

gRPConf

INDIA 2026

@HarishKotra

gRPConf INDIA 2026

Dead Reckoning for gRPC Streams

Resumable AI Inference



@HarishKotra

DISCLAIMER

The views expressed are my own. If you disagree, feel free to fork them, open an issue, or submit a PR.



Who am i?

Developer Experience Leader & Consultant

Previously

- DevX Lead @ quirq.ai
- Developer Relations Lead @ gaianet.ai
- Community & Strategy @ AngelHack
- & others...

Organized over 200+ large scale public hackathons for many Fortune 500 companies over 13+ years.

Contributor @hermes, @passmark, @open-metadata, @widemem-ai



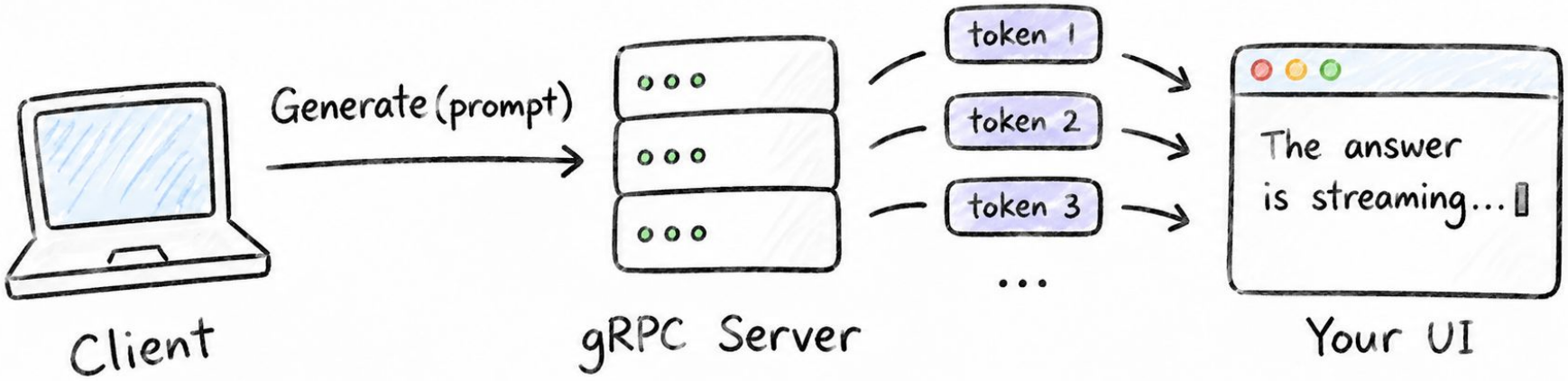
@HarishKotra

AGENDA

- The Problem with Streaming
- Retry vs. Resume
- Designing Resumable Streams
- Lessons from Building It
- Live Demo

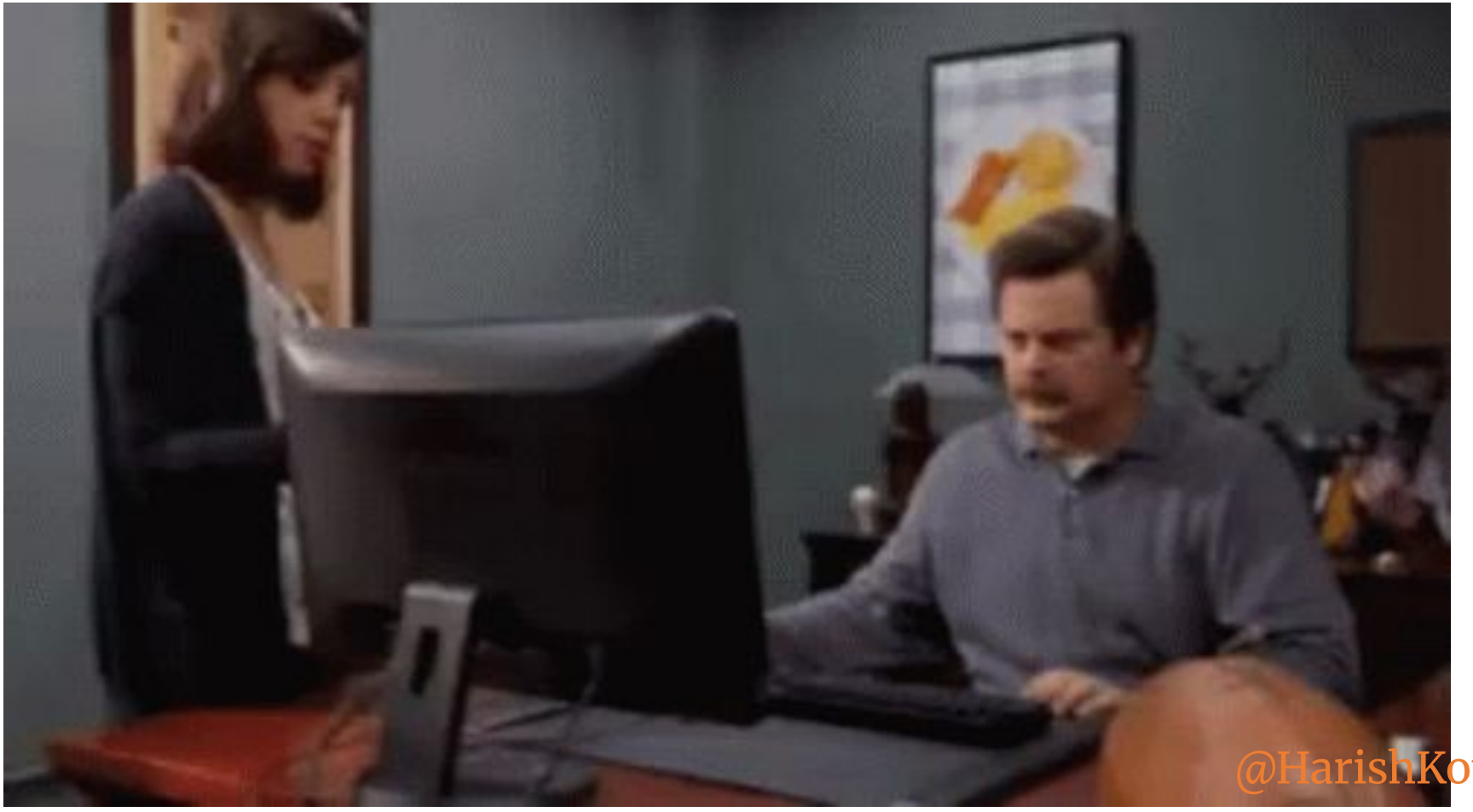


Streaming is easy. Recovering is the problem.



