40 · 실험 조건과 상세 근거 ↗](#)

# 사용자별 암호에서 보안과 대기 비용을 함께 봤습니다

PR #6306 · 2026.10.15 열림 — 릴리스된 4건과 구분한 별도 제안입니다.

## 내가 정한 범위 · 내부 사용자와 HTTP 인증

방송 경로 하나의 암호를 여러 사용자가 공유하는 대신, 내부 사용자 설정과 HTTP 인증 응답에서 SRT 암호를 받을 수 있는 범위를 택했습니다. JWT에 암호를 넣는 방식은 토큰이 암호화되지 않은 SRT streamid로 전달될 수 있어 제외했습니다. 지원 범위와 제외 이유를 함께 공개하도록 결정했습니다.

| 설계·검증 경계   | 결과                                  |
|------------|-------------------------------------|
| 우선순위와 호환성  | 사용자 암호가 경로 암호보다 우선 · 빈 값은 경로 값으로 복귀 |
| 입력과 노출     | 10-79자 검사 · API 응답의 비밀값 가림          |
| 실제 SRT 송수신 | publish/read 12개 하위 테스트             |
| HTTP 응답    | 본문 유효성·누락·지연·잘림 등 27개 하위 테스트        |
| 응답 자원 한도   | 본문 128KiB 제한 · readTimeout으로 대기 제한  |

## 통과 결과와 함께 남긴 비용

성공 응답의 본문을 읽는 동안 경로 관리자가 묶일 수 있습니다. 20회 중앙값에서 100ms 본문 지연 조건은 0.27→102.53ms, 1초 지연은 0.29ms→1.00초, 끝나지 않는 본문은 0.36ms→2초로 늘었습니다. 정상 조건의 작은 차이를 속도 개선으로 내세우지 않고 지연 한계와 함께 제안했습니다.

## 기여와 현재 상태

저는 지원·제외 범위와 호환성·검증 조건을 판단하고 채택을 확정했습니다. 구현·프로브 실행·리뷰는 AI 협업입니다. PokeClip 포크 반영과 상류 정식 채택은 별개이며, 현재 상류 PR이 열려 있으므로 병합·공식 릴리스 성과로 집계하지 않습니다.

위 숫자는 해당 프로브·fixture의 관측입니다. 임의의 모든 인증 서버 지연이나 실제 운영 부하를 보장하는 결과가 아닙니다. 지연된 2xx 본문이 경로 관리자를 점유하는 비용은 남아 있습니다.

사용자별 SRT passphrase PR #6306 · 설계·테스트·한계 ↗

# Shakedown · 분석 결과를 배포와 복구까지 연결

SoftBank Hackathon · Emerald 팀 · 2026.10.08-10 작업 기록

## 서비스와 내가 맡은 범위

저장소를 분석해 이미지를 만들고, 로컬·AWS에 배포한 뒤 HTTP 시운전 결과를 확인하는 서비스입니다. 저는 PostgreSQL 샘플과 로컬 Target API에서 시작해 엔진 통합, 규칙 우선 이미지 생성, 선택한 AWS 구성의 실제 적용, DB 연결과 복구 검증을 맡았습니다.

| 본인 기여                  | 기존 팀 기반과 구분                          |
|------------------------|--------------------------------------|
| 실행 엔진 연결·오류 상태 보완      | HTTP 시운전·규칙/Claude 원인 보고서는 김태현 구현    |
| ECR·선택 계획 적용·AWS 결함 수정 | 초기 ECS/ALB/RDS 어댑터와 공유 세션 기반은 김재환 구현 |
| 보안 게이트 연동·문법 검사 보완     | 초기 게이트와 다국어 기본 보완은 김도경 기여            |
| 다중 DB·TLS·복원·직접 배포     | GCP·Azure·팀 HTTPS 자동화를 제 실측에 합산하지 않음 |

## 내 판단이 드러나는 선택

AI가 임의 인프라를 만들게 하기보다 허용된 구성 안에서 판단하게 했습니다. 수동 실행 성공과 제품 API 지원, 데이터 보존과 서비스 자동 기동을 따로 확인했고, 재부팅 뒤 삭제한 서비스가 살아나지 않는 조건도 요구했습니다.

