Resilience by Default

Operating non-deterministic agents on infrastructure that has to hold

Ugur Basak Director of Engineering, Platform Engineering & Developer Experience, TomTom

RBLN Europe 2026 Main Stage, Ballroom I / II

WHERE THIS COMES FROM

Two decades keeping the SDLC standing under load

VERIPARK

2007 to 2016

Software architect for Tier-1 banks.

- Secure, multi-tenant digital banking across EMEA
- Regulated and audited by default
- Controls were the product, not an afterthought

BOOKING.COM

2016 to 2022

Senior EM, Developer Experience & Infrastructure. 35+ engineers.

- Global SDLC standardization for 2,500+ engineers
- Payments and Infrastructure Security teams
- Millions of transactions, 24/7, cannot go down

TOMTOM

2022 to now

Director of Engineering. 120+ engineers.

- Internal Developer Platform for 1,500+ developers
- 20+ fragmented tools consolidated into one platform
- 1B+ API calls per day behind a governed gateway

The through-line is resilience: containment, paved roads, safe recovery.

3 A.M.

Your on-call agent decides to roll back

An autonomous agent is on your incident rotation. It sees error rates climb, reaches for the runbook tool, and rolls back the deploy that is currently holding production up.

It has the credentials. It is not even wrong that a rollback often helps. It is wrong right now, and nobody woke up.

Agents plan

Agents call tools

Agents hold credentials

Agents act

The question is not whether the agent is smart. It is what it was allowed to do. **Can we let this run in production?**

Agents don’t need new principles.

They raise the stakes on the ones we already have.

The resilience patterns we built for pipelines and services still apply. Two things change: the blast radius is bigger, and a handful of failure modes are genuinely new. First, the load-bearing half: not new.

NOTHING HERE IS NEW

You have solved this before

SOFTWARE DELIVERY, FIFTEEN YEARS OF IT	AGENTIC SYSTEMS, NOW
Flaky test, non-deterministic build	Same prompt, different action
Leaked production credential	Over-scoped agent token
Fork bomb / runaway job in CI	Runaway loop, agent calling agent
Unreviewed migration shipped to prod	Unreviewed tool call in prod
Config drift, no known-good baseline	No intended state to reconcile against

Same failures. Same fixes. The one real difference is **scale**: an agent runs the loop thousands of times an hour, with real credentials, unattended.

01

Three patterns that kept the SDLC standing

- **Blast-radius containment** Draw the failure domain before the failure.
- **Golden paths** Make the safe way the only way in.
- **Idempotent recovery** Recover by restoring state, not by re-running the actor.

