

When Event-Driven Systems Become the Source of Truth:

Lessons from CDC, Kafka, and
Replay-Safe Inventory Platforms

Ishan Shah

Software Engineer - Distributed Systems, Event-Driven Architecture, Platform Engineering



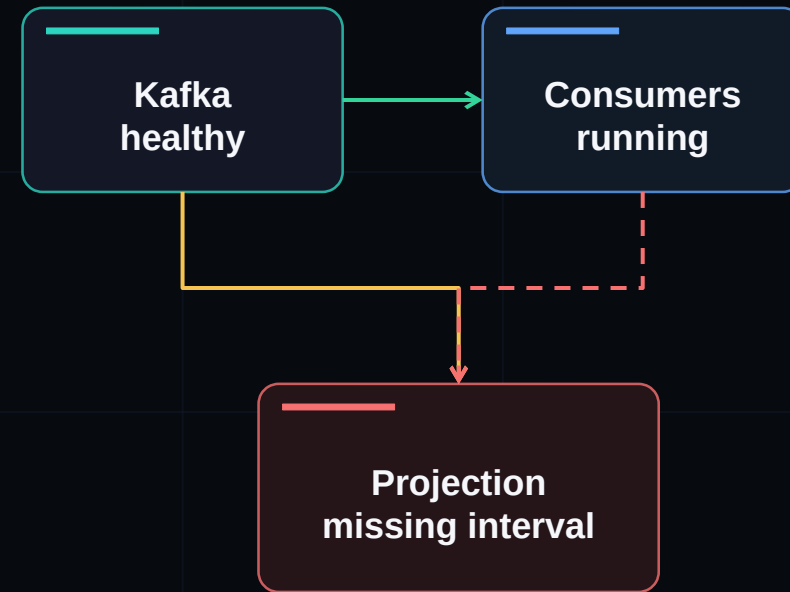
Second axis of reliability

Kafka was healthy. Inventory state was not.

**It is 2:13 AM. Brokers are green.
Consumers are running. Lag is
low.**

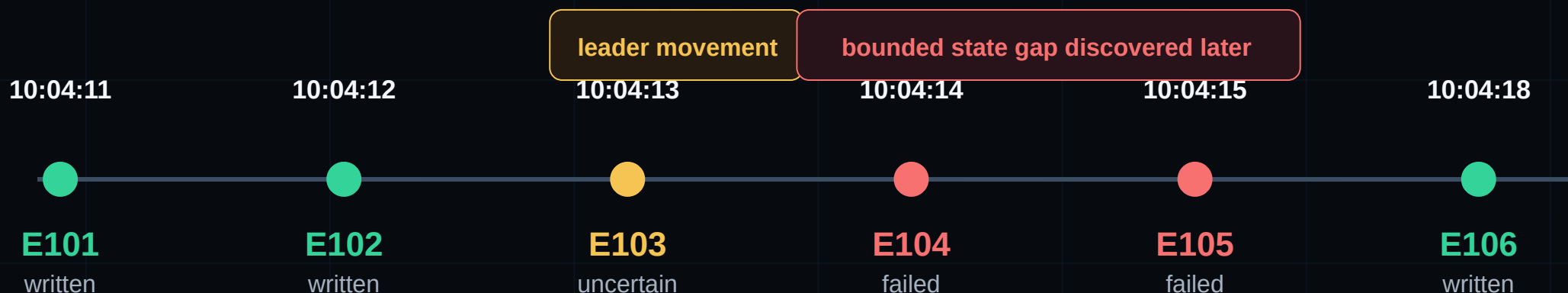
Then Finance finds a bounded inventory gap in
one downstream projection.

**You still have the events. Can you replay
them without duplicating business effects?**



**The system can be alive and
still be wrong.**

The hardest record is the one with an ambiguous outcome.



What range do you replay?

Did E103 already change the sink?