One request. Many responses.

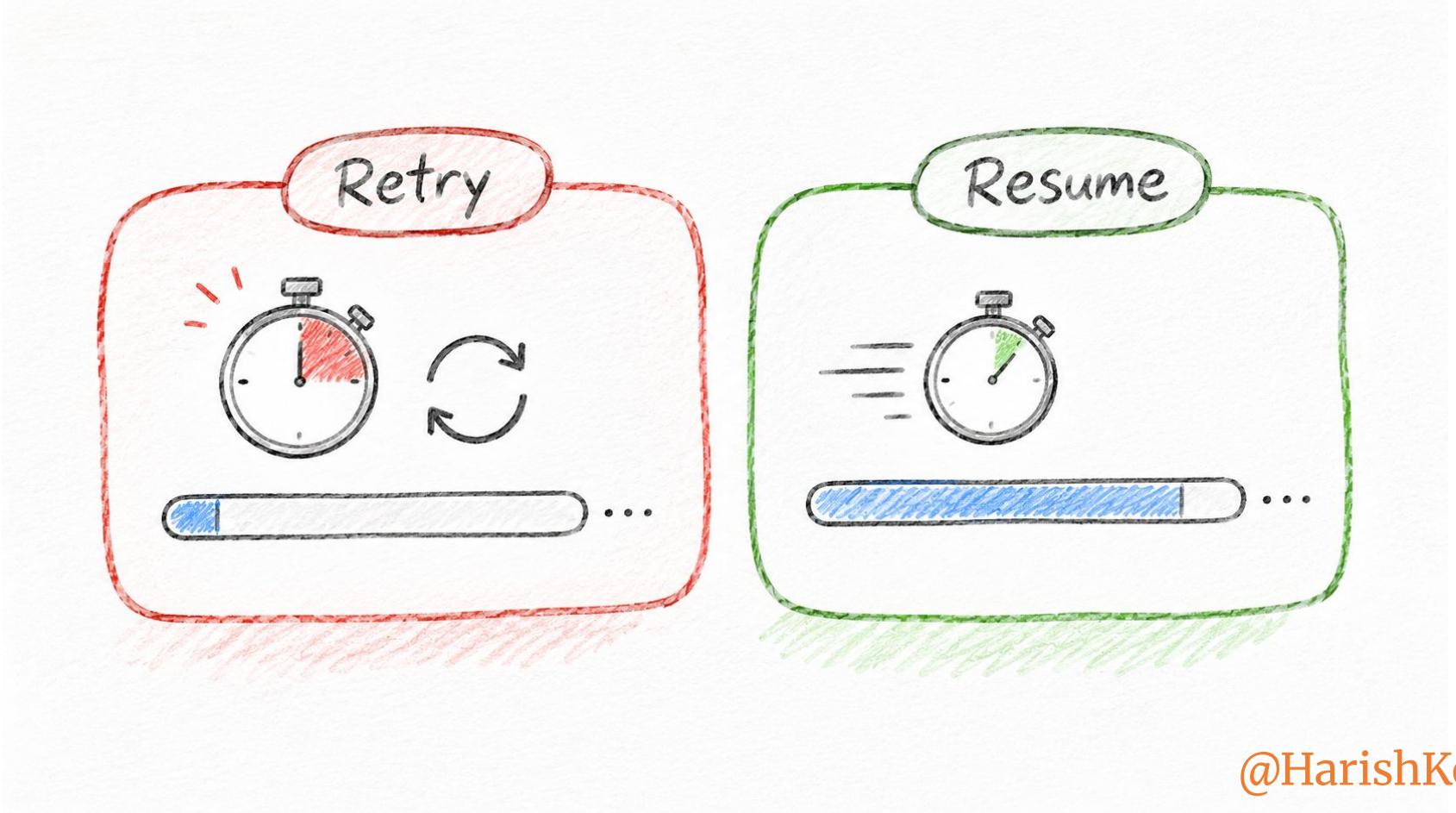
And then, one fine friday!



The failure isn't the connection. It's the lost progress.

