THE LINUX FOUNDATION

OPEN SOURCE SUMMIT INDIA

Agentic Delivery, Guardrailed

AI in CI/CD & Platform Engineering for Production-Safe Delivery Intelligence

Closed-loop delivery intelligence: recommend → assist → automate — only within guardrails.

#OSSUMMMIT

THE LINUX FOUNDATION

About Us

Kalyan Kolachala

Currently India MD at SAI Group, a global enterprise AI leader

Worked previously at Intuit and Hitachi Vantara as India site head for development platform, AI/ML, SaaS and Hybrid Cloud products.

Over 3 decades of experience comprising of engineering leadership, product strategy, site leadership, SaaS, Cloud, design and architecture, AI/ML and GenAI.

Active in open source, contributed to C++ standard and SOA standards with W3C/OASIS.

Speaker in global events on technology and strategy

Graduate in Computer Science from Indian Institute of technology, Kharagpur

SymphonyAI

SAIGroup

Building the leaders in Enterprise AI solutions for the largest vertical markets, including financial services, retail, manufacturing, life sciences and healthcare. SAIGroup India is the center of innovation and excellence with over 1000 world class talent.

Backed by $1 billion of flexible, committed equity capital from Dr. Romesh Wadhwani, a highly successful AI and software entrepreneur and philanthropist (Wadhwani foundation).

Portfolio Companies:
SymphonyAI: Enterprise AI leader in financial services, retail, CPG, manufacturing
ConcertAI: Leader in diagnostics, clinical trials and AI solutions for life sciences and healthcare
GetWell RhythmX: Get Well, a leader in patient engagement software, and RhythmX AI, a leader in AI-powered precision care, combined to form GW RhythmX to usher in the next generation of precision care.
JazzX AI: JazzX is defining the future of enterprise work—by building AI-native digital workers that actually get the job done.

About Us

Manas Ray

Distinguished technology leader with experience building enterprise-grade platforms powered by AI, cloud, and modern software architectures.

He is leading the architecture and AI initiatives across SAIGroup companies. He has led technology initiatives across telecom, retail, banking, healthcare, and cybersecurity, with prior leadership roles at Walmart Labs, Zeta, and Hitachi Vantara.

Manas brings more than 2 decades of deep expertise in architecture, product engineering, open-source advocacy, and scaling high-performing technology products.

JazzX AI

SAIGroup

JazzXAI is an enterprise agentic AI platform built to deliver *Enterprise General Intelligence (EGI)*—autonomous, reasoning-driven systems that operate reliably within the boundaries enterprises. Unlike general-purpose AI optimized for open-ended tasks, EGI is intelligence scoped to the enterprise: grounded in domain knowledge, accountable to policy, and trustworthy enough to run mission-critical workflows.

The vision of JazzX is to enable organizations to reengineer the knowledge-intensive work processes performed by experts with domain-specific expertise today, using domain general AI digital assistants to create "super workers" with a much higher degree of autonomy productivity.

AGENDA

How the session runs

Five movements: the problem, the architecture, the patterns, the guardrails, and how to ship it.

01

The problem & the reframe

Why “chat with logs” isn’t enough — and treating the platform as a product.

02

The reference architecture

The closed delivery-intelligence loop, one change end-to-end, components by plane, mapped to OSS.

03

Four practical patterns

Triage copilot (+ runbook lifecycle), flaky-test detection, test prioritization, change-risk scoring.

04

Making it production-safe

Earned autonomy, the five non-negotiable guardrails, and AI as a testable dependency.

05

Prove it & take it home

Metrics that matter, anti-patterns — then Q&A.

The Gap

Most teams don’t need more “chat with logs.”

They need a platform that turns delivery signals into reliable, auditable outcomes.

! The demo that impresses

- “Ask the AI why the build failed.”
- One-off chat, no memory of the system
- Plausible answers, no evidence trail
- Nothing changes in the pipeline

✓ The platform that ships

- Triage grounded in change logs + metrics + events + runbooks
- Connected to Git, CI, CD, telemetry
- Every claim links to its source
- Acts only inside policy guardrails

The Reframe

Treat the platform as a product

A unified “delivery intelligence” layer — not a pile of scripts and a chatbot.

Has users

Developers & on-call engineers are customers with workflows, not ticket-fillers.

Has contracts

Inputs (signals) and outputs (recommendations) are versioned and testable.

