RAPTORS CONFERENCE 2026

Intelligent CI/CD at Scale

Autonomous agents for software delivery

Karishma Verma
Engineering Manager

Lightning talk
5 minutes

Pipelines see syntax, not semantics

```
payment-api/schema/order.json
@@ -14,7 +14,6 @@
   "amount": "number",
-  "discount_code": "string",
   "currency": "string"

1 file changed, 1 deletion  →  run every test
```

WHAT IT MISSES

A field was removed. That breaks the contract.

`invoice-svc` still reads it, via the event bus.

No static graph shows that link.

WHERE DEPENDENCIES HIDE

Synchronous calls SEEN

HTTP, gRPC

Usually visible in a static graph

Async messaging HIDDEN

Queues, topics, event buses

Producers can't see consumers

Shared state HIDDEN

Database tables, caches

Coupled through data, not code

Four agents that decide, not just run

1

Change Agent

the detective

What changed, and why?

→ `ChangeClass`

2

Impact Agent

the office gossip

Who depends on it, even through the event bus?

→ `BlastRadius`

3

Verification Agent

the food critic

Which tests actually matter for this change?

→ `TestPlan`

4

Deployment Agent

the cautious parent

Canary or blue-green? How fast, and when to roll back?

→ `RolloutPlan`

Orchestrator Checks policy at every handoff and writes each decision to the Risk Ledger.

Sits on top of your existing CI/CD: webhooks, deploy controllers, OpenTelemetry, service catalog, test reports.

Semantic change classifier

THREE INDEPENDENT SIGNALS

RuleFilter	file path + extension heuristics
ASTAnalyzer	interface, schema, config patterns
MessageParser	sanitized NLP signal from the commit message

SCORING

```
agreement = CountAgreement(signals)
confidence =
  0.90  if 3 agree
  0.75  if 2 agree
  0.55  if 1 agrees
if any_high_risk and conf < 0.70:
  conf = 0.70     # high-risk floor
category = MajorityVote(signals)
is_ambiguous = conf < 0.80
```

OUTPUT · DESIGN TARGETS

`{ category, confidence, risk_hypothesis, is_ambiguous }`

Precision ≥ 0.85 · Recall ≥ 0.80 · High-risk false-negative rate < 5% · Latency p95 < 2 s

Blast-radius inference

SIX EVIDENCE SOURCES

E_1	**Service mesh**	1.0	
E_2	**Distributed tracing**	0.9	30-day window, drop > 90 d
E_3	**Build manifests**	0.8	
E_4	**Shared data stores**	0.6	
E_5	**Message topics**	0.6	
E_6	**Incident correlation**	0.4	365-day window, no infra failures

SCORING

$$\text{consumers} = E_1 \cup \ldots \cup E_6$$

$$\text{confidence} = \frac{\sum w_i \cdot e^{(-\text{age}_i/30)}}{\sum w_i}$$

$$\text{conflicts} = (E_2 \neq E_1)$$

```
if confidence < 0.60: escalate
```

Conflict resolution

runtime > static

recent > stale

declared > inferred

Output: direct = $E_1 \cup E_3$ · runtime_coupled = consumers − direct · confidence · conflicts

Contract test synthesis

PSEUDOCODE

```
if category != SERVICE_INTERFACE:
    return ∅

for s in blast_radius.direct_consumers:
    schema = registry.get(s)
    if schema is null: skip
    for op in schema:
        T⁺ = positive(op)  # → 2xx + schema
        T⁻ = negative(op)  # missing → 400

for t in tests:
    if fails(t, HEAD):   discard t
    if fails(t, HEAD⁻¹): flag for review
```

Scope

Interface changes only. Contract tests only.

Self-check

A test must pass on HEAD. Failing on the previous commit flags it for review.

Quarantine

New tests start quarantined. Promote after 5 runs at > 90% pass rate.

Targets

False-positive rate < 10%. Flakiness < 5%.

A field disappears from an API

```
payment-api   -  "discount_code": "string",
```

1	**Change**	Breaking change to a service interface.	`SERVICE_INTERFACE · 0.90`
2	**Impact**	order-svc calls it directly. invoice-svc reads it off the event bus, which the static map missed.	`conf 0.65 · traces ≠ mesh`
3	**Verification**	Contract tests for those two services only. Everything unrelated is skipped.	`smoke ∪ history ∪ blast radius`
4	**Deployment**	Not sure enough for blue-green. Canary at 5%, auto-rollback, and a human is flagged.	`CANARY · rollback err > 0.5%`

Names and values are examples. Logic follows Algorithms 1 to 6 in the paper.

