

The life of a message

Schema, delivery, repeat, replay: a message is tested four times in the queue.

Muhammet Şafak • October 3, 2026



Four stops

- 01 **Schema** validate at the edge
- 02 **Evolution** without breaking the contract
- 03 **Idempotency** the same message twice
- 04 **DLQ replay** reset, select, test



Schema

The queue doesn't look inside the payload; you add the shape guarantee at the edge.

THE QUEUE CARRIES BYTES, NOT
TYPES

Where is the schema validated?

An upstream service dropped a field
from the payload; the worker threw a
KeyError in production.

Producer side

primary defense

Consumer side

safety net → DLQ

Schema registry

per message type

99

The consumer side is the
seatbelt, **not the steering.**

— **Muhammet Şafak** Validating the schema at the edge of the queue · sade.dev

LOOSENING IS SAFE, TIGHTENING IS NOT

Changing a schema without breaking it

✓ Safe

Add an optional field

Drop the required constraint

Widen an enum, relax a minimum

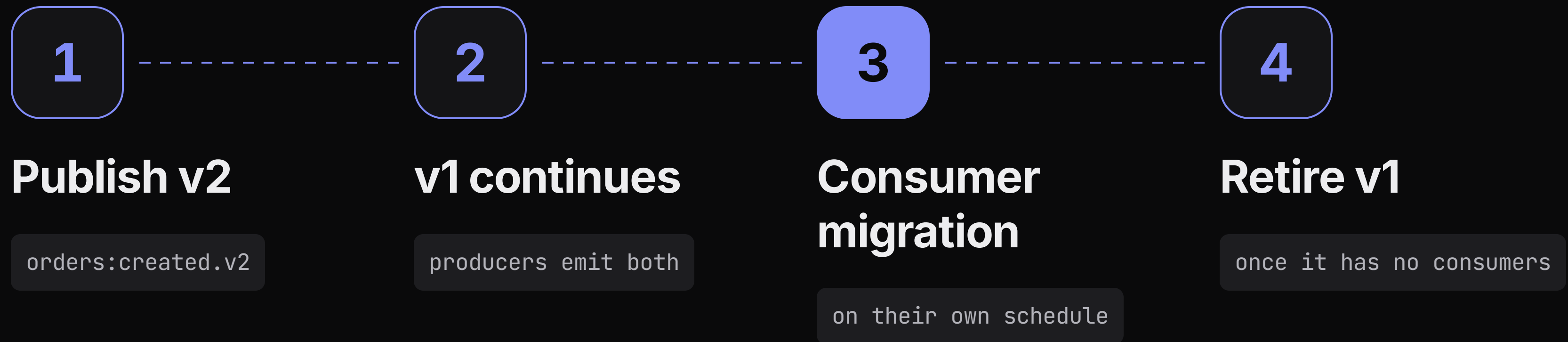
✗ Breaking

Add a required field

Remove, rename or retype a field

Narrow an enum, tighten a rule

A new identity, parallel lives



Repeat

The broker keeps its promise: a message arrives at least once, sometimes twice.

The same message
can arrive twice.

The charge must not
happen twice.

The payment worker charged the card and crashed before it could ack the message; the broker redelivered it, and the customer was charged twice.

$F(F(X)) = F(X)$

Idempotent or not?

	Not idempotent	Idempotent
EXAMPLE	Increase the balance by 10	✓ Set the order status to paid
TWICE	Adds 20, or two emails	✓ Same as the effect applied once
NEEDS	A key + dedupe	✓ Possibly nothing

When do you record the key?

SHAPE	WHEN RECORDED	WINDOW	FITS
Seen-set	After the handler returns successfully	Yes	Third-party calls: email, charge
Transactional	In the same DB transaction as the change	None	A row you own

Key and constraint



SENDER

The sender produces the key: the message id or the Idempotency-Key header.



PAYLOAD

Don't derive it from the payload alone: two legitimately identical requests collide.



CONSTRAINT

A constraint, not a check: a unique constraint turns a concurrent second insert into a conflict.

Replay

Replay isn't moving bytes back: it's resetting a message and safely reprocessing it.

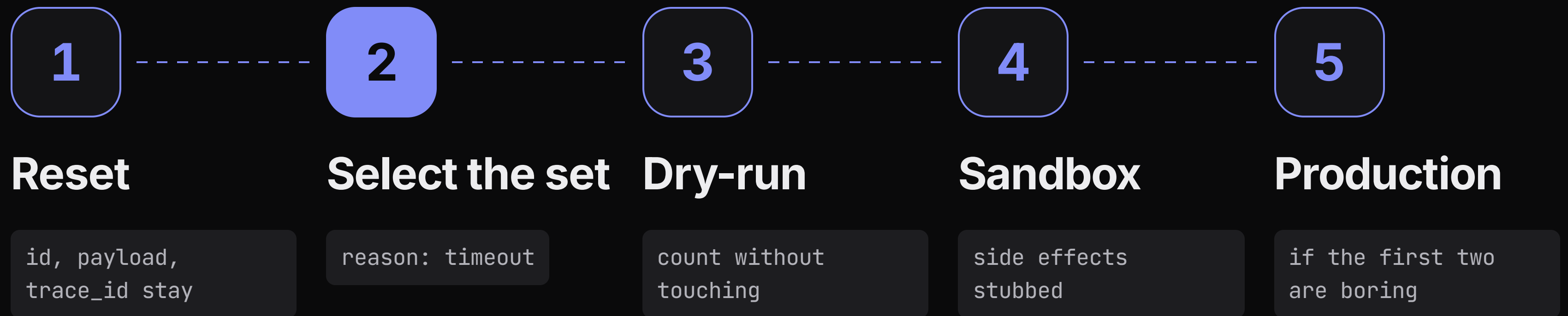
20

EXAMPLE SCENARIO

minute outage

The downstream API was down for twenty minutes; four thousand messages used up their retries and landed in the DLQ.

Replaying a DLQ safely



Thank you

sade.dev

Validate the schema at the edge, suppress repeats, replay by testing first.