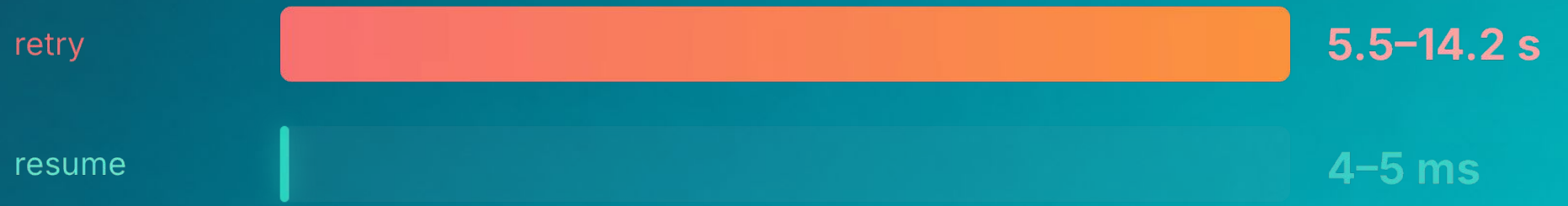


The cost of starting over

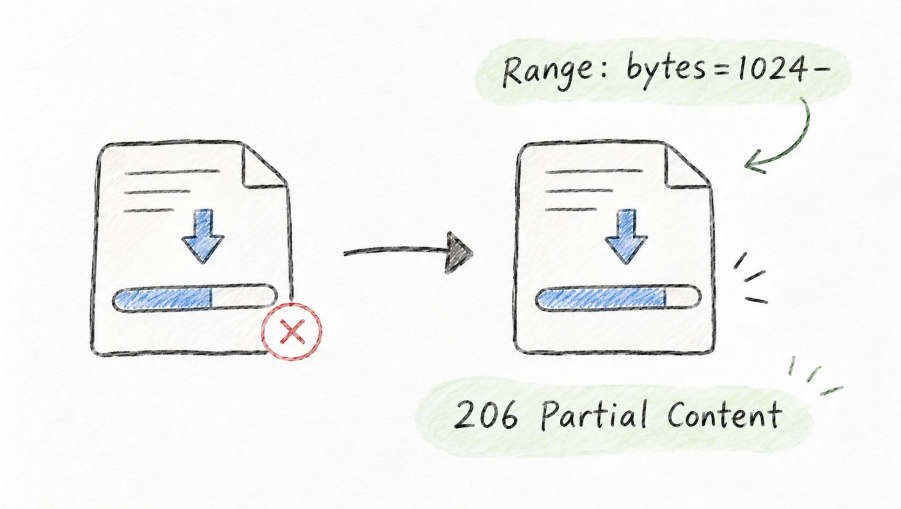


The cost of starting over

gRPConf INDIA 2026



HTTP has had this idea for decades



HTTP/1.1 · 1999

