

The background of the slide is a photograph of the Aurora Borealis (Northern Lights) over a frozen, cracked body of water under a starry night sky. The aurora displays vibrant green and blue-green bands of light. The text is centered over this image.

Aurora Engine

Josias Buchli

Wieso Aurora Engine?

Was sie kann

- Video hochladen → automatisch transcodiert → sofort streambar.
- Grundlage für adaptive Qualitätsstufen, Multi-Device-Streaming
- und eine skalierbare Mediaplattform.

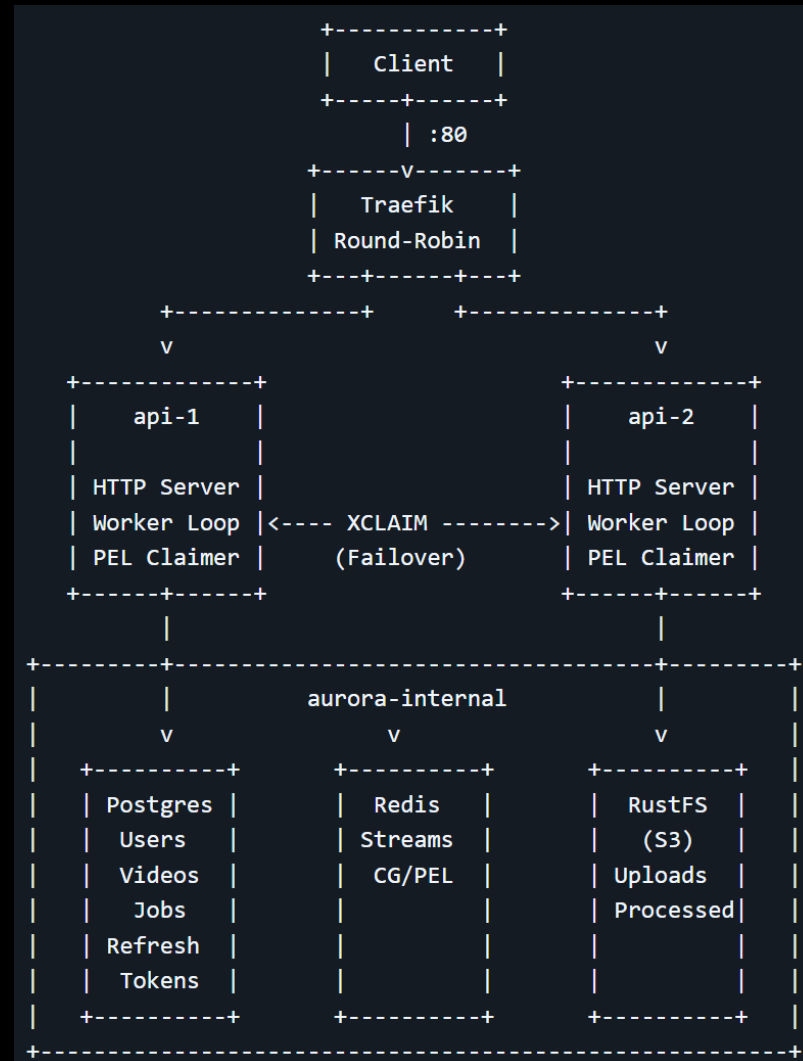
Das Problem

- Transcoding ist langsam und ressourcenintensiv.
- Fällt der Server mittendrin aus, ist das Video kaputt oder verloren.

Die Lösung

- 2 Go-Instanzen teilen sich die Arbeit.
- Stirbt eine, übernimmt die andere – automatisch, ohne Datenverlust.

Architektur-Übersicht



Request-Fluss: Sync vs. Async

Synchron (REST-API):

