# Dead Letter Management At Scale

A Bootiful Case Study

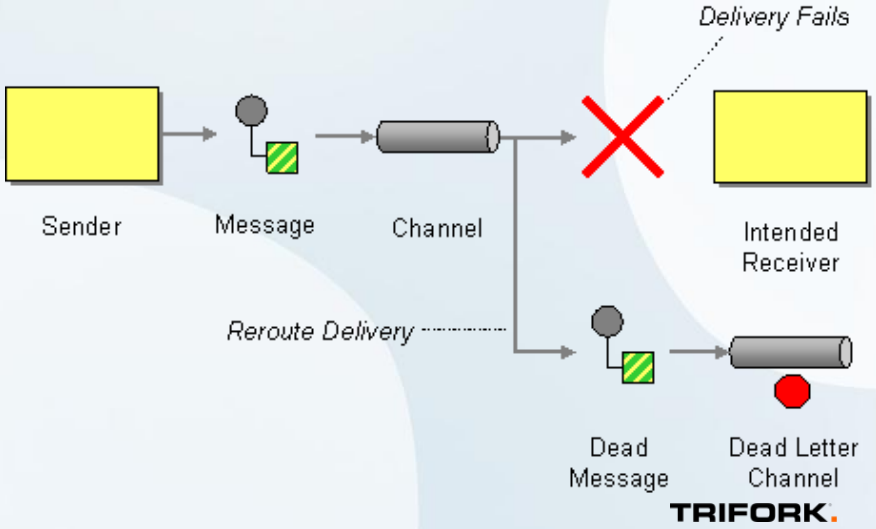
# Context

- Big integration platform for Nederlandse Loterij running on AWS
- **Message queues** for async tasks with retries
  - Send mails
  - Finalize payments
  - Compensating transactions
  - ...