Range: bytes=1024-

206 Partial Content

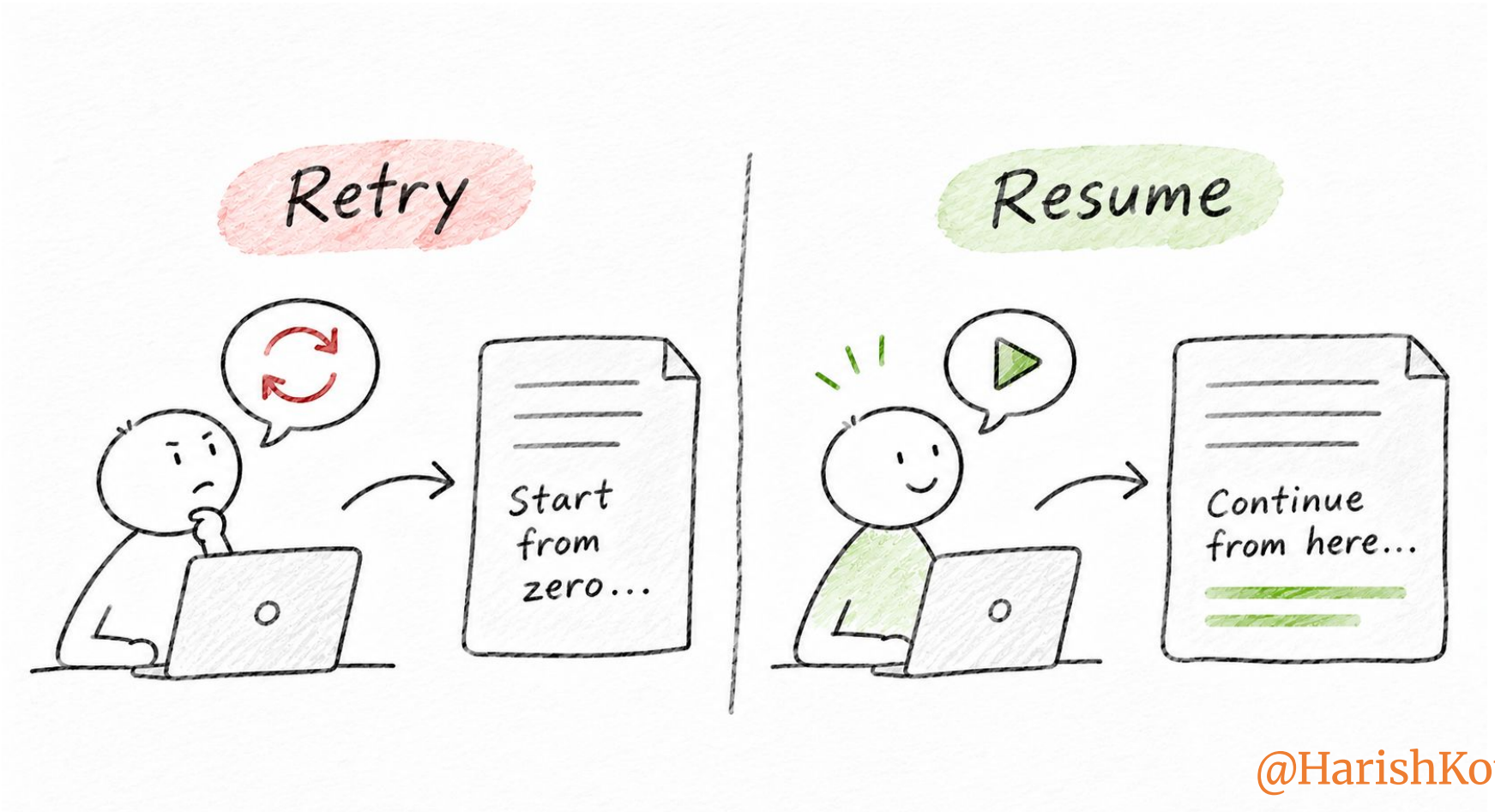
continues at byte 1024

gRPC · today

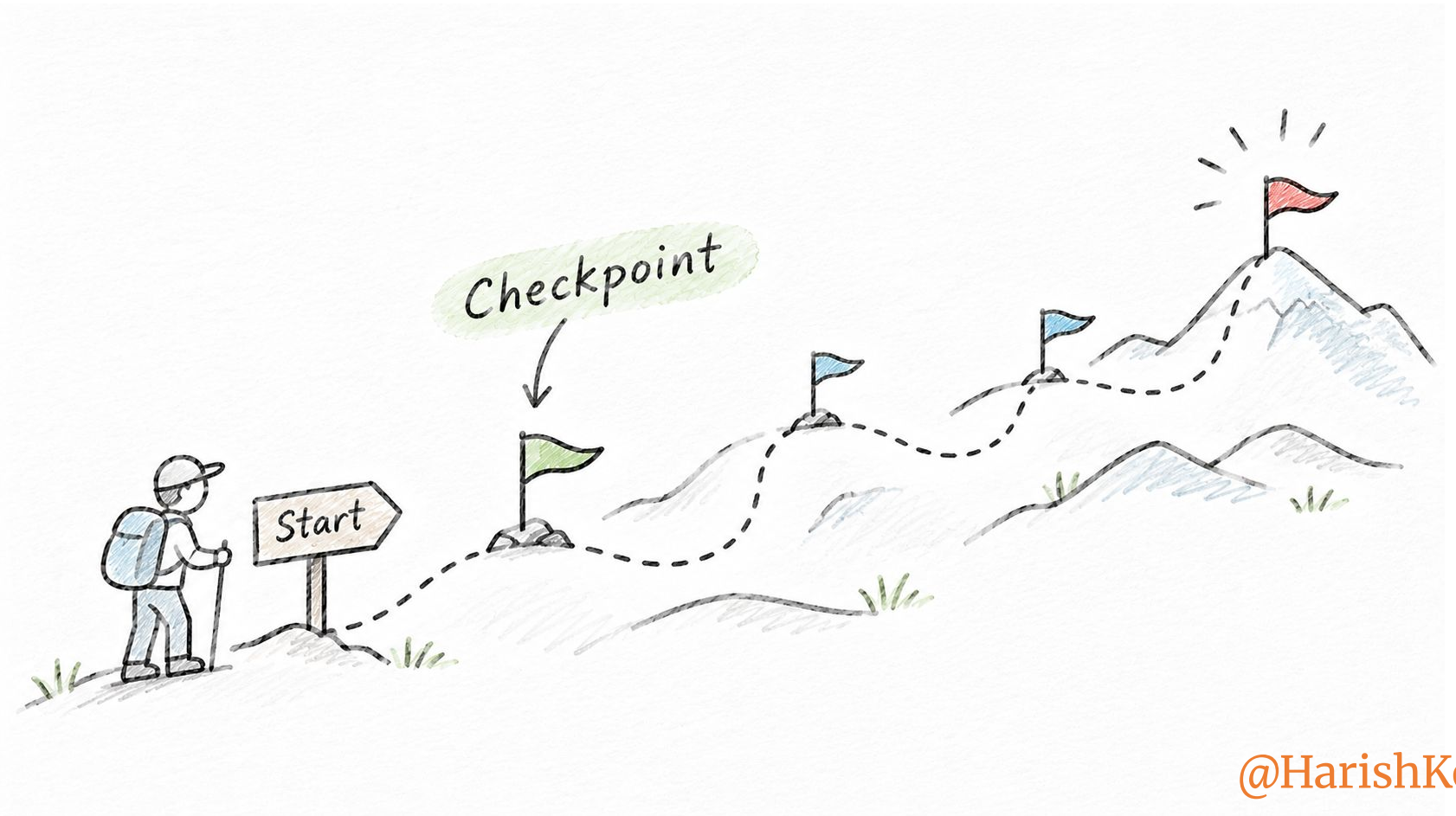
Generate(stream) again

no offset. no resume.