Has a roadmap

Capabilities ship incrementally: recommend first, automate last.

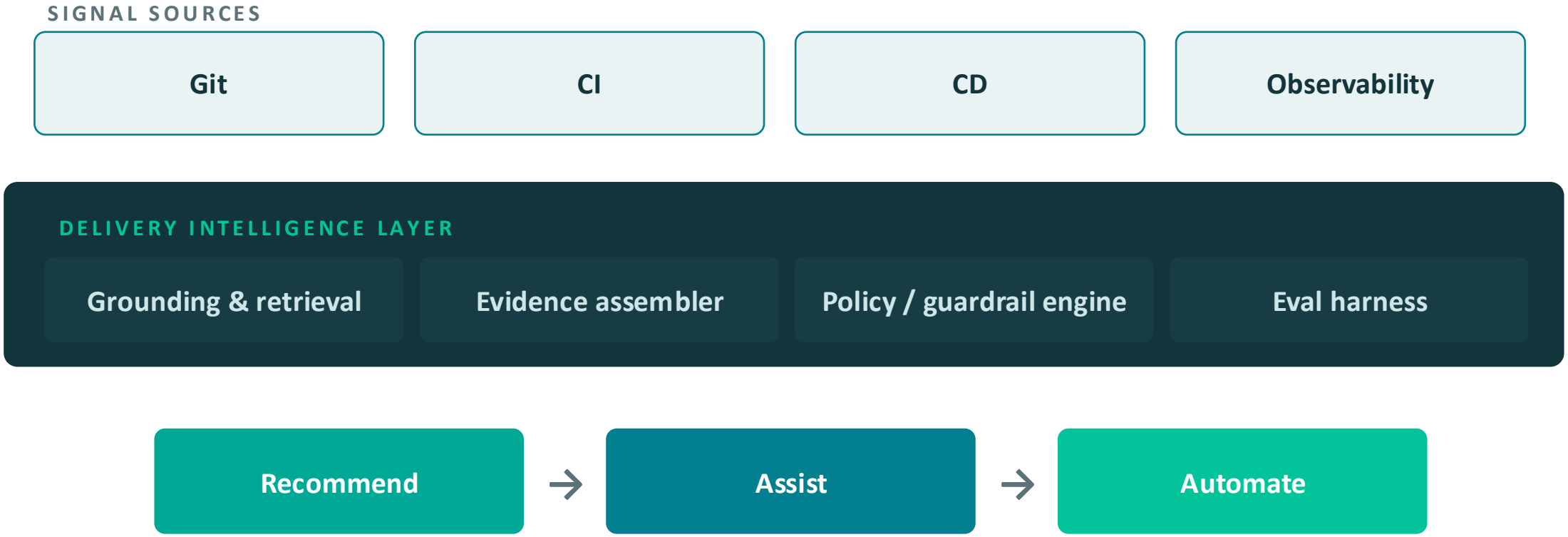
Has SLOs

Trust is measured — false-red rate, MTTR, adoption — like any product.