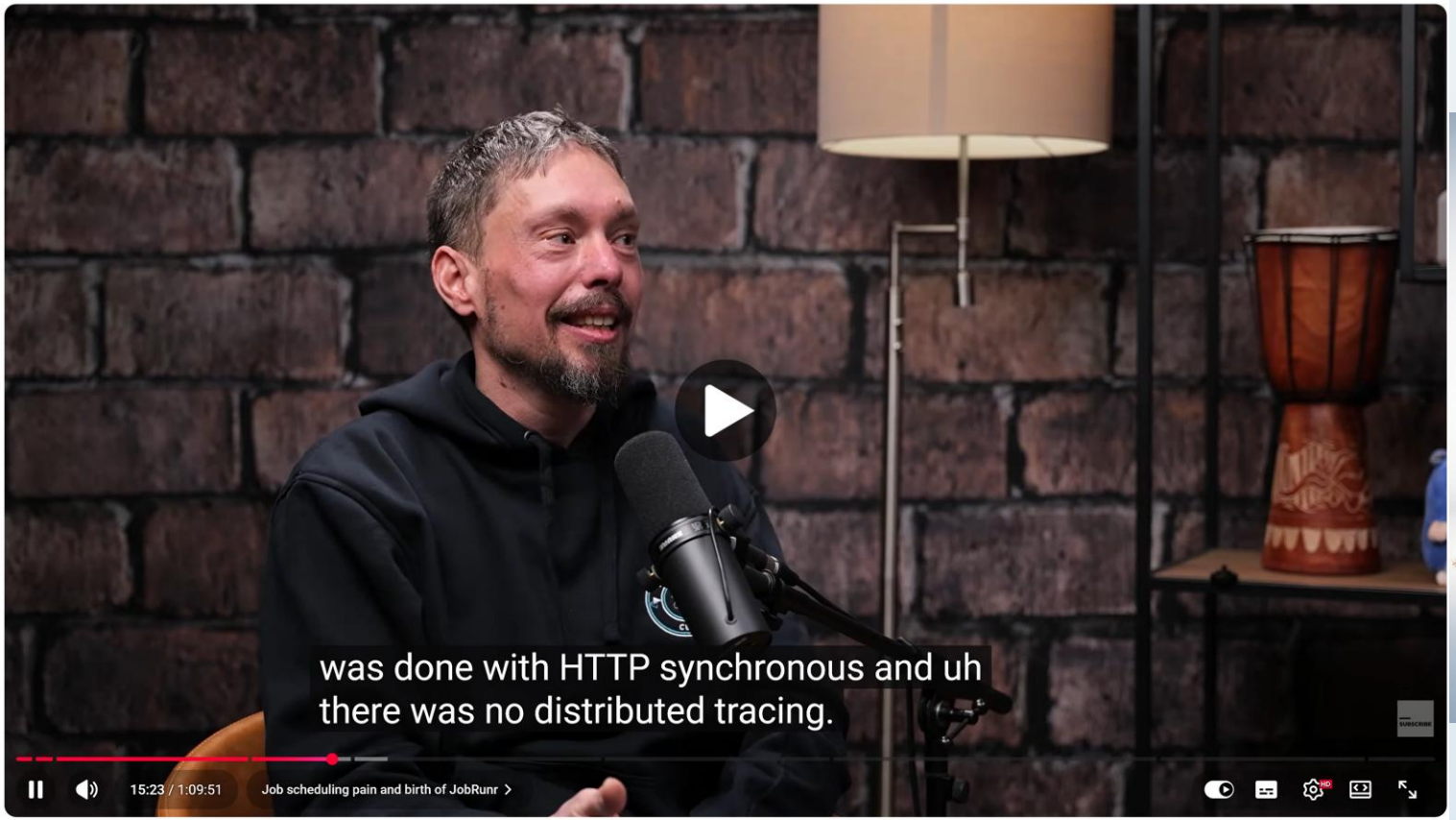

TRIFORK.

# Dead Lettering

- Message handling may fail repeatedly
- Becomes **Poison Pill**
- So, move to other queue: **Dead Letter Queue**

# Dead Letter Graveyard




JobRunr: Java Job Scheduling, OSS Monetization, $17K Deals – Ronald Dehuysser | The Marco Show

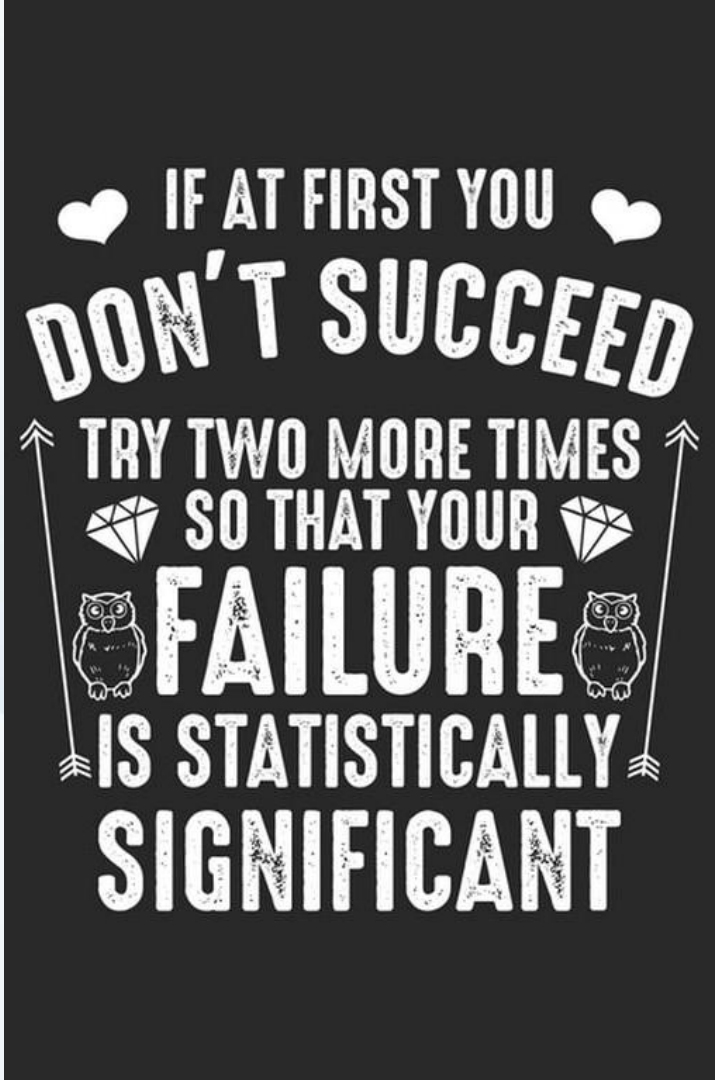
TRIFORK.