The obvious move is to reset offsets. That is why it is dangerous.

```
$ kafka-consumer-groups \  
  --reset-offsets --to-offset 812400 \  
  --execute
```

Obvious solution

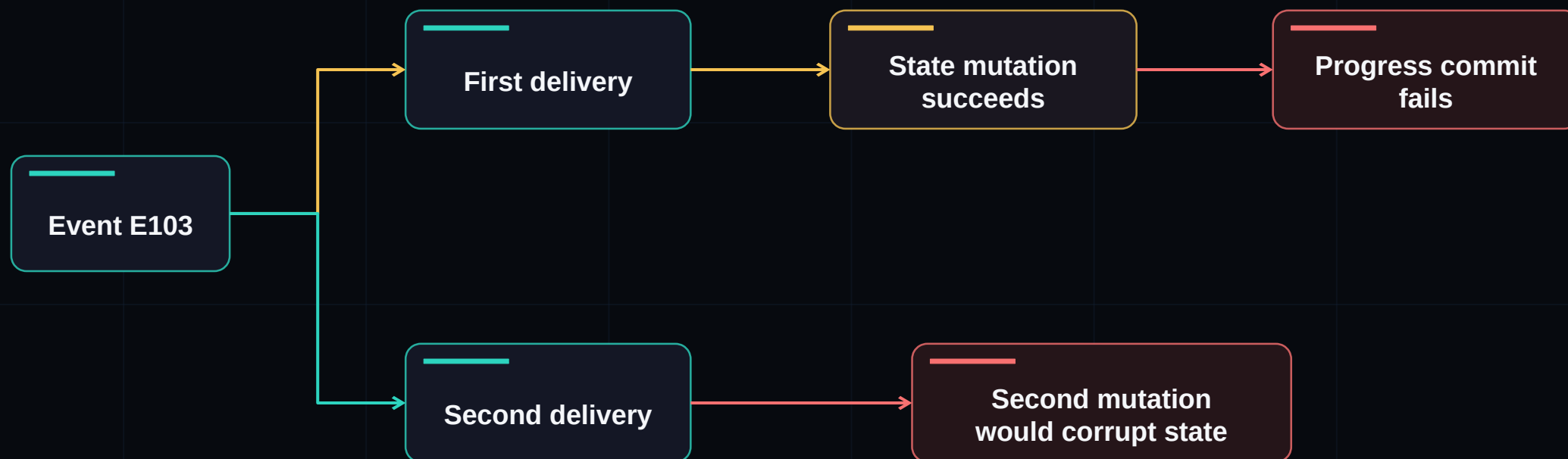
The broker did what we asked.

Historical records
delivered again

Business effect
executes again

The most dangerous recovery plan is: just replay it.

Delivery is a transport fact. Effect is a business fact.



The second delivery is expected.

The second business effect is the bug.

One key cannot carry four different meanings.

TransferShipped

event_id: EVT-103
operation_id: T-4821
event_type: shipped
source_sequence: 17
transfer_id: TX-77

event identity

Is this the same business fact?

aggregate identity

Which state object owns it?

ordering key

Which events must be sequenced?

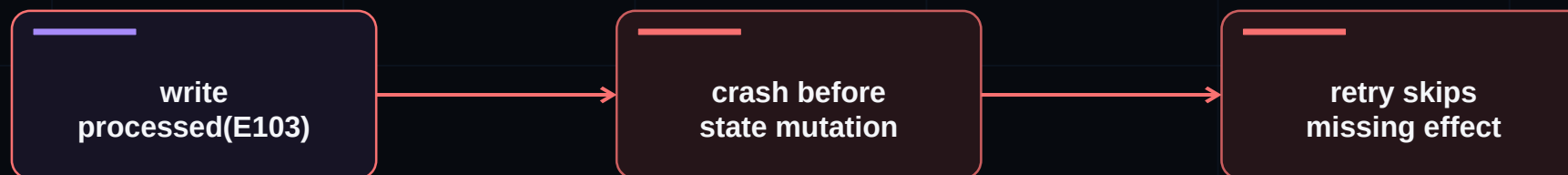
idempotency key

Which effect must not repeat?

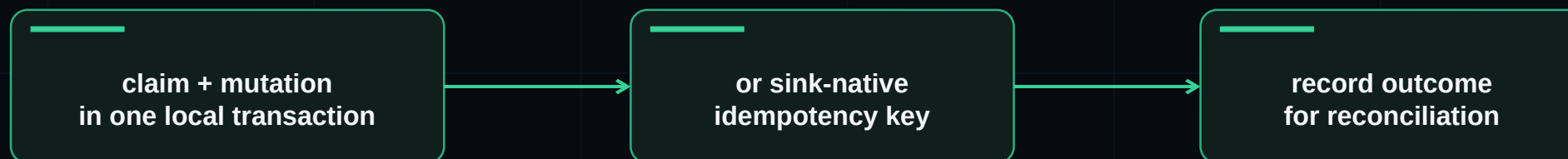
A new delivery UUID is not a stable business event ID.