Retry is not response resumption

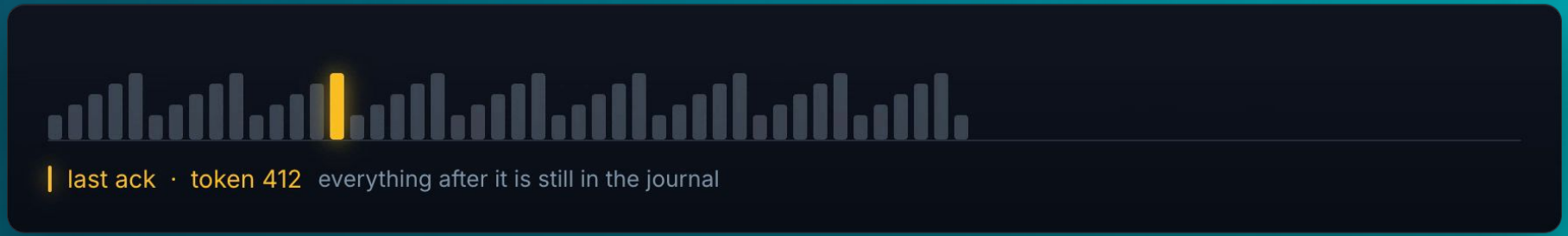


The reframe: checkpointing, not retrying



The reframe: checkpointing, not retrying

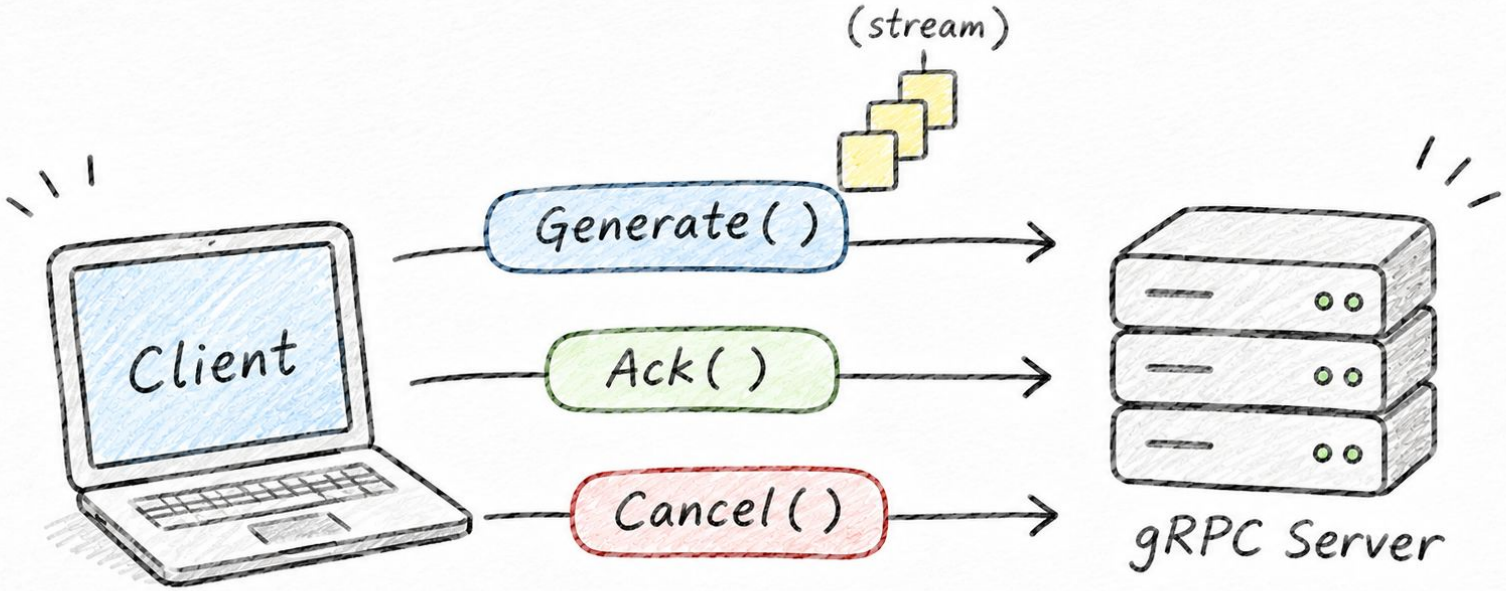
gRPConf INDIA 2026



sailor a fix, then dead reckoning

stream a cursor, then the journal

Designing the protocol



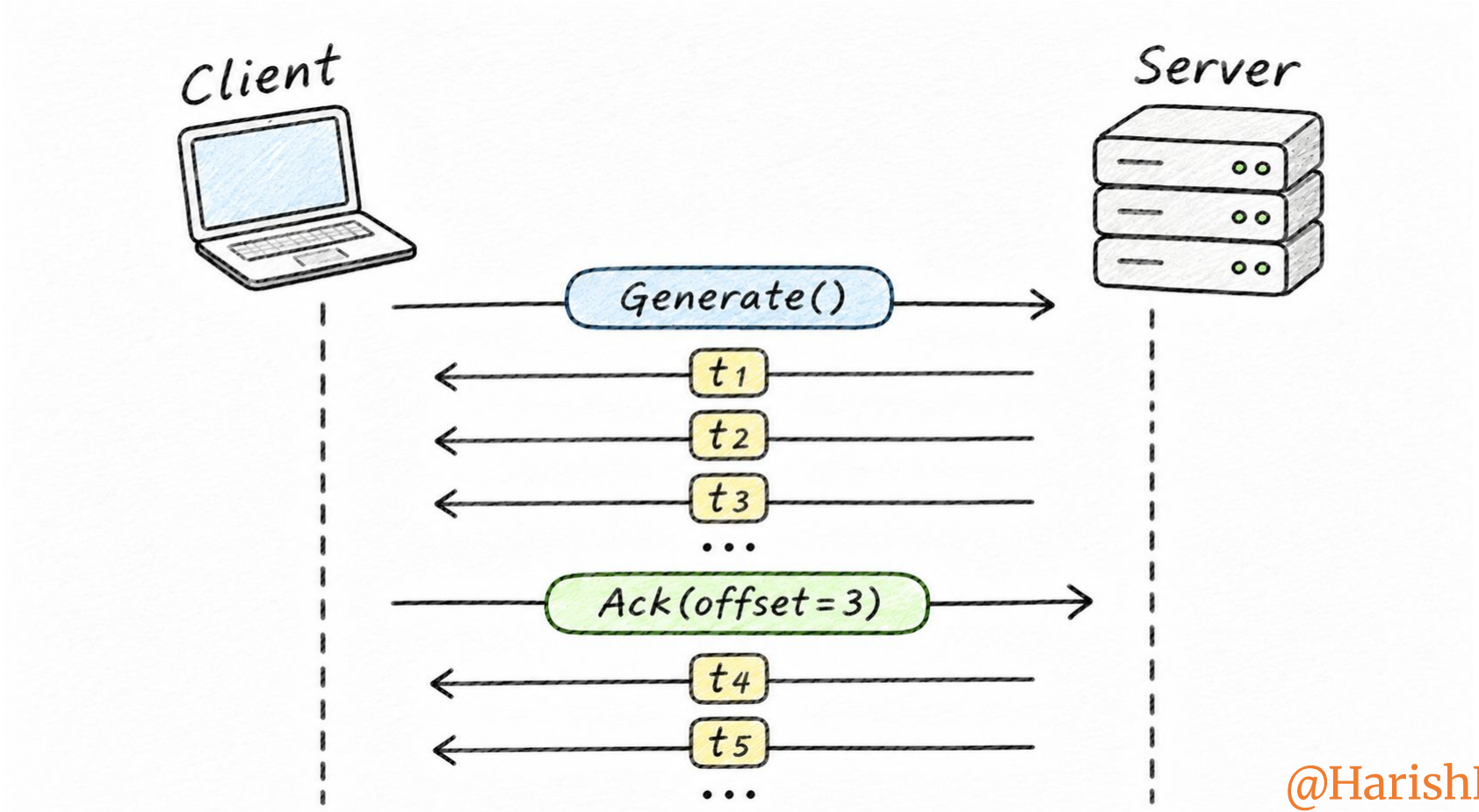
Designing the protocol

gRPConf INDIA 2026

```
01 rpc Generate(GenerateRequest) returns (stream GenerateResponse);
02 rpc Ack(AckRequest) returns (AckResponse);
03 rpc Cancel(CancelRequest) returns (CancelResponse);
04