개인 PR #2·#5·#10·#11·#12·#29·#31은 모두 병합 확인(2026.10.15 조회). 조사·구현·실험 실행에는 AI를 활용했고 요구·범위 선택과 결과에 따른 판단을 본인 기여로 구분합니다. 전체 UI→검사→LLM→AWS의 단일 E2E 성공을 주장하지 않습니다.

저장소와 팀 구성 ↗  
작업·실측 회고 30건 · 고정 커밋 ↗

# AI에 맡기는 범위부터 제한했습니다

예측 가능한 생성은 규칙으로 처리하고, 모델의 판단은 실행 전에 다시 검증했습니다.

## 내가 정한 이미지 생성 순서

기존 Dockerfile → 지원 스택 규칙 → 규칙 실패 때 Claude 1회 → 결과 검증 → 별도 소스 복사본에서 빌드로 정했습니다. 모든 저장소를 처음부터 모델에 맡기는 대신, 규칙이 왜 실패했는지와 허용된 출력 형식을 전달합니다. 근거가 부족하면 null을 허용하며 원본을 덮어쓰지 않습니다.

## 규모 판단에도 실행 가능한 선택지를 뒀습니다

소스만으로 피크 트래픽과 허용 중단 시간을 알 수 없어 사용자 운영 입력을 받습니다. 규칙의 최소 등급을 정한 뒤 모델은 소·중·대 템플릿과 근거 ID 안에서 선택하게 했습니다. 임의 IaC 생성보다 검토할 자원·권한·비용의 범위를 좁히는 선택입니다.

| 실제 API 시험 · 2026.10.09 | 확인한 결과                                     |
|------------------------|--------------------------------------------|
| 규칙 미지원 Flask           | fallback 1회 응답 3.679초 · Docker 빌드·HTTP 200 |
| 소·중·대 판단 · 각 1회        | 3.876 / 4.551 / 4.434초                     |
| large 요구의 선택 검증        | small 선택 400 · large 선택 200                |

위 시간은 단일 호출 관측값이며 품질·속도 개선율이 아닙니다. 모델 입력은 제한된 분석 정보와 실패 진단으로, 원본 코드·README·환경변수 전체를 보내지 않습니다. Dockerfile 검사는 실행 샌드박스가 아니며 카탈로그 수치는 처리량 보장이 아닙니다.

이미지 생성 PR #11 ↗  
실제 AI 호출·실행 보고서 ↗

## 선택한 설계가 실제 자원에 반영되도록 했습니다

“계획 저장 성공”과 “배포 완료” 사이의 계약·상태·시간 예산을 연결했습니다.

### 내가 정한 실행 계약

초기 기능은 추천·저장과 고정 설정의 AWS 실험이 분리돼 있었습니다. architecture\_plan\_id로 최신 선택·프로젝트·분석 근거를 다시 확인하고 서버 카탈로그를 적용하도록 연결했습니다. 클라이언트가 보낸 임의 CPU·태스크 수는 그대로 실행하지 않습니다. 선택 저장만으로 자원이 생성되지도 않습니다.

| 실제 배포에서 만난 충돌            | 수정과 이유                                                          |
|--------------------------|-----------------------------------------------------------------|
| 준비 전 403 차단 → ECS 생성 실패  | ALB-target group의 기본 forward 연결을 유지하고 별도 우선순위 1 규칙으로 403/공개를 제어 |
| Spring 기동 약 41초 / 유예 30초 | 건강 검사를 끄지 않고 유예를 120초로 조정                                       |
| 중지 차단은 됐지만 120초 뒤 502    | 내부 어댑터 600초 / 외부 엔진 630초로 상태 수렴 대기 조정                           |

### 단순 성공 응답을 넘어 확인한 것

배포 중에는 차단을 유지하고 실제 healthy 태스크와 AZ를 확인한 뒤 공개했습니다. 보완 뒤 large 중지는 189.70초에 204·INACTIVE·running/pending 0으로 끝났습니다. 앞선 medium의 502 기록은 그대로 남겼습니다.

시간 예산은 이 샘플의 실제 기동·중지 결과를 바탕으로 정했습니다. 앱 중지는 RDS·ALB 등 기반 스택 삭제와 다르고 DB 변경 자동 롤백도 아닙니다. 초기 AWS 어댑터는 팀 기반이며 제 기여는 연결·경계 수정과 검증입니다.