Reference Architecture

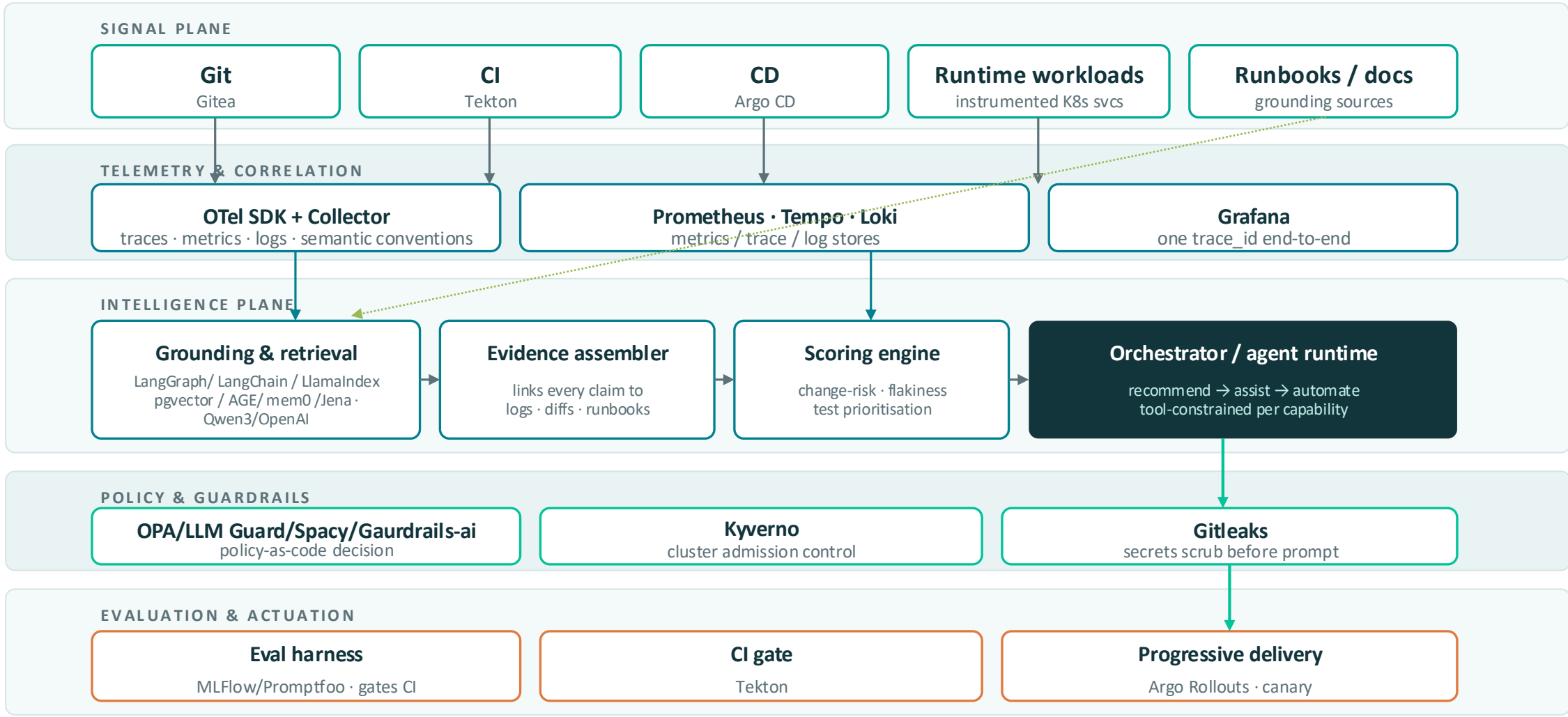
The delivery intelligence layer

A closed loop over your existing tools — OpenTelemetry is the connective tissue.



Our Architecture backed by OSS

Components by plane



OSS Building Blocks

Same loop, mapped to OSS

Nothing here is mandatory — pick one per layer, stay on tools you can self-host and audit.

Signals — Git / CI / CD	Git + Gitea / Forgejo	Tekton · Argo Workflows · github-actions	Argo CD · Flux
Telemetry & correlation	OpenTelemetry SDK + Collector	Mimir · Tempo · Loki · ClickHouse	Grafana (one trace ID end-to-end)
Grounding & retrieval	LangGraph/LangChain /LlamaIndex/agno/crew	pgvector/qdrant · AGE/Kuzu/JanusGraph · mem0/Zep · ArcadeDB	vLLM (Qwen3) · Frontier Models (OpenAI/Claude)
Policy & guardrails	OPA / Conftest (policy-as-code)/LLM Guard · NeMo Guard · Guardrails ai · Llama Gaurd	Kyverno · Gatekeeper	Secrets scrub: Gitleaks / TruffleHog
Evaluation & orchestration	Promptfoo · DeepEval · MLFlow	Run in existing CI (GitHub Actions, Jenkins)	Flagger / Argo Rollouts (canary)

OpenTelemetry is the spine — one correlation key (e.g. a project / trace ID) threads Git → CI → CD → runtime.

Telemetry - SemConv

Instrumenting CI/CD: spans, metrics, resources

OTel GenAI/CICD semantic conventions — the signals the delivery-intelligence loop reads.