# Imagine This At Scale

- 50+ SQS queues / DLQs
- Over 20M messages per month
- Dozens of 3rd party backends
  - Often triggered by messages
- Part of essential processes
  - Ensuring tickets are paid
  - Deposit callbacks

# Why Do Messages Fail?

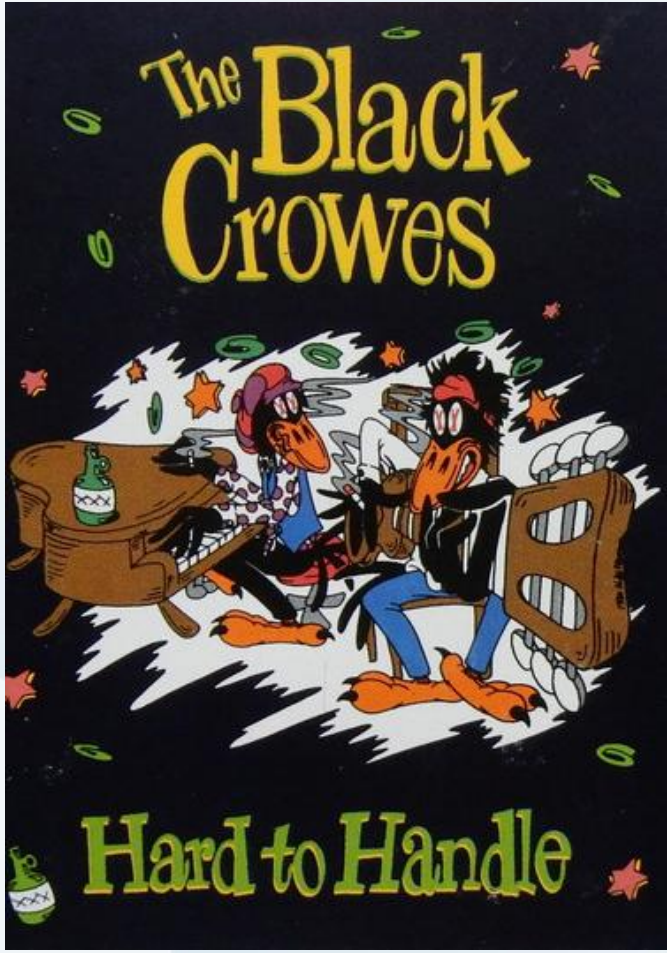
- Transient issues
  - Backend temporarily unavailable
  - API key expired
- State doesn’t match the world
  - Reservation already cancelled
- Problem in sender or receiver
  - Payload uses wrong schema
  - Bugs



# Handling depends on cause

# How To Handle

- Transient errors:
  simply **retry**
  - But not too often
- Other cases:
  find out **cause** & decide
- Sometimes clear from payload
- Typically requires **log inspection**



# Limited built-in support for DLQ handling

- Maybe move back all messages
  - Even that can take years…


- That's usually it…
- Need much more

# Requirements

# Inspection of payload and logs

- Need to know *why* handling fails
- Almost always involves **logs**
  - Payload doesn't reveal much
- In high-volume systems, requires **trace IDs**

# Moving and deleting

- Manually and automated
- Per-queue
- Manually also per-message


# Schedule automatic moves

- Different schedule per DLQ
- Tracking nr. of “de-queues”
  - “move” often results in new message
- Configurable drop after max nr. of dequeues
  - Esp. for non-prod environments!