PATTERN 1

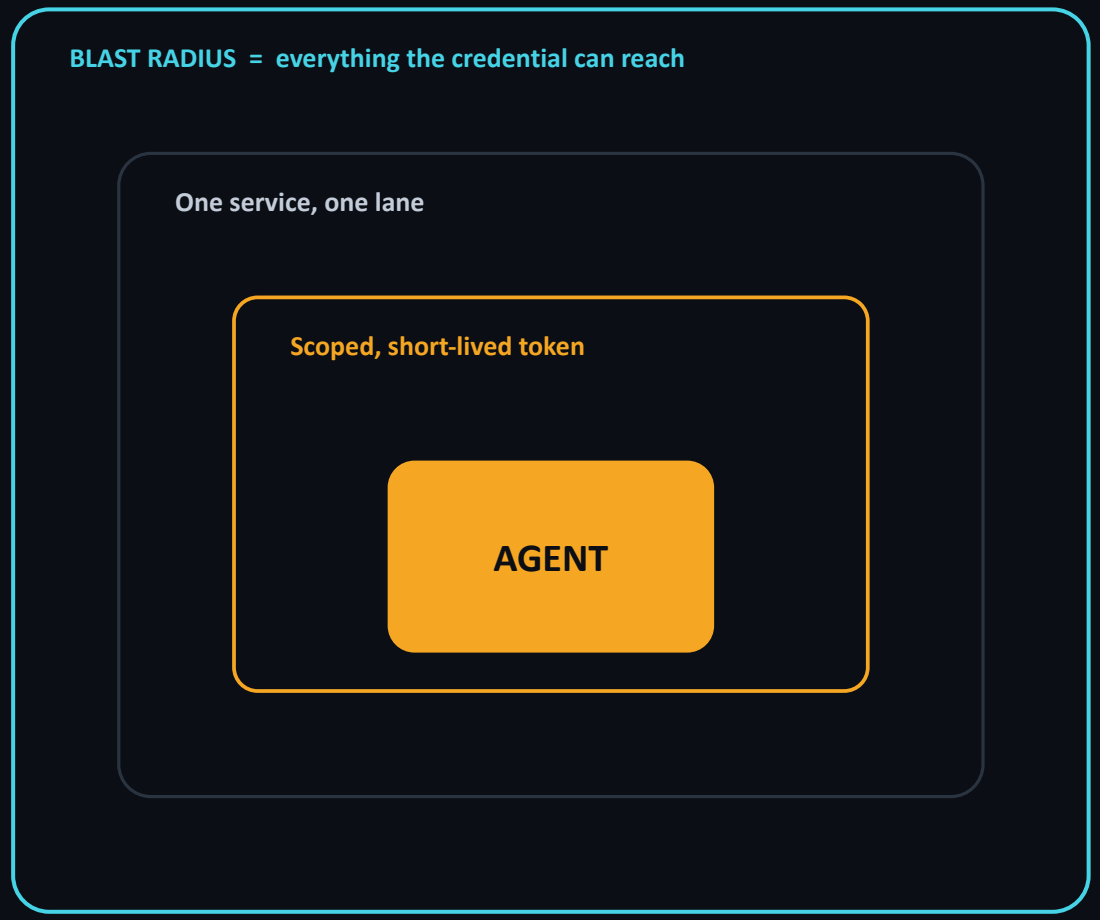
Blast-radius containment

THEN

1B+ API calls per day behind a governed Gloo / Envoy gateway. A bad partner integration could not reach past its own lane. In Payments, failure domains were drawn so one service could not take down settlement.

NOW (for agents)

The agent’s blast radius is everything its credentials can reach. One narrowly scoped, short-lived token per task. No standing access to prod. Assume the agent is compromised and ask what it can still touch.

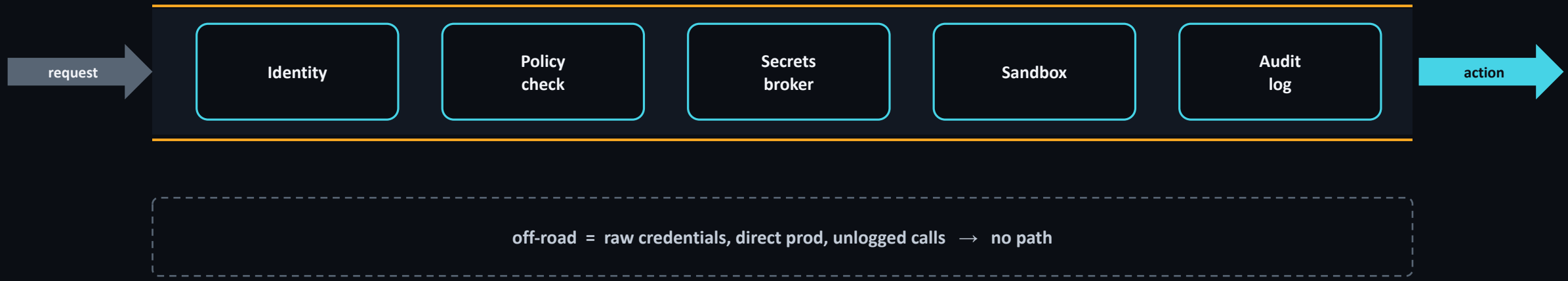


PATTERN 2

Golden paths

THEN 20+ fragmented tools into one platform. 1,500+ developers on paved roads with security, observability and compliance built in by default. The right way was the easy way.

NOW The agent takes the same road. Every gate on it is a deterministic mechanism: identity, policy, secrets broker, sandbox, audit. The agent's reasoning is non-deterministic. The mechanisms it passes through are not.



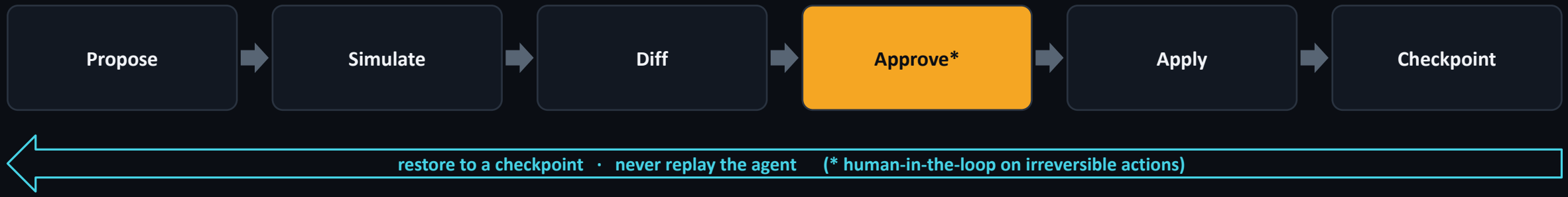
Controls are the intent. Mechanisms are how the platform enforces them, the same way every time.