계획 적용 PR #12 ↗

선택 계획 실제 배포 보고서 ↗

# AWS 설정 적용과 장애 뒤 복구를 분리 검증했습니다

각 실험의 복구 확인 시간이며, 처리량·무중단·정밀 RTO 보장으로 쓰지 않습니다.

실제 엔진→AWS · 10.09

medium

large

태스크당 CPU / 메모리

1 vCPU / 2GiB

2 vCPU / 4GiB

정상 태스크 / AZ

2개 / 2개

3개 / 3개

RDS 설정

Single→Multi-AZ

Single→Multi-AZ

CPU 60% 확장 범위

2-4

3-12

세션·쓰기·조회

교차 인스턴스 통과

세 인스턴스 통과

별도 장애·확장 시험

관측 결과

large 태스크 강제 종료

58.69초 뒤 정상 3개·3 AZ 복귀 · 세션/데이터 유지

CPU 자동 확장

desired/running 2→4 · 96.53초에 healthy ≥3, 이후 4개

RDS 강제 AZ 전환

71.57초 뒤 기존 데이터 조회·같은 세션 추가 쓰기 복구

## 내가 성공에서 분리한 것

정책이 있다는 이유만으로 자동 축소가 검증됐다고 쓰지 않았습니다. 최초 16분 동안 축소는 미확인이었고, 추가 관측 때는 정리가 시작돼 유효한 재시험이 아니었습니다. RDS 전환에는 실패 폴링과 healthy 0 구간도 있었습니다.

계획 적용은 FastAPI TestClient→이미지/ECR→AWS에서 use\_ai=false로 검증한 경로입니다. 각 장애 시간에는 폴링·API 지연이 포함됩니다. large DB도 db.t3.micro였고 최대 12태스크·대표 부하·최소 IAM·전체 브라우저 E2E는 미검증입니다.

2026.10.09 result.json · checks.task\_recovery 발췌

```
"seconds": 58.69, "healthy_tasks": 3,
"zones": ["ap-northeast-2a", "ap-northeast-2b", "ap-northeast-2c"],
"session_and_data": true
```

선택 구성·태스크 복구 실측 ↗

확장·RDS 전환 실측 ↗

# 잘못된 Java 구문이 검사를 통과했습니다

기대값을 낮추지 않고, 검사 도구가 놓치는 경계를 다른 검사로 보완했습니다.

## 실제 연동에서 드러난 반례

class Broken { void broken( { 입력은 SCAN\_FAILED여야 했지만 Semgrep 1.180.0 경로에서 ALLOW가 나왔습니다. 기존 회귀 테스트 두 개가 같은 실패를 보여줬고, 최적화를 끄는 시도도 해결하지 못했습니다.

## 선택한 보완과 기각한 대안

기존 회귀 테스트 2개의 기대값은 낮추지 않았습니다. 저는 Tree-sitter의 기존 Java grammar를 Semgrep 앞에 두어 구문 오류를 먼저 거부하도록 했습니다. javac·Gradle은 클래스패스·다운로드·빌드 스크립트 실행이 섞여 이 단계의 범위에서 제외했습니다. parser 부재·crash·timeout도 SCAN\_FAILED로 처리합니다.

| 검증 질문               | 결과와 범위                                    |
|---------------------|-------------------------------------------|
| 실패하던 원래 테스트 2개      | SCAN_FAILED 기대값 그대로 통과                    |
| 불완전 구문·현대 Java 구문   | 괄호/대입 오류, record·sealed·switch 등 검사       |
| 검사 과정의 실행 여부        | 클래스패스 없는 구문·static initializer 미실행 확인     |
| 후속 고정 grammar/규칙 검사 | 게이트 427 통과 · Windows 전용 1 skip, 엔진 207 통과 |

PR #11의 연동·보완 기여입니다. 초기 보안 게이트는 팀원 작업이며 문법 정상은 컴파일·타입·보안 안전 보장이 아닙니다. C/C++·C#·Ruby·PHP 등 구문 전용 범위는 REVIEW로 남겼고 전체 언어 지원으로 쓰지 않았습니다.