# DIY

Claude Code v2.1.7

Welcome back Joris!

Sonnet 4.5 · API Usage Billing · Joris's Individual Org
C:\repos\NLO\Dev\sqs-dlq-mgmt

**Tips for getting started**
Run /init to create a CLAUDE.md file with instructions for Claude

**Recent activity**
No recent activity

> Summarize what the current application does

● I'll explore the codebase to understand what this application does.

● Explore(Explore application purpose)
  └ Done (25 tool uses · 37.4k tokens · 1m 32s)

● This is a **web-based AWS SQS Dead Letter Queue (DLQ) Management Tool** built by Nederlandse Loterij (Dutch Lottery).

**What It Does**

The application helps operations teams manage messages that have failed processing and ended up in AWS SQS dead letter queues. It provides a user-friendly web interface to:

**Core Capabilities:**
- **Discover & Monitor DLQs** - Automatically finds all dead letter queues across multiple AWS regions and displays message counts
- **Recover Messages** - Move messages from DLQs back to their original source queues (with protection against repeatedly failing messages)
- **View & Search Messages** - Display up to 200 messages with details including timestamps, trace IDs, and formatted JSON content
- **Delete Messages** - Remove individual messages or purge entire queues
- **Trace Investigation** - Direct links to Datadog logs for troubleshooting

**Technical Stack:**
- Spring Boot 3.5.8 with Java 25
- OAuth2 authentication via Authentik (with role-based access control)
- Multi-region AWS SQS support (default: eu-central-1, eu-west-1)
- Thymeleaf + Bootstrap UI
- Asynchronous background processing for queue operations
- GraalVM native image support

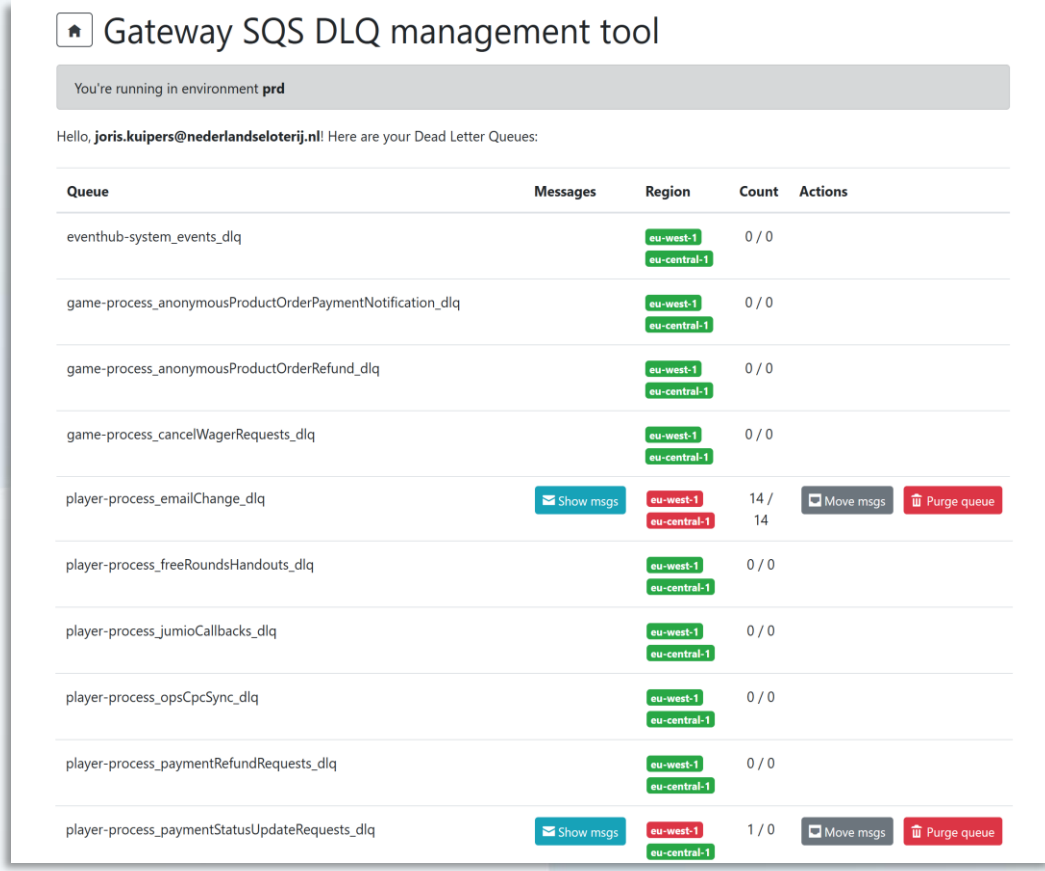
**Typical Workflow:**
1. User logs in and sees dashboard of all DLQs grouped by name with message counts per region
2. Can view detailed messages, move them back to source queues, or permanently delete them
3. Operations run asynchronously in background threads
4. Integrates with Datadog for distributed tracing and log correlation