- Client → Traefik → api-1 oder api-2 → Postgres / S3

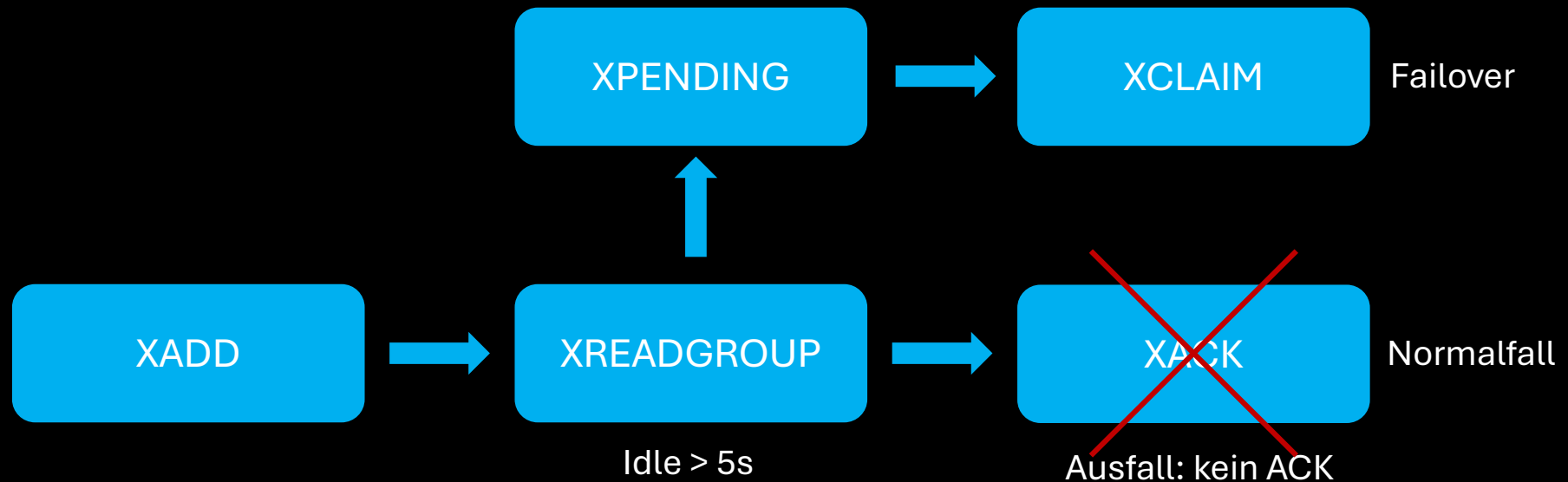
Asynchron (Video-Processing):

1. Upload → Video nach RustFS, Metadaten in Postgres
2. Publish → XADD in Redis Stream "video-events"
3. Consume → Worker liest via XREADGROUP (Consumer Group)
4. Process → FFmpeg: Thumbnail + 720p Transcode
5. Store → Output nach RustFS (processed bucket)
6. Acknowledge → XACK → Message aus PEL entfernt



Failover-Mechanismus

Redis Streams: Consumer Group + Pending Entries List (PEL)



Tech-Stack

Komponente	Technologie	Version
Backend	Go	1.25
Gateway, Load Balancing	Traefik	3.6
Datenbank	PostgreSQL	18.4
Message Broker	Redis	8.2
Object-Storage	RustFS (S3-API)	latest
Video Processing	FFmpeg	7.1
Auth	JWT (HS256)	jwt/5
Frontend	Plain HTML / JS / SCSS	-
Load Testing	hey	latest

Designentscheidungen

Entscheidungen	Begründung
Eigenes HTTP-Framework (pkg/framework/)	Volle Kontrolle, kein Framework Overhead, Trie-basierter Router
Clean-Architecture	Handler (Requests), Service (Logic), Repository (SQL)
Distroless Container	Kein Shell, kein apt - minimale Angriffsfläche
2 Netzwerke	aurora-edge (Traefik ↔ APIs) aurora-internal (APIs ↔ DB/Redis/S3)

Load-Test Ergebnisse

[STEP] Running hey: 1000 requests, 50 workers, duration 30s

[INFO] Target: GET http://localhost/api/videos?limit=20 (JWT-protected, DB read)

Summary:

Total: 30.0105 secs

Slowest: 0.1879 secs

Fastest: 0.0013 secs

Average: 0.0076 secs

Requests/sec: 6602.9935

Total data: 9709791 bytes

Size/request: 49 bytes

Latency distribution:

10% in 0.0036 secs

25% in 0.0048 secs

50% in 0.0066 secs

75% in 0.0092 secs

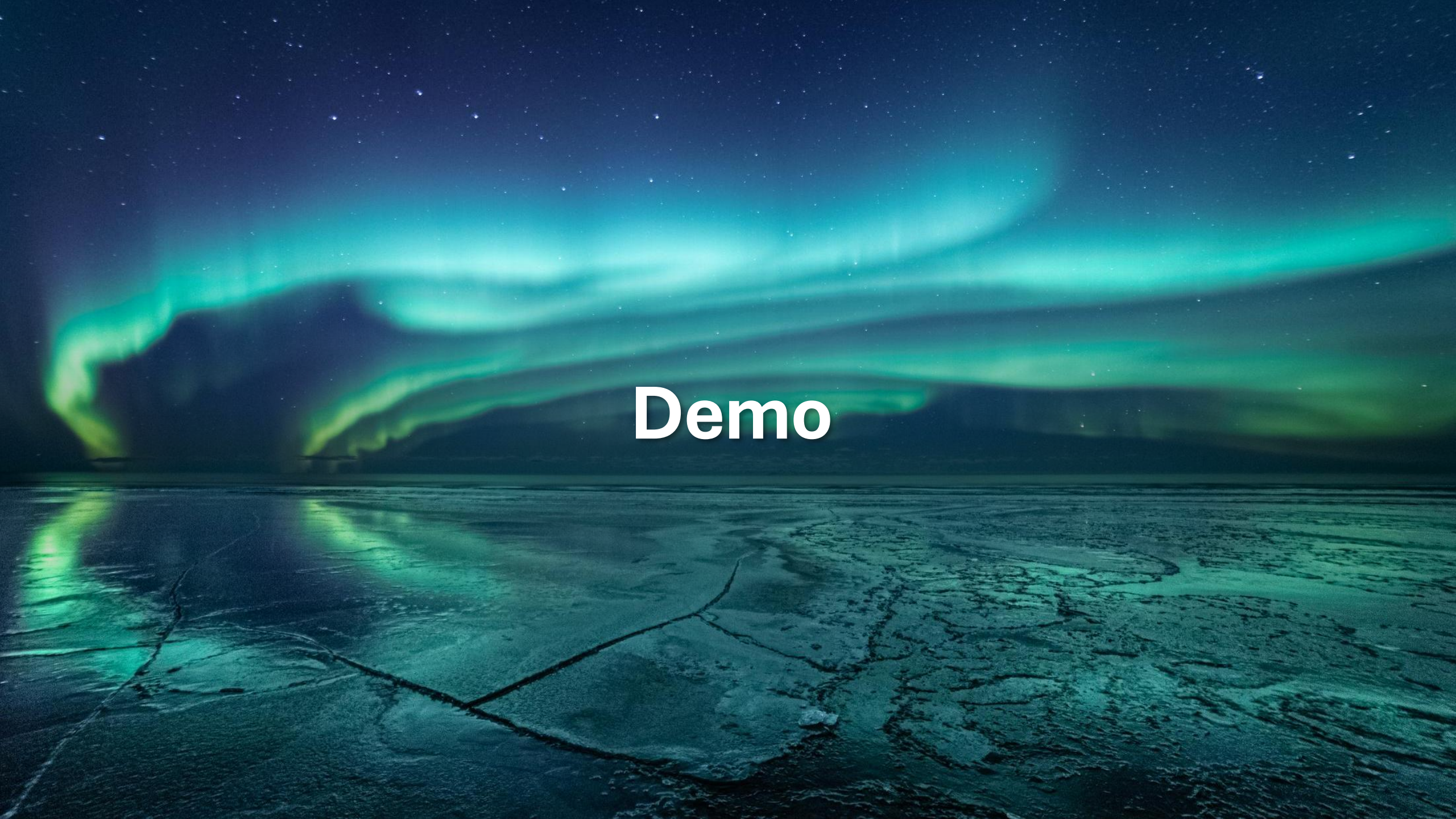
90% in 0.0126 secs

95% in 0.0153 secs

99% in 0.0219 secs

Bottleneck-Analyse

- Nächster Bottleneck → Postgres Connection Pool
- Beim Transcoding: CPU/FFmpeg (anderer Pfad als Load-Test)

A wide-angle photograph of the Aurora Borealis (Northern Lights) over a frozen body of water at night. The sky is dark blue and filled with stars. The aurora appears as vibrant green and blue horizontal bands across the sky, with some wispy, vertical structures on the left. The frozen surface of the water in the foreground is cracked and textured, reflecting the colors of the aurora. The word "Demo" is centered in the middle of the image in a white, sans-serif font.

Demo