[고정 커밋의 실제 scanner 보고서 ↗](#)

[Java 사전 검사 수정 ↗](#)

# TLS 연결 성공과 잘못된 호스트 거부는 다릅니다

암호화·서버 신원·장애 전환을 하나의 성공으로 묶지 않았습니다.

## MySQL에서 보완한 두 검사

mysql2가 인증서 체인과 호스트명을 별도로 검사한다는 점을 확인하고 rejectUnauthorized와 verifyIdentity를 명시했습니다. 정상 RDS CA 연결과 일부러 틀린 호스트의 거부를 짝으로 검사했습니다. 드라이버가 오류를 HANDSHAKE\_SSL\_ERROR로 감싸므로 임의 TLS 실패가 아니라 호스트 불일치 메시지까지 확인했습니다.

| 실제 MySQL 8.4.7 · mysql2 3.24.5 | 결과                                |
|--------------------------------|-----------------------------------|
| 정상 CA와 호스트                     | HTTP 200 · TLS 연결 · Unicode 쓰기/읽기 |
| 정상 네트워크 주소 · 잘못된 TLS 호스트       | 호스트 불일치로 거부                       |
| 초기 프로브의 잘못된 오류코드 기대            | 연결 성공으로 오인하지 않고 검사 도구를 정정         |

## MongoDB는 암호화와 가용성을 별도 구성

사용자 선택에 따라 EC2 3대·3 AZ replica set에 TLS·멤버 인증·앱 전용 계정을 적용했습니다. 앱은 CA와 앱 자격만 받고 서버 private key·관리자 자격은 받지 않습니다. 대신 인증서·백업·멤버 교체 운영 책임이 늘어납니다.

| Mongo 단일 primary 장애 | 관측                                   |
|---------------------|--------------------------------------|
| EC2 중지              | 새 primary 쓰기 2.13초 · 거친 표본 2개에서 실패 0 |
| 프로세스 SIGKILL        | 쓰기 12.24초 · HTTP 실패 2회 · 기존 선택 문서 보존 |

2026.10.10 실측. 장애 수치는 표본·API/SSM 지연을 포함합니다. Mongo 호스트명 음성 시험은 실패 원인을 격리하지 못해 확증으로 쓰지 않습니다. MySQL 결과를 모든 드라이버로, 단일 장애를 무중단·EC2 자동 교체로 확대하지 않습니다.

MySQL TLS·복원 PR #29 ↗

Mongo TLS·장애 실측 ↗

## 백업의 존재 대신 복원한 데이터로 확인했습니다

원본을 건드리지 않는 새 대상에서 이전 데이터와 새 쓰기를 함께 검사했습니다.

### 내가 정한 복구 확인 기준

스냅샷이 있다는 사실과 실제로 복구할 수 있다는 사실을 구분했습니다. MySQL은 별도 RDS로 복원하고 앱의 URL Secret·불변 버전을 새 endpoint에 맞춰 재배포했습니다. 이전 UUID·Unicode 행 조회와 새 쓰기를 확인하고, 응답의 DB 호스트가 바뀌었는지도 확인했습니다.

| 대상·복원 경로                           | 확인한 것                                      |
|------------------------------------|--------------------------------------------|
| MySQL 수동 스냅샷 → 새 private RDS       | 기존 행 조회 · 새 쓰기 · 검증된 TLS · 새 endpoint      |
| Mongo 보조 노드 종료 → EBS 스냅샷           | 새 암호화 볼륨에 복원 · XFS nouuid                  |
| Mongo 격리 standalone · network none | 원래 문서와 새 쓰기/읽기. 원본 replica set에 합류하지<br>않음 |

### 자동화 권한의 한계는 성공으로 바꾸지 않았습니다

Mongo 예약 스냅샷은 조직 SCP로 거부됐습니다. 다른 스케줄러로 우회하거나 수동 복원 성공을 자동 백업 성공으로 쓰지 않았습니다. 원본을 보존하는 복구 실험과 운영 백업 자동화는 완료 조건이 다릅니다.

2026.10.10 · PR #29 병합. MySQL 자동 스냅샷 available은 관측했지만 복원에 쓴 것은 수동 스냅샷입니다. PITR·장기 보관·전체 replica set 재구성은 미검증입니다. AWS 45·로컬 12 테스트와 실험 자원 제거를 별도로 확인했습니다.