It's essentially a **queue recovery dashboard** that makes it easy to rescue failed messages and investigate issues in the Dutch Lottery's gateway infrastructure.

✻ Cogitated for 1m 50s

# DLQ Management Tool

- Spring Boot
- MVC / Thymeleaf
- AWS SDK V2
- Spring Security + OIDC
- Multi-region
- GraalVM Native Image
  - because we can



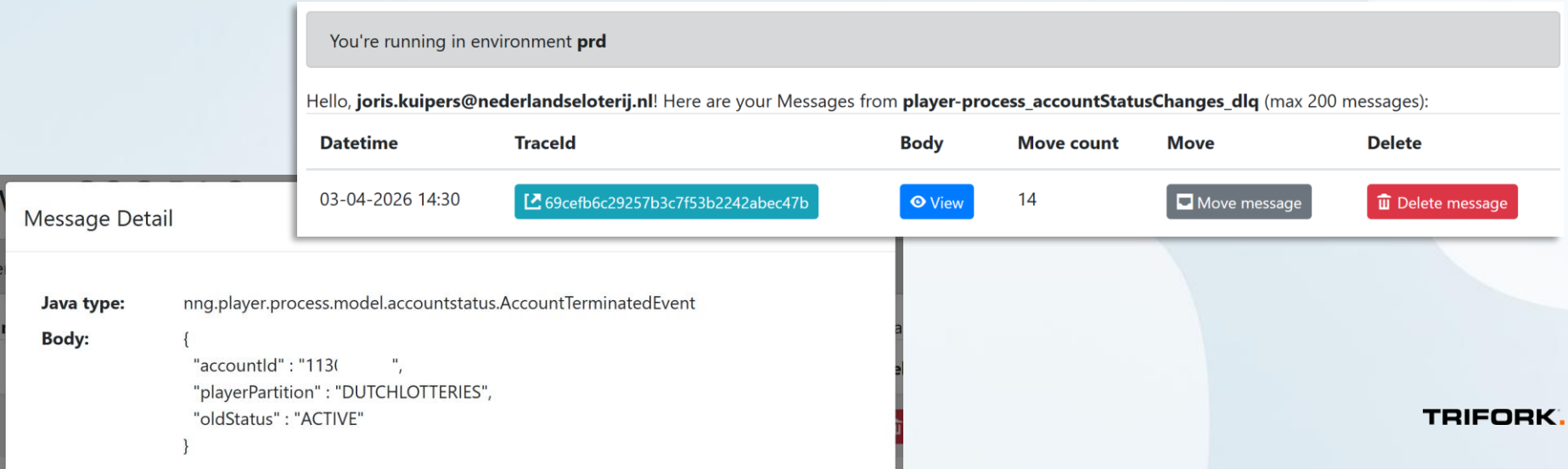
# Scheduled Dequeues

- Move back to source queue, tracking #
  - Defaults to every hour per DLQ
  - Can use different schedule
- Makes system mostly **self-healing**
  - Most dead lettering caused by transient errors
- Drops messages on non-prod environments
  - After max nr of dequeues
  - Prevents endless retries and ever-growing DLQs