Dedupe can create a new ambiguous outcome if the boundary is wrong.

Bad boundary



Good boundary



A dedupe cache that forgets too soon turns history into a duplicate.

Dedupe memory



24 hours

E103
forgotten

Replay need



six-month damaged interval

E103
replayed

Retain IDs to the horizon, or use sink-native/natural idempotency.

Lifecycle order comes before partition order

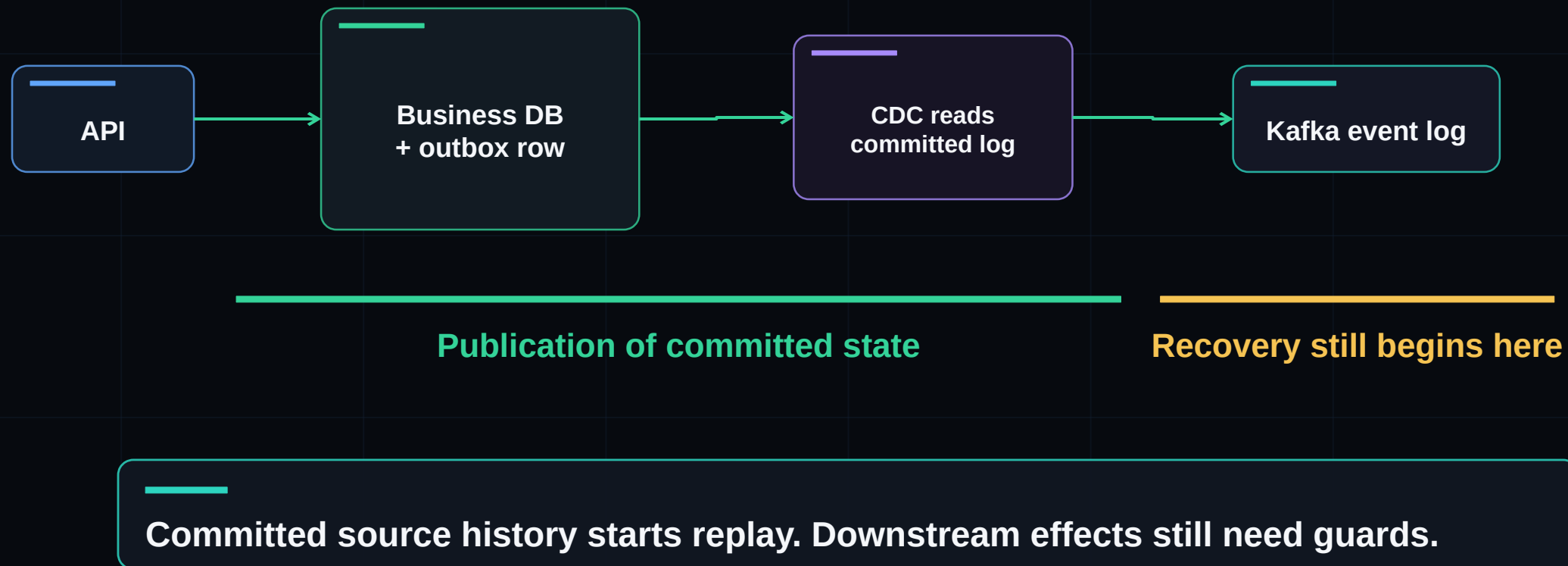
operation_no = the ordering lane



**Replay Received before
Shipped**

**Replay stale event after live
change**

Outbox and CDC protect committed publication



The same event can produce two correct-looking but different answers.



Replay must choose history or policy migration explicitly.

External effects require an explicit boundary because no shared transaction exists.