DB 복원·실제 연결 보고서 ↗

# 재부팅 뒤 복구와 삭제 뒤 미복구를 함께 확인했습니다

데이터 영속성·앱 기동·제어 API·삭제 상태는 서로 다른 책임입니다.

## 초기 실험이 드러낸 공백

EC2 재부팅 뒤 데이터 2건은 남았지만 컨테이너는 0개였고 수동 compose up이 필요했습니다. 저는 이를 자동 복구 성공으로 보지 않고 앱·제어 API의 자동 기동을 요구했습니다. 컨테이너가 없다는 사실만으로 삭제와 종단을 구별할 수 없어, 삭제 의도도 디스크에 남기고 재부팅 뒤 미부활을 함께 검증하기로 했습니다.

| 내가 정한 동작    | 구현·검증 기준                                                     |
|-------------|--------------------------------------------------------------|
| 직접 배포       | 운영자가 정한 origin만 허용 · 실제 URL 200 확인 · 수동 Compose 편집 없이 API 배포 |
| 앱과 제어 계층 복구 | Docker restart: unless-stopped + Docker 뒤 systemd Target API |
| 주소 예약       | 한 미삭제 배포만 허용 · 경쟁 요청 409 · 실패 배포도 DELETE 전까지 예약              |
| 삭제          | 컨테이너/네트워크 제거 · tombstone 영속 · DB 볼륨 보존                       |

| 실제 EC2 · Node/PostgreSQL | 관측 결과                                           |
|--------------------------|-------------------------------------------------|
| 첫 재부팅 · boot ID 변경       | 새 POST·수동 시작 없이 API/앱/DB 기동 · 데이터 2건·외부 HTTP 확인 |
| DELETE 후 두 번째 재부팅        | 컨테이너 미부활 · GET 404 유지 · DB 볼륨 보존                |

PR #31 병합(10.10, 10.14 재확인). 당시 엔진 421+로컬 16=437 통과이며 다른 시점 테스트와 합산하지 않습니다. 실제 사내망·HTTPS 발급·HA·호스트 유실·MySQL/Mongo 호스트 재부팅은 이 시험 밖입니다. Docker 기동 정책 외에 앱의 DB 연결 재시도도 필요합니다.

직접 배포·재부팅 PR #31 ↗

제품 API·두 번 재부팅 실측 ↗

## 참고 기록으로 남긴 두 사례와 후속 조건

PokeClip 35건·MediaMTX 8건의 43사례를 보존했습니다. 14·31·43은 본문, 13·32는 참고 기록으로 남겼습니다.

| 참고 유지               | 이유와 다시 본문에 넣을 조건                                                                  |
|---------------------|-----------------------------------------------------------------------------------|
| CASE 13 · 계정별 경로 조정 | 옛 reconciler는 연결별 인증으로 대체. 부분 목록의 삭제 금지·키 폐기 판단은 유효하나 현재 방식의 비교·구현 근거와 함께 재작성 필요  |
| CASE 32 · 채팅 하이라이트  | 채널 기준선·고유 참여자·수동 대체 경로의 제품 판단은 유효. 팀 연구 대비 본인의 구체적 선택과 반례 검증을 더 확보하면 독립 사례로 강화 가능 |

### 참고로 뒤도 버리지 않은 사실

13번의 인증 지연 주입(10접속·0.1/0.3초 → 마지막 수락 1.059/3.057초)은 당시 진단 기록으로 남깁니다. 32번은 정확도·효율 개선 실측과 본인 선택의 구체적 근거가 더 필요합니다. 이는 경험 자체를 부정하는 이유가 아니라 지금의 대표 주장과 근거가 맞지 않는 이유입니다.

### 본문으로 되돌린 판단

14번은 층을 나눠 원인을 좁히는 디버깅, 31번은 상시 VAD를 건너낸 범위 선택, 43번은 본인이 검증 범위를 넓히고 유실·대기의 비용을 확인한 경험으로 복구했습니다. 설계와 과거 실험임을 표시해도 본인 판단을 설명할 가치가 있습니다.

사례별 기술 글 전체 모음 ↗

구현 상태·자료 안내 ↗

## 전체 목차 · 문제와 검증

쪽 번호는 PDF의 실제 페이지 순서입니다. 제목을 누르면 해당 사례로 이동합니다.