1 · Pipeline Run span SERVER

Name `{action} {pipeline}`
Required `cicd.pipeline.result`
success · failure · timeout · skip · cancellation · error
Cond. `error.type` on failure (low-cardinality)
+ `name` · `run.id` · `run.url` · `run.state`

2 · Task Run span INTERNAL · child

Required
`task.name` e.g. Run Go Linter
`task.run.id` · `task.run.result` · `task.run.url`
`error.type` conditional on failure
Optional `task.type` build/test/deploy

EXAMPLE TRACE

```
Trace: cicd-pipeline-run-abc123
● SERVER BUILD My Service
    result=success
  └ INTERNAL Lint Code
        success · test
  └ INTERNAL Security Scan
        guardrail · Trivy SBOM
  └ INTERNAL Build & Test
        failure · error.type=compile
  └ INTERNAL Deploy Staging
        skip
```

METRICS *per-pipeline; run-id opt-in*

`pipeline.run.duration` Histogram
name · state · result · error.type

`pipeline.run.active` UpDownCounter
name · state

`worker.count` UpDownCounter
worker.state: available/busy/offline

`pipeline.run.errors` Counter
name · error.type

`system.errors` Counter
component: controller/scheduler

Resources cicd.pipeline · .run · .worker (worker.id required)

VCS vcs.change.count · .duration by open/merged/closed

Logs *placeholder — less mature than spans/metrics*

Choosing OSS Gaurdrails

Open-source AI guardrails, compared

All free/OSS — “cost” = self-host compute, not licensing. They’re complementary; most teams layer them.

Tool	Approach	Cost	Latency	Accuracy	Best for
NeMo Guardrails	Colang state-machine, conversational rails	Higher	Higher	Med	Multi-turn chatbots & agents; strict flow / topic control
Guardrails AI	Composable I/O validators + repair (Python Hub)	Med	Low	Var	Python apps: output quality, structured data, PII, RAG
OPA	Policy-as-code (Rego), infra-level enforcement	Low	Very low	High*	Enterprise governance, agent tool-use, K8s/API policies
Llama Guard	Open-weight safety classifier (7-12B)	Med	50-500ms	High	Content-safety moderation layer (pre/post LLM)
LLM-Guard	Lightweight scanner library (~15-20 scanners)	Low	Very low	Var	Fast input/output scanning toolbox (PII, injection)

Defense-in-depth (production): fast scanner (LLM-Guard / OPA rules) → safety classifier (Llama Guard) → app validation/flows (Guardrails AI /NeMo) → infra policy (OPA) → observability.

The Loop, made concrete

One change, walked end to end

PR #2417 bumps a timeout in the document processing service. Watch it travel the loop.

GIT	CI	INTELLIGENCE	CD	OBSERVE
1	2	3	4	5
PR #2417 opened	**Build + tests run**	**Score + ground**	**Canary rollout**	**Watch + feed back**
Diff + change history captured; trace_id stamped on the commit.	Triage copilot stands by; test selection runs document management suite first.	Change-risk = MEDIUM. Evidence assembled, policy checked, evals green.	Policy maps MEDIUM → 10% → 50% → 100% with auto-rollback armed.	OTel spans on same trace_id; p99 + error rate become new signals.

↺ runtime signals (p99, errors) flow back as new inputs — next change is scored with this deploy's outcome

FOUR PRACTICAL PATTERNS

From signal to safe action

Triage copilot · Flaky-test detection · Test prioritization · Change-risk scoring

Pipeline failure triage copilot

RECOMMEND

Grounded in CI/CD logs + runbooks. No claim without a link.

HOW IT WORKS

1. Pipeline fails → logs, diff, and recent deploys are pulled as context
2. Retrieve matching runbook steps + similar past failures
3. Model proposes likely cause — each statement cites its source
4. Posts to the PR / channel as a suggestion, never an auto-fix

SAMPLE OUTPUT (annotated)

Likely cause

DB migration timed out — connection pool exhausted

Evidence

- job log L412–418 · runbook: db-migrate#retry
- 3 similar failures in last 14 days