PATTERN 3

Idempotent recovery

THEN Build and test cycle 2 hours to 10 minutes. We could re-run anything safely. Deploys were replayable. Recovery meant “run it again.”

NOW Replay is not safe when the actor is non-deterministic. Lean on a combination of mechanisms, not on re-running the agent: transactional tool calls, dry-run then diff then apply, checkpoints, a human-in-the-loop gate on the irreversible.



Tier the human. Gate what you cannot undo (human-in-the-loop). Supervise what you can (human-on-the-loop). Automate what is cheap to reverse. Gate everything and the human is the bottleneck.

THE NEW SURFACE

Where the failure modes are genuinely new

INPUT-DRIVEN

Prompt injection is control-flow hijack : untrusted input steers the planner, not just the output

Confused deputy at scale : the agent acts with more access than the person who asked

SCALE-DRIVEN

Non-determinism breaks safe retry : the same trigger can produce a different, worse action

Cascading autonomy : one agent calls another, loops fan out, your own orchestration becomes the DoS

No clean intended state : nothing deterministic to reconcile the result against

You will not enumerate all of these before you ship. That is the point.

02

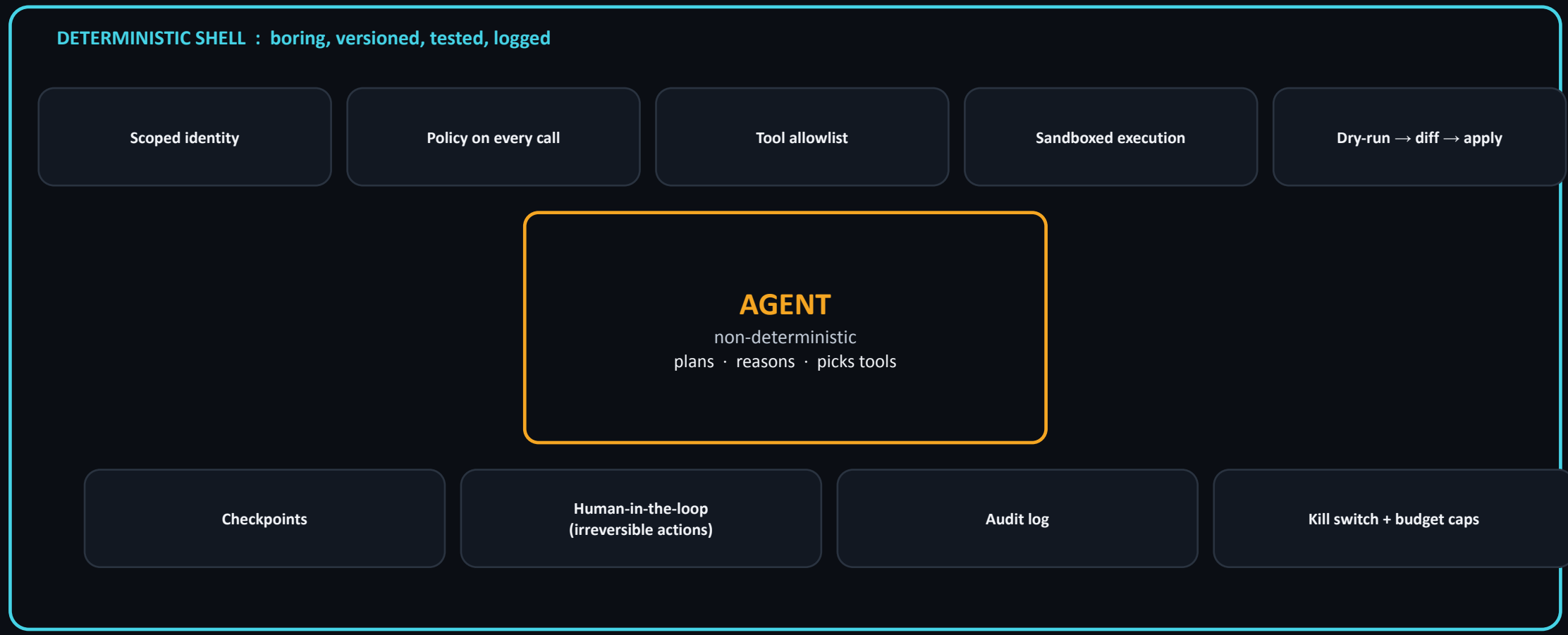
So what do you actually build?

You do not make the model deterministic.

You wrap the non-deterministic core in a Deterministic Shell.