- 01 김태윤
- 02 프로젝트와 기여
- 03 기술과 활동
- 04 김태윤 · 백엔드 포트폴리오
- 05 PokeClip의 구조와 나의 역할
- 06 직접 만들 범위를 실험 뒤 다시 정했습니다
- 07 완료 알림은 계기, 파일과 DB가 복구 기준
- 08 정상·알림 유실 4조건에서 276파일을 회수
- 09 S3 성공과 DB 완료 사이에서 죽는다면
- 10 검은 화면에 겹친 세 원인을 단계별로 나눴습니다
- 11 오류 0건인데 61.94초가 보이지 않았습니다
- 12 시간값만 바꾸자 39.87초 전체가 잡혔습니다
- 13 Go 재포장: 같은 I/O 범위에서 p50 7.69배
- 14 동시성을 두 배로 늘려도 처리량은 비슷했습니다
- 15 재접속의 0.2초 빈틈은 태그만의 문제가 아니었습니다
- 16 재생용 파일의 생성과 저장 완료를 분리했습니다
- 17 첫 조각을 놓치는 조회 계약을 정정했습니다
- 18 두 시간축에서 8,020개 기대 결과와 일치했습니다
- 19 식별자를 바꿀 때 소비자의 조회를 지켰다
- 20 캐시 키와 원본 인증 전달을 분리했습니다
- 21 원본 미디어 요청 1,254건이 28건으로 줄었습니다
- 22 이미 캐시에 있는 영상도 권한을 확인했습니다
- 23 두 번째 요청을 곧바로 캐시 적중으로 세지 않았습니다
- 24 새 저장 계층보다 현재 작업 단위를 먼저 검토했습니다
- 25 8개 객체의 다운로드 완료 시간을 비교했습니다
- 26 오디오를 많이 나누는 것보다 편집 목적을 먼저 정했습니다
- 27 트랙 수와 실제 소리의 보존을 따로 확인했습니다

## 전체 목차 · 판단과 기여

쪽 번호는 PDF의 실제 페이지 순서입니다. 제목을 누르면 해당 사례로 이동합니다.

- 28      마이크를 고르는 문제에 상시 VAD가 필요할까
- 29      일회용 연결과 장기 권한을 나눴습니다
- 30      DB를 정본으로 두는 발행 설계
- 31      미완료 조각 뒤의 완료 파일을 먼저 공개하지 않습니다
- 32      긴 원장과 최근 1시간 창의 비용을 나눠 봤습니다
- 33      정리 코드보다 테스트 DB의 소유권을 먼저 확인했습니다
- 34      root 없이 녹화하려면 새 볼륨의 소유권도 맞아야 했습니다
- 35      테스트 성공·커버리지·실행 범위를 따로 다뤘습니다
- 36      릴리스 교체를 9개 의존 계약으로 검증했습니다
- 37      설정 오류가 방송 요청까지 잠복하지 않게
- 38      종료 신호와 작업 완료는 다르다
- 39      인증 기본값의 표현과 공유 참조를 함께 고쳤습니다
- 40      기존 기여를 이어 실제 소스의 온라인 경계로 옮겼습니다
- 41      실패한 명령과 시작조차 못 한 명령을 구분했습니다
- 42      로그를 더 모으는 선택은 종료 비용도 바꿉니다
- 43      유실률과 종료 꼬리를 같은 실행 단위로 봤습니다
- 44      병합된 수정과 채택되지 않은 접근을 구분합니다
- 45      사용자별 암호에서 보안과 대기 비용을 함께 봤습니다
- 46      Shakedown · 분석 결과를 배포와 복구까지 연결
- 47      AI에 말기는 범위부터 제한했습니다
- 48      선택한 설계가 실제 자원에 반영되도록 했습니다
- 49      AWS 설정 적응과 장애 뒤 복구를 분리 검증했습니다
- 50      잘못된 Java 구문이 검사를 통과했습니다
- 51      TLS 연결 성공과 잘못된 호스트 거부는 다릅니다
- 52      백업의 존재 대신 복원한 데이터로 확인했습니다
- 53      재부팅 뒤 복구와 삭제 뒤 미복구를 함께 확인했습니다
- 54      참고 기록으로 남긴 두 사례와 후속 조건