05 GenerateRequest  { string prompt; string session_id; int32 resume_from_token; }
06 GenerateResponse { string token; int32 token_index; bool is_final; bool replayed; }
07 AckRequest       { string session_id; int32 acked_token_index; }
```

session_id names the generation

The real bug: who owns the generation?



The real bug: who owns the generation?

gRPConf INDIA 2026

two writers, one journal

client A → 25 → 176

client B → 176 → 185

gap → 176 ⇢ 185

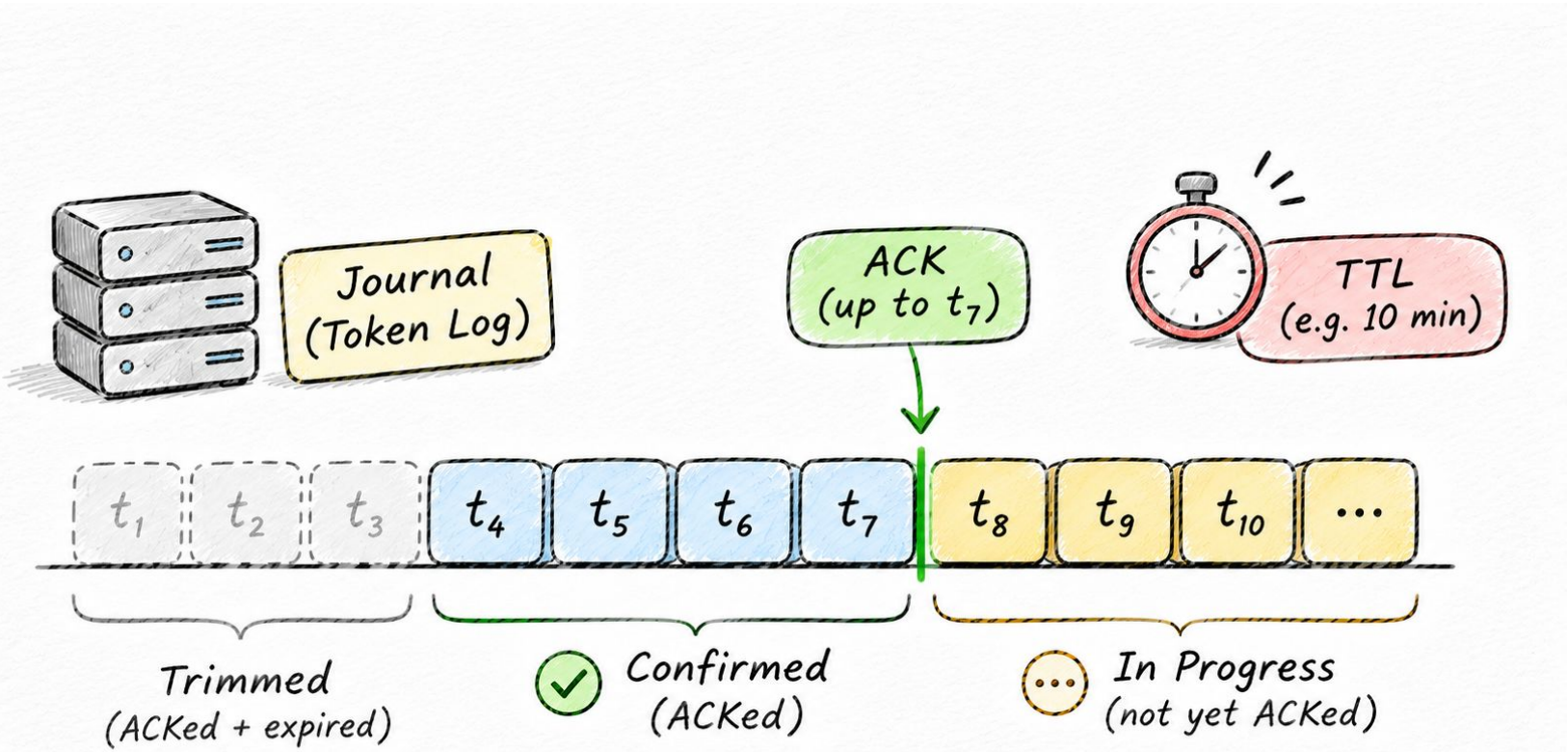
one owner per session

browser disconnected (cancelling 1 call(s); generation continues)