Local state

**claim effect
+ mutate state
+ record outcome**

one local transaction

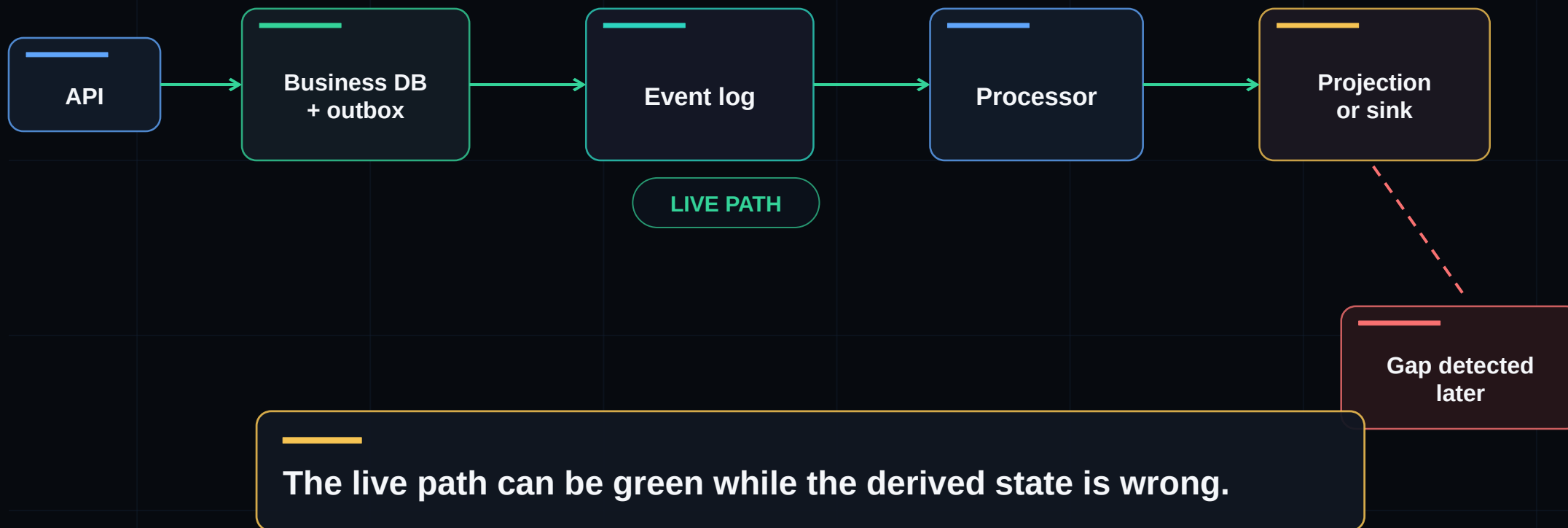
External sink

**idempotency key
or observable outcome
or compensation**

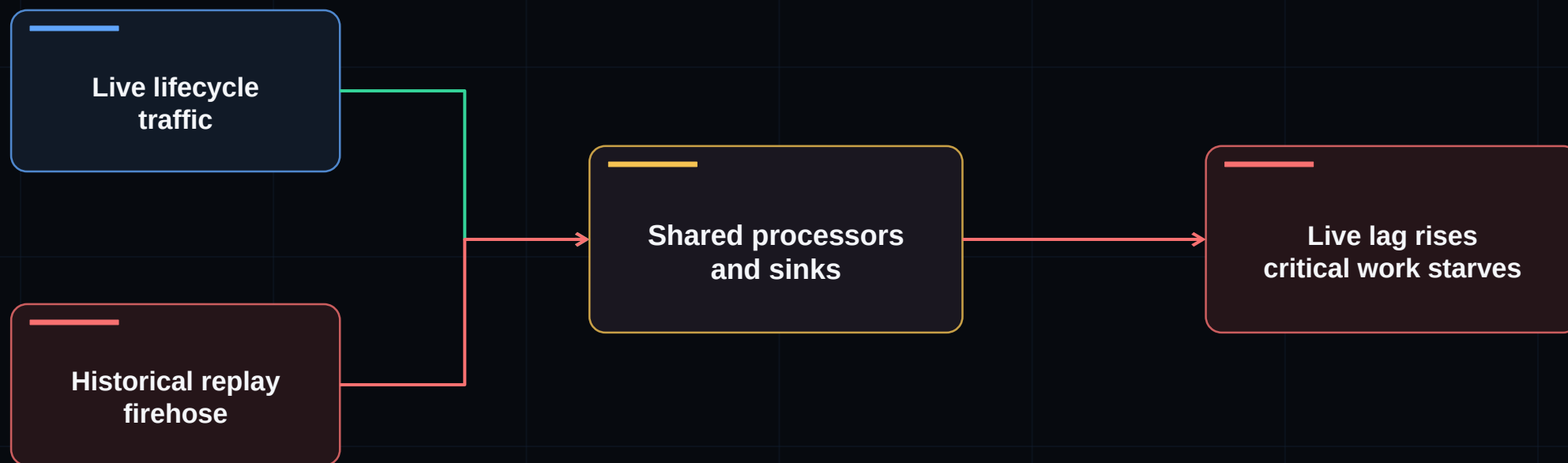
provider boundary

Exactly-once transport is not exactly-once business effect.