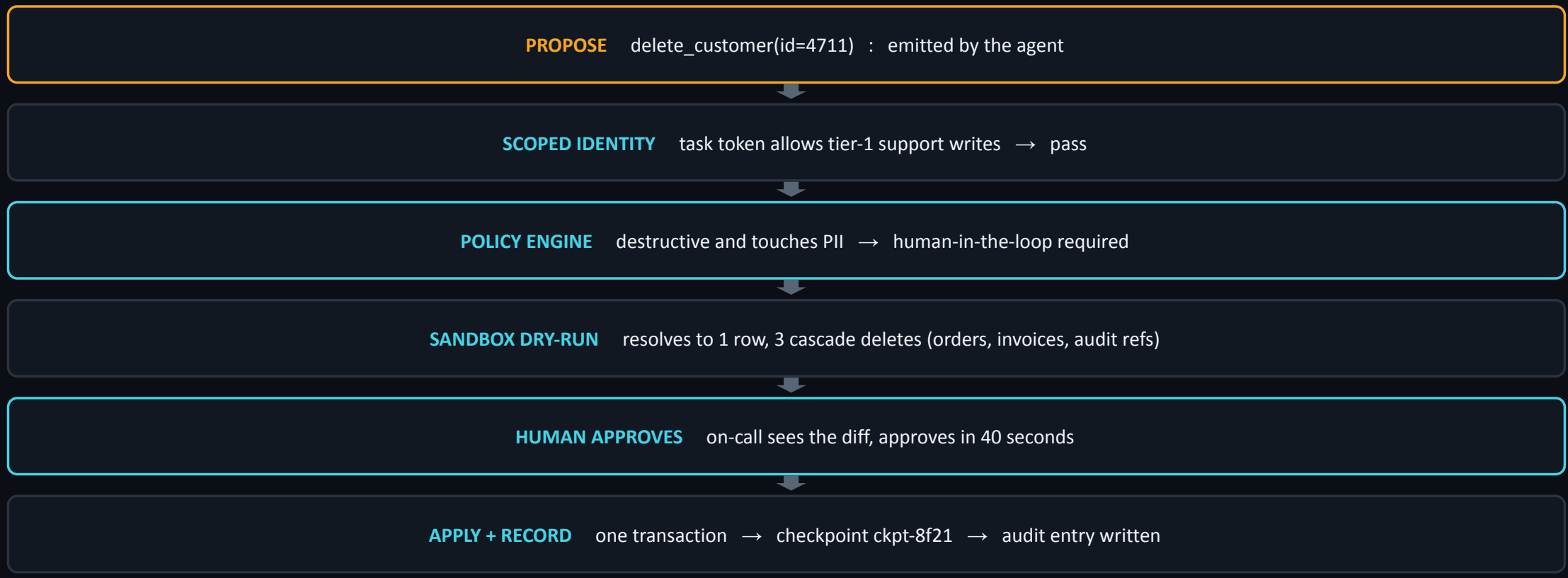
THE CORE IDEA

The Deterministic Shell



Let the core be creative. Make the shell dull and total.

One tool call through the shell



The agent proposed. The shell decided how it happened, and left a record.

THE OLD DISCIPLINE

This is just SDLC discipline

Version
prompts and tools in source control

Test
evals in CI, block the merge

Observe
trace and cost per run

Gate
policy as code on every action

Budget
hard spend and rate caps

Govern
an auditor can read the guardrails

AI is a workload. Not a miracle.

If your SDLC is already mature, you are most of the way there. If it is not, the agent will find every gap.

THE OPEN QUESTION

Will it hold?

- The Deterministic Shell constrains the thing that made agents useful. How far can you box the core before the value is gone?
- Policy engines assume you can enumerate bad actions. Can you?
- We are betting a composition of safe steps stays safe. It might not, and we do not have a good theory for when it breaks.

I would ship this. I would not call it done.

IF YOU TAKE NOTHING ELSE

Three things to take back

1 **Contain the credential.** The blast radius is everything the token can reach.

2 **Make the safe path the only path.** Every gate a deterministic mechanism. No off-road.

3 **Recover to a state, not a prompt.** Checkpoints and transactions, never replay the agent.

Remember the agent that rolled back your deploy at 3am.

With a human-in-the-loop gate on irreversible actions, that call pauses for sign-off and pages you. It does not page itself.

Same question for your stack: would your agent platform pass the review you already run for your payment system?

Thank you

Ugur Basak

Director of Engineering, Platform Engineering & DX, TomTom

DM me on this. I am still working out where the shell breaks.

Slides and past talks are on the site. I am here all day. Come find me and argue with me.

LinkedIn · DM me

ugurbasak.com/speaking