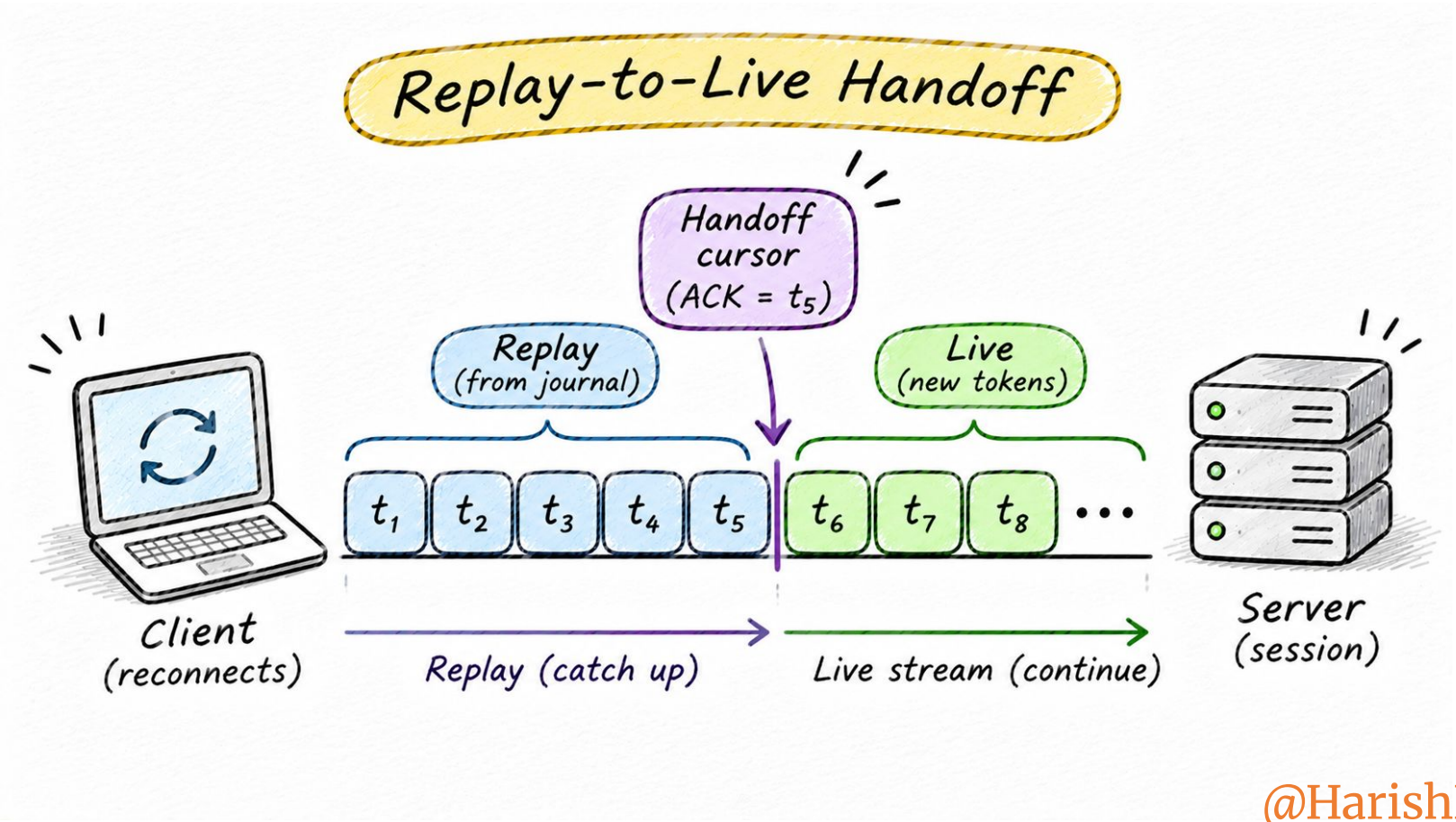
detached (0 still attached)

A dropped client is not a cancelled generation.