Suggested next step

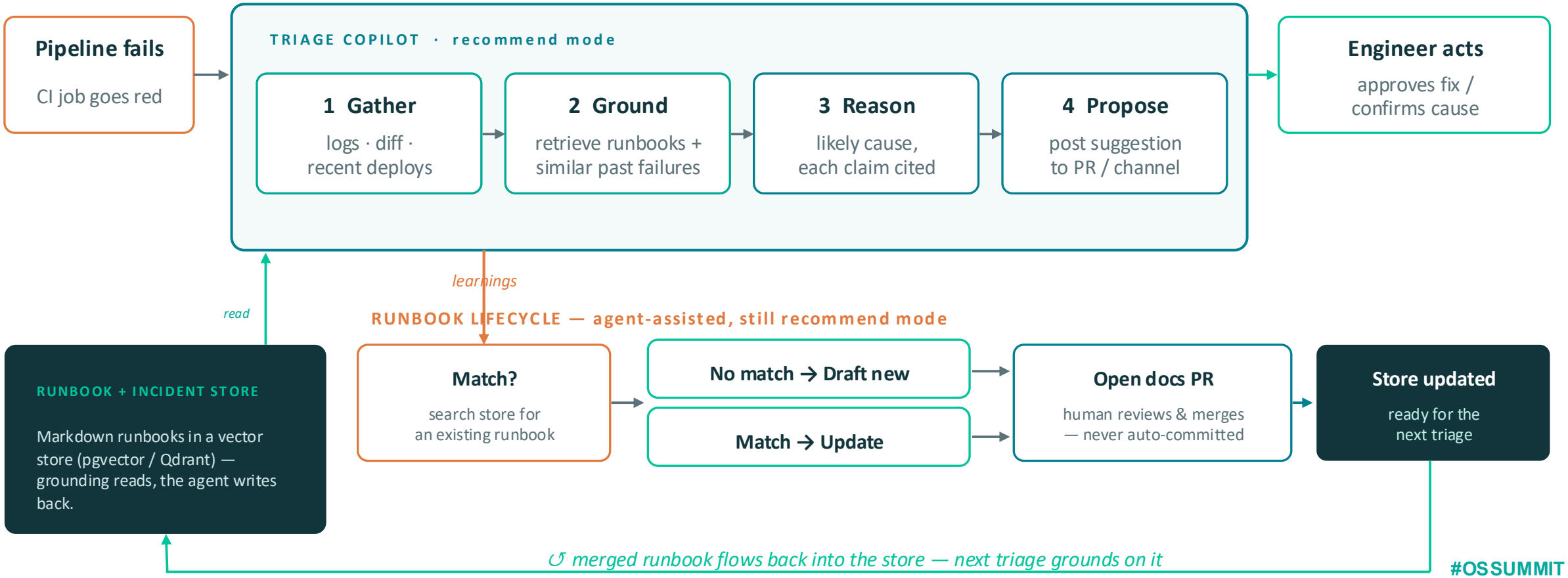
Re-run with pool size 20; owner: @platform

⚠ Recommendation only — human approves

Pattern 1 – Deep dive

Triage flow — and the runbook it leaves behind

Every triage either uses a runbook or writes one — so the next failure is faster to resolve.



ASSIST

Flaky-test detection

Separate real failures from noise — protect trust in the red signal.

HOW IT WORKS

1. Track pass/fail history per test across retries and branches
2. Score flakiness: fails then passes with no code change
3. Quarantine high-flake tests; open a ticket with the evidence
4. Surface flake rate as a platform health metric

SIGNAL, NOT VIBES

Why it matters

A flaky suite trains engineers to ignore red.

What the platform tracks

- flake_score = flips / runs (windowed)
- quarantine if score > threshold

Open-source friendly

Built on test history you already emit — JUnit XML, OTel test spans.

Pattern 3/4

ASSIST

Test prioritization & selection

Shorten the feedback loop — run what the change actually touches.

HOW IT WORKS

1. Map code paths → tests using coverage + change history
2. On a PR, rank tests by probability of catching this change
3. Run the high-value subset first for fast early signal
4. Full suite still runs — selection reorders, never skips silently

THE TRADE-OFF, MADE EXPLICIT

Goal

Cut feedback time without losing coverage confidence.

Guardrail

Selection affects ORDER + speed, not whether a test is allowed to gate merge.

Measured by

- time-to-first-failure · escaped defects

Change-risk scoring

AUTOMATE

Drive progressive delivery — risky changes earn a slower rollout.

HOW IT WORKS

1. Score each change: size, blast radius, author/area history, test signal
2. Low risk → standard rollout. High risk → canary + tighter watch
3. Wire the score into the CD policy, not a dashboard nobody reads
4. Auto-rollback thresholds set per risk tier

RISK → ROLLOUT POLICY

Low

- deploy 100% · standard monitors

Medium

- canary 10% → 50% → 100%

High

- canary + manual gate + fast rollback

This is where recommend becomes automate — safely.