Here is the system we are trying to repair.



A correct replay can still become the next incident.



The forced properties

bound

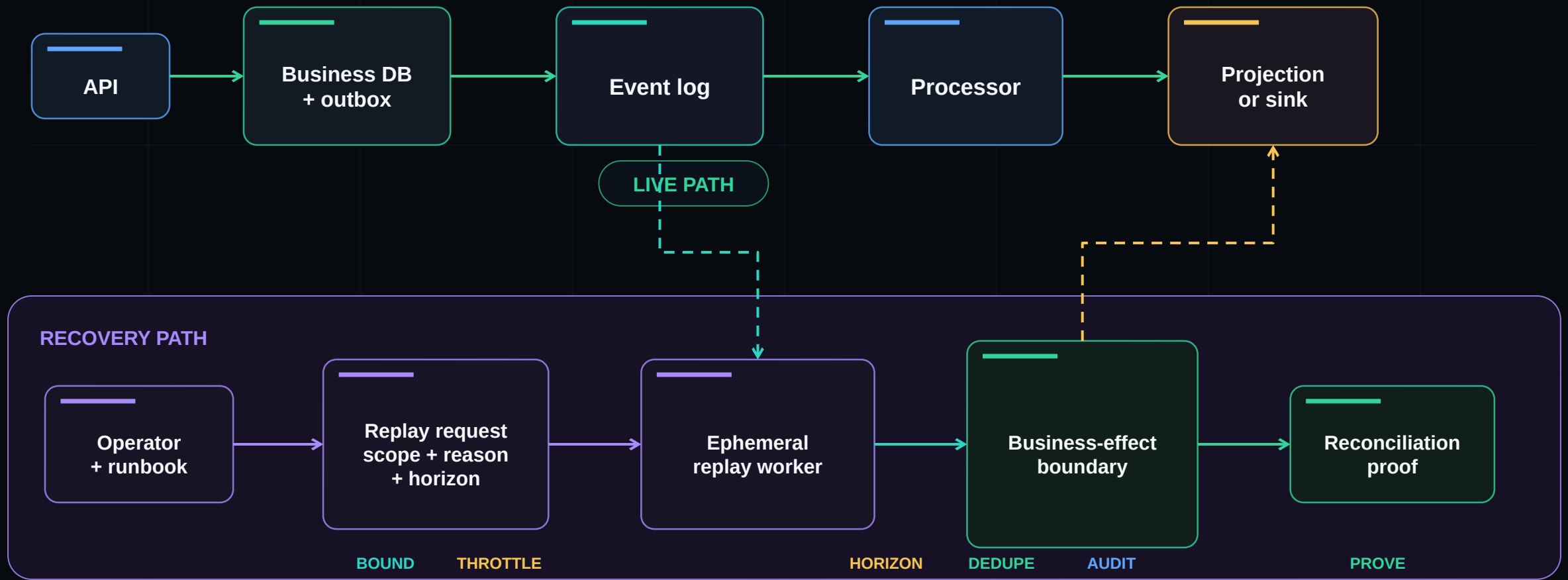
throttle

isolate

gate

observe

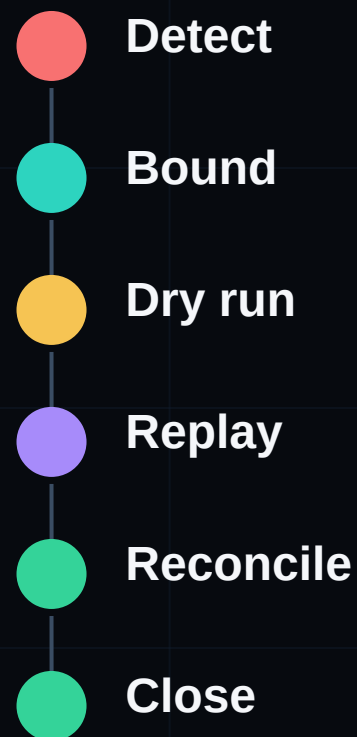
Recovery becomes a controlled production change



The runbook is an audit artifact, not a memory of what happened.