The journal: ACKs, retention and TTL



The dangerous boundary: replay versus live



The dangerous boundary: replay versus live

gRPConf INDIA 2026

journal replay

the answer continues exactly as it was

exact · no model

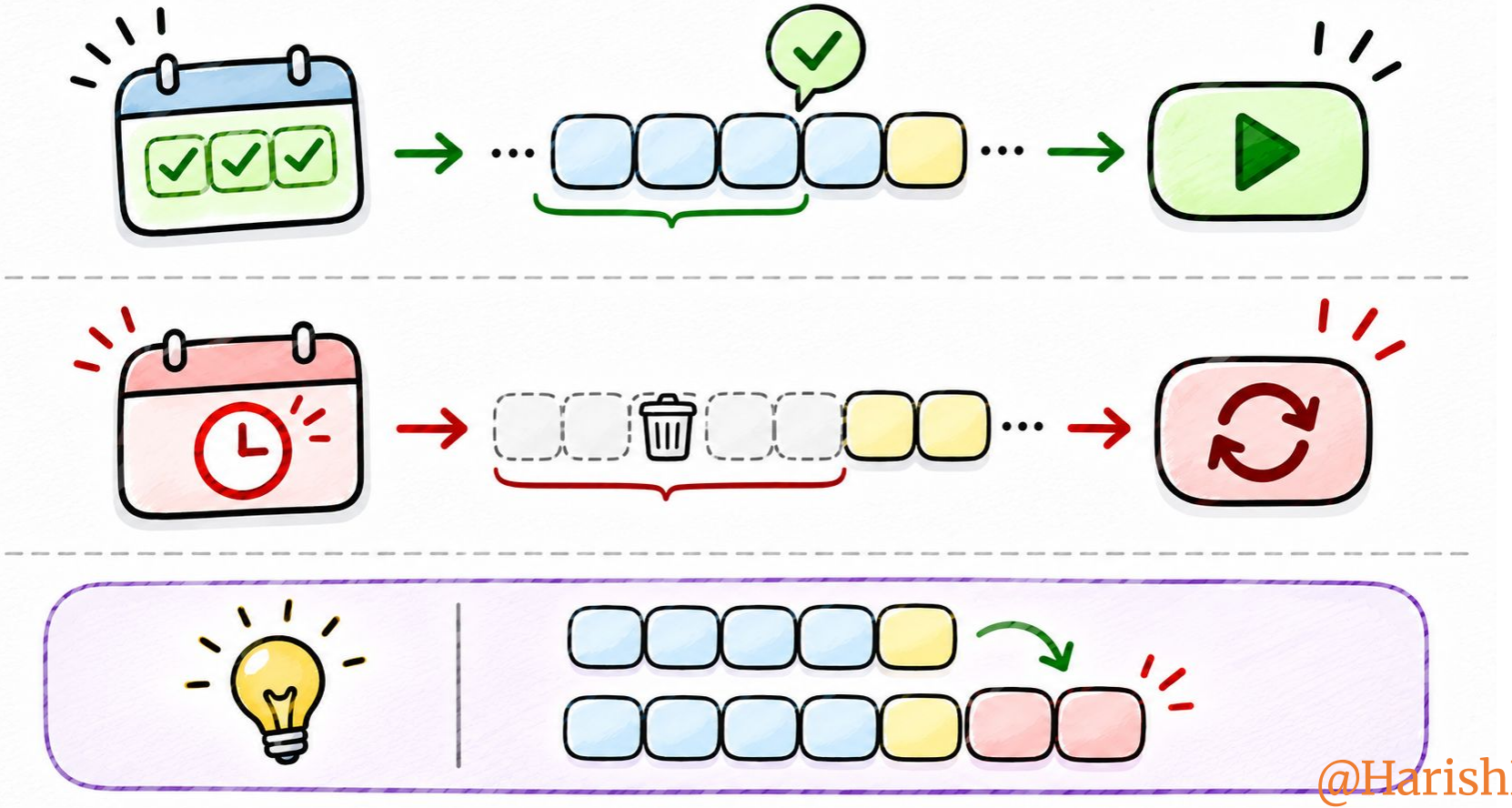
continue after restart

the answer continues as a new sample

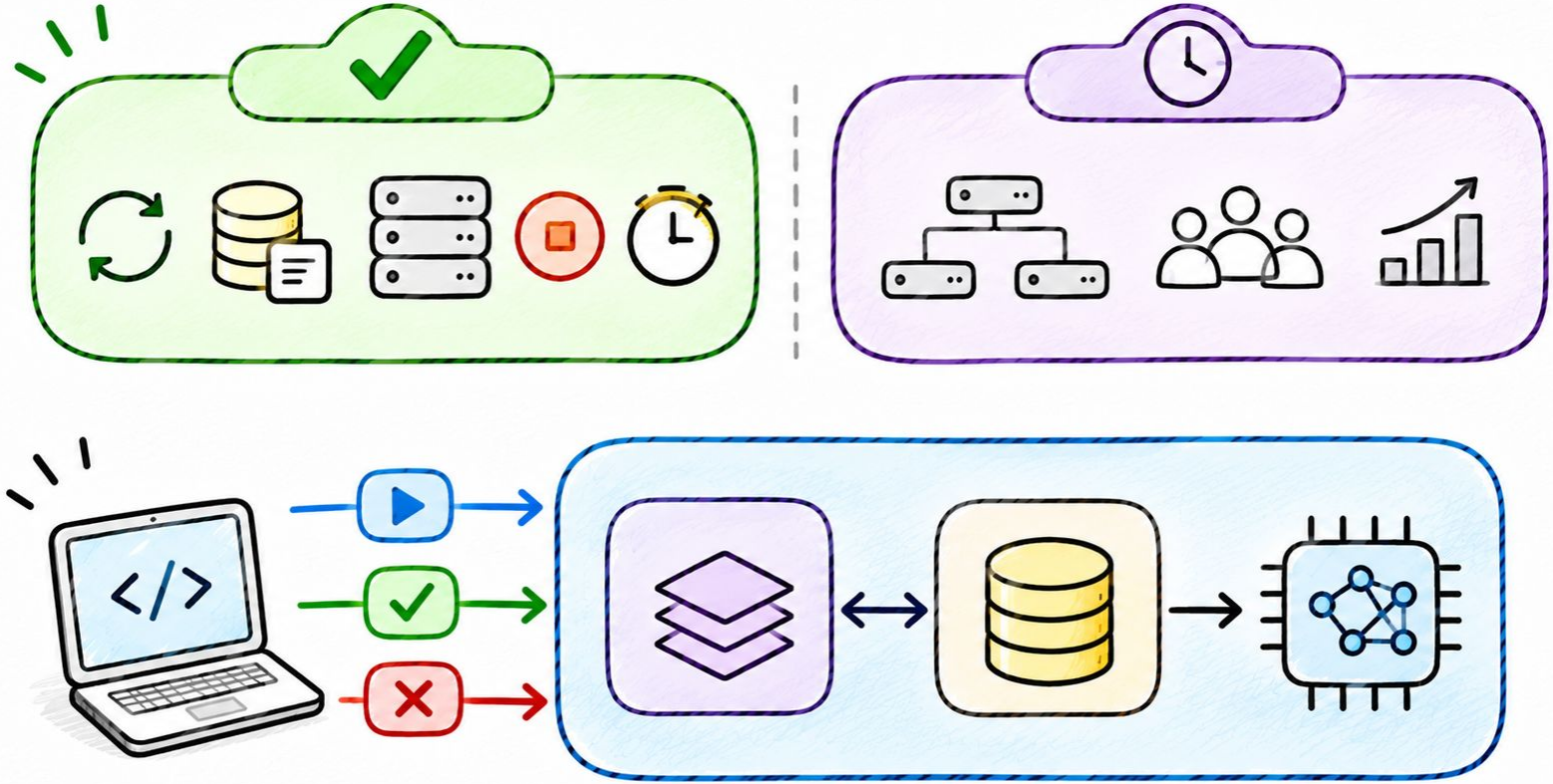
new sample · unseeded

@HarishKotra

When resumption cannot be guaranteed



What this implementation solves and what it doesn't



Kill the stream. Resume the generation.



What We Built

- Cursor-based resumption
- Token journal & ACKs
- Generation independent of connection
- Recovery without restarting inference



Thank you



@HarishKotra