Earned Autonomy

Autonomy is earned, not granted

Each capability climbs the ladder only after it proves itself at the rung below.

Recommend

AI proposes; human decides. Every output carries evidence.

Promote when: accuracy + adoption hold

Assist

AI prepares the action; human approves with one click.

Promote when: false-positive rate is low & stable

Automate

AI acts inside policy; human is notified, can override.

Stays here only while guardrails + evals stay green

Production-Safe by Design

Five guardrails, non-negotiable

✓ Evidence-first

No claim without a link. Outputs cite logs, diffs, runbooks — or they don't ship.

§ Policy-as-code

Boundaries live in version control (e.g. OPA). The AI operates inside them, never edits them.

* Secrets redaction

Context is scrubbed before it reaches a model. Tokens and keys never enter a prompt.

Prompt-injection defense

Treat logs & PRs as untrusted input. Constrain tools, validate every action.

≡ Evaluation harness

AI behavior is tested like any dependency — fixtures, regressions, CI gates.

The point

If you can't test it and audit it, it doesn't belong in your pipeline.

Make it Testable

AI behavior is just another dependency

Pin it, test it, gate it. The same discipline you apply to any library.

1

Fixtures

Curate real failures with known-good triage answers as golden cases.

2

Assertions

Did it cite the right evidence? Stay inside policy? Avoid hallucinated fixes?

3

Regression gate

Run evals in CI on every prompt / model change — block merges that regress.

4

Drift watch

Track quality in production; alert when accuracy or grounding slips.

Open-source building blocks: your existing CI runner + a thin eval suite — no new platform required.

Prove the Impact

Four metrics, before and after

Illustrative targets — instrument your own baseline first, then show movement.

Cycle time

Faster idea → prod via test selection & smart rollout

False-red rate

Flaky detection restores trust in the red signal

MTTR

Triage copilot cuts time-to-cause on failures

Change failure rate

Risk scoring + progressive delivery catch bad changes early

These map to DORA — leadership already knows how to read them.

From Signals to Outcomes

Computing DORA from OTel CI/CD signals

The correlation key joins VCS change → CD deploy → runtime — that join is what makes DORA computable.

Deployment Frequency

How often you ship to prod.

FROM SIGNALS

Count of `cicd.pipeline.run`
`where task.type=deploy`
`result=success, env=prod`

```
= deploy count / time
```

src: Counter / rate over run spans

Lead Time for Changes

Commit → running in prod.

FROM SIGNALS

`vcs.change` commit ts
`→ prod deploy span`
joined on correlation key

```
= deploy_ts − commit_ts
```

src: Trace join (VCS ↔ CD)

Change Failure Rate

Share of deploys that break prod.

FROM SIGNALS

failed / rolled-back deploys
`cicd.pipeline.run.errors`
(deploy, prod)

```
= failed deploys / all deploys
```

src: Counters, ratio

Failed Deploy Recovery

Time to restore after a bad deploy.

FROM SIGNALS

failed deploy / incident
`→ next success or rollback`
same service, correlated

```
= restore_ts − failure_ts
```

src: Span timestamps

↺ One correlation key (trace_id / project_id) threaded Git → CI → CD → runtime is the prerequisite — *without the join, lead time and change-failure rate can't be derived from CI/CD signals alone.*

Avoid the Traps

Where agentic delivery goes wrong

Auto-fixing in prod on day one

Skipping recommend/assist destroys trust the first time it's wrong.

Outputs with no evidence

An unsourced "root cause" is a guess. Engineers stop reading it fast.

Treating AI as untestable

If you can't regression-test it, you can't safely change the prompt or model.

Feeding raw logs to a model

Secrets leak and injection lands. Scrub and constrain first.

Dashboards no policy reads

A risk score nobody wires into CD is theater. Connect it to the loop.

No baseline metrics

Without before/after you can't prove value — or detect regressions.

Build the loop.
Guard the rails.
Earn the autonomy.

Start with one pattern, grounded in evidence and gated by evals. Ship recommend before automate — and let the metrics make the case.

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT INDIA

Kalyan Kolachala

Managing Director, SymphonyAI Group India, Earlier India site/product head ...

Manas Ray

Distinguished Architect | LLM & Agentic AI | Cloud-...

#OSSUMMMIT

THE LINUX FOUNDATION