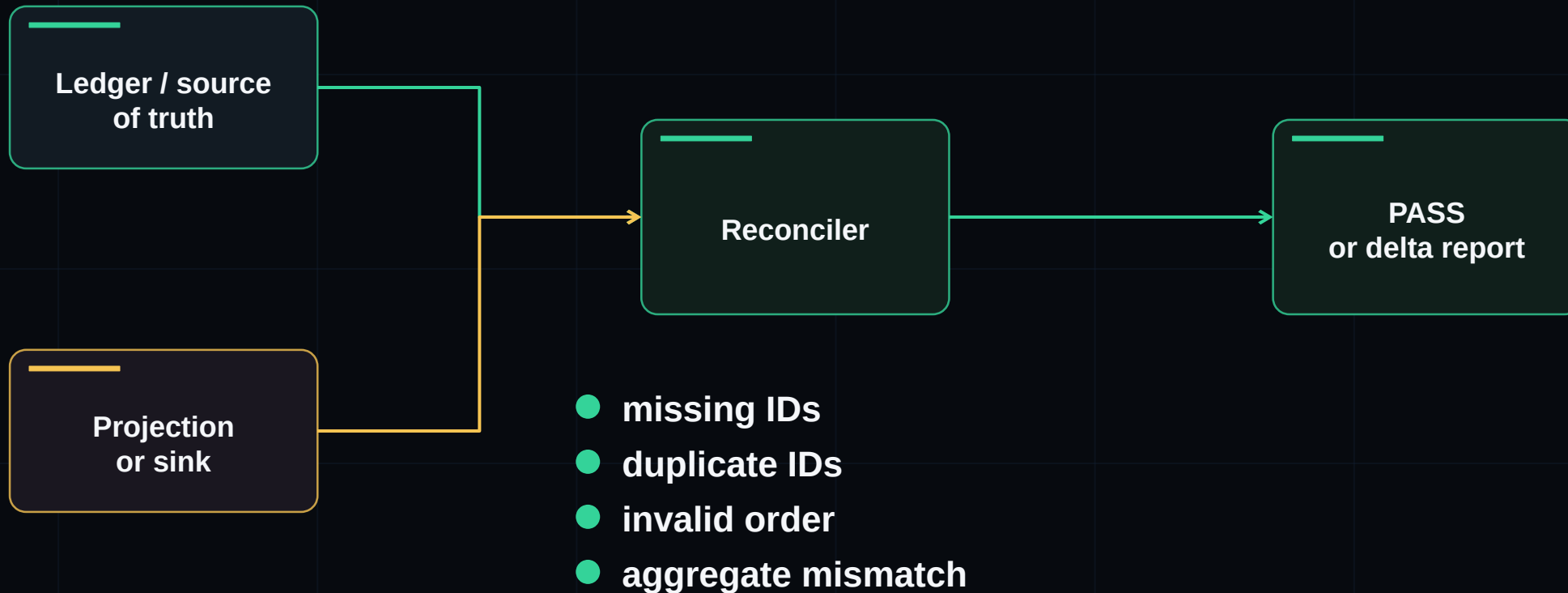
replay_id: INC-4821

scope	topic, partition, offsets, event types
mode	dry run -> approved repair
limits	rate, quota, priority lane
guard	identity, order, dedupe, stale reject
result	applied, skipped, rejected, quarantined
proof	reconciliation report + deltas



Evidence records. Proof closes.

Replay repairs state. Reconciliation proves it.



Lag = 0 only proves processing position.

Before you call a system replay-safe, answer these.

01 What uniquely identifies the business event?

02 Which aggregate owns the state transition?

03 What key preserves the lifecycle order?

04 Which effect key prevents duplicate mutation?

05 How long must dedupe remember?

06 Where is authoritative committed history?

07 Do old events use old meaning or a migration?

08 Where do external side effects become repeat-safe?

09 How is replay bounded, throttled, and isolated?

10 What proves the repaired state is correct?

A system is not recoverable because it kept the events.

It is recoverable when replay is bounded, repeat-safe, controlled, observable, and proven by reconciliation.

Questions