# Manual Inspection & Management

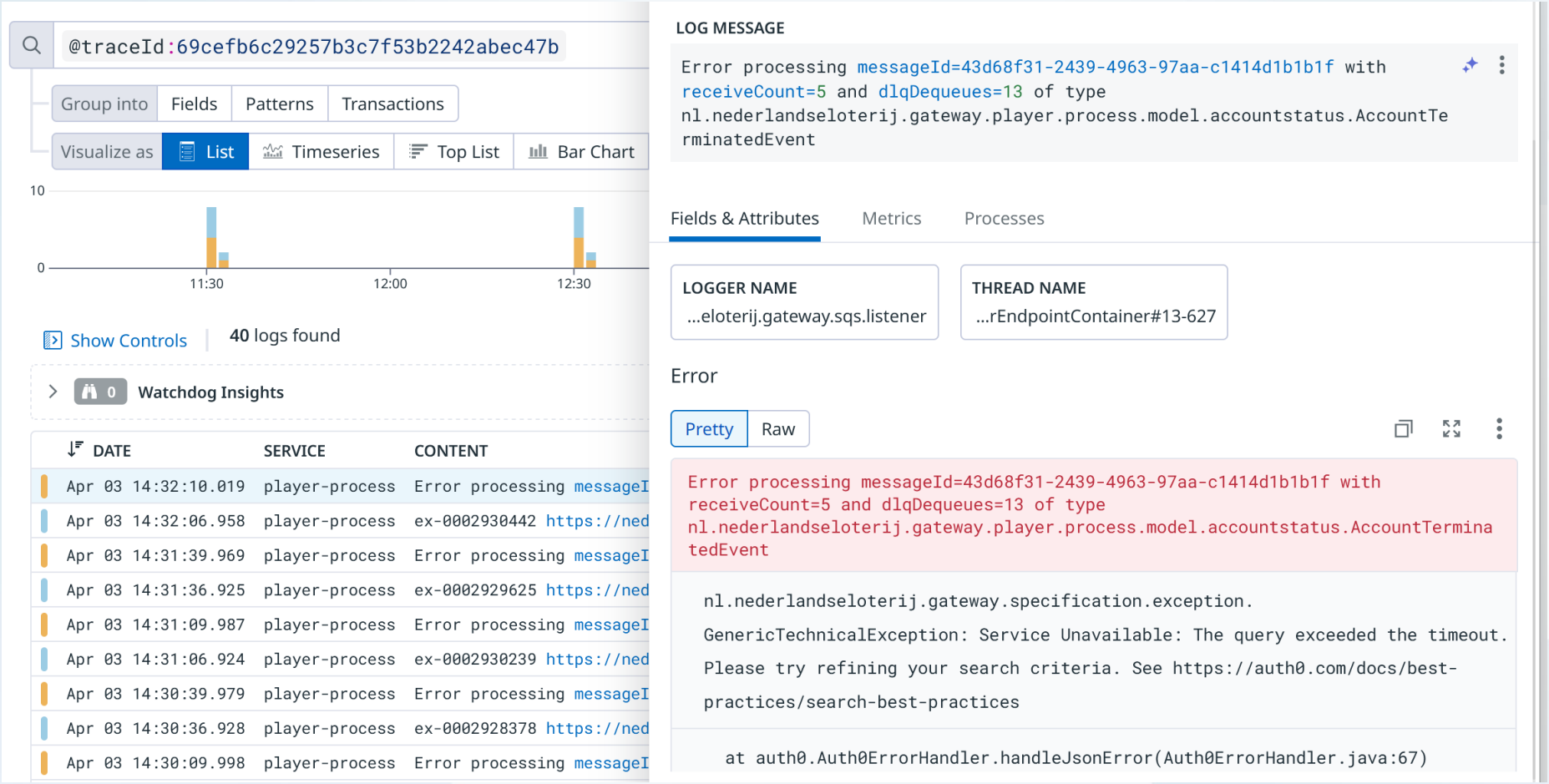
Inspection of payload *and* logs

- Dynamically link to Datadog by trace ID
- Understand *why* msg failed




TRIFORK.

# Log Correlation

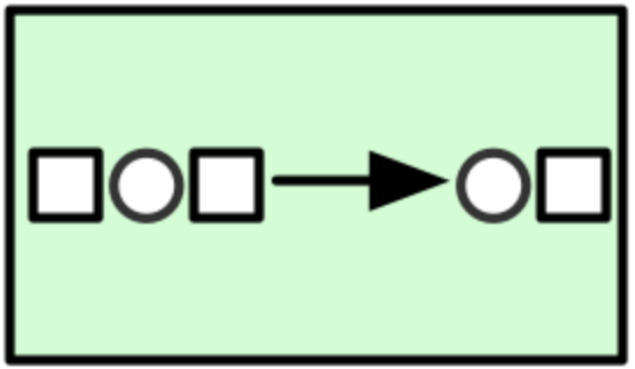


# DEMO & CODE

# Concepts & Gotcha’s

# Idempotency

- Retries should be *safe*
  - E.g. don’t issue two tickets when you order one
- Important even without DLQs
  - Regular messages retried before DL-ed
- Often transaction ID or similar

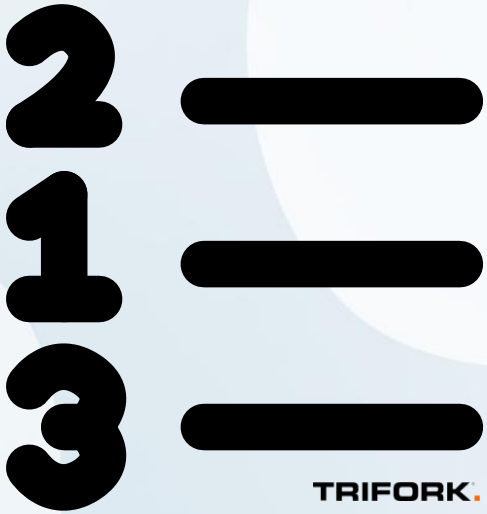


Idempotent Consumer

# Ordering

- DLQs assume out-of-order processing
- Usually happens anyway
  - unless FIFO queue and single consumer
  - Some systems support partitioning

- Blocked on single message?
  You've lost the "at scale" game



# Gotcha's

- Reading without consuming
  - SQS messages temporarily invisible
  - Might have other side effects
- "Move" often means new message
  - Losing message ID & timestamp
  - Copy attrs / headers
  - Track # of dequeues!
- Use bulk operations
  - Faster *and* cheaper

# Gotcha's

- *Essential* to have trace ID in messages
  - Might work out of the box
  - If not, build this yourself!
    - incl. context restore
- Try to derive source queue
- Avoid payload incompatibilities
  - Backwards compatibility
  - Or ensure Qs & DLQs empty on deploy

# Conclusion

📩 Make all message handling **idempotent**
👀 Make full message lifecycle **observable**

🪦 Don’t let DLQs become message graveyards:
🔁 Automate **retries** to handle transient errors
👨‍💼 Provide **management** capabilities
🧹 Auto **cleanup** for non-prod environments

# Thank you!

CTO TRIFORK AMSTERDAM

Joris Kuipers

trifork.nl

@joris.kuipe.rs

TRIFORK.