Autonomy, with limits

RISK	STRATEGY	GATE	WHO DECIDES
Low	Standard deploy	Smoke tests pass	The pipeline
Medium	Canary, 5% first	Auto-rollback if errors rise	The pipeline, carefully
High	**Blue-green**	A human approves the switch	A person, every time

Every decision lands in the **Risk Ledger**: the change, the evidence, the call, and any override.

Override

One click. The reason is logged.

Circuit breaker

Agent errors send everything back to manual approval.

Shadow mode

New capabilities decide, but don't act.

Confidence gate

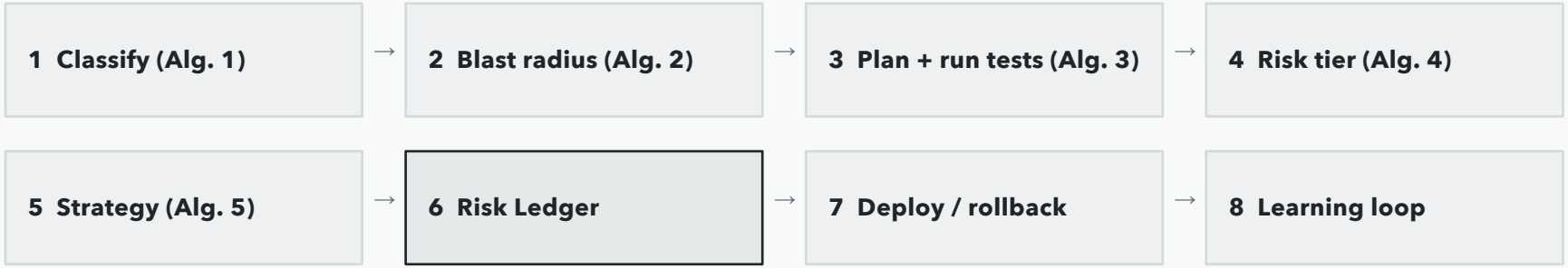
Low confidence goes to a human.

Risk tier and deployment strategy

TIER	WHEN	STRATEGY	GUARDRAILS	ROLLBACK
HIGH	category ∈ {DATA_SCHEMA, SERVICE_INTERFACE, INFRASTRUCTURE_DEFINITION} and blast-radius confidence > 0.70	**BLUE_GREEN**	Human approval + full test suite + expert review	Any production error or operator override
MEDIUM	classification confidence < 0.70, or conflicts, or blast-radius confidence < 0.60. Also the default.	**CANARY**	5% initial exposure. Progress if p99 < 200 ms and errors < 0.1%	Error rate > 0.5% or p99 > 500 ms
LOW	category ∈ {APPLICATION_CODE, CONFIGURATION_FILE, DEPENDENCY_DECLARATION} and blast-radius confidence > 0.80	**STANDARD**	Smoke tests pass	Error rate > 1% or p95 > 500 ms

Rules are evaluated top to bottom. Anything that matches no rule is MEDIUM.

End-to-end orchestration



ESCALATE TO A HUMAN IF ANY:

```
classification.confidence < 0.70
blast_radius.confidence   < 0.60
blast_radius.conflicts    = true
risk_tier = HIGH
synthesized test validation fails
learning-loop policy proposal
```

HARD STOPS AND LEARNING

- Blocking test failure → abort
- Approval not granted → abort
- Rollback trigger → automatic rollback
- Policy changes need human approval and at least 100 samples

Applied in production

21

services my teams own, all on fully automated CI/CD



■ set up by hand □ onboarded themselves

First 2	We set them up by hand.
Then	We automated the onboarding.
Other 19	They onboarded themselves.

Once the pattern worked for two services, *it scaled without adding work for each new one.*

Start small. Expand step by step.

1 Observe

Agents are read-only

Tracing on critical services
Ledger records, no actions
Build the dependency map

2 Shadow

People decide

Log agent vs. human calls
Track agreement over time
Tune policy thresholds

3 Low risk

Agents act

Low tier only
Circuit breaker armed
Canary plus auto-rollback

4 Expand

Step by step

Medium tier, guarded
High risk stays human
Feed outcomes back

Give your pipeline judgment. Give it limits. Make it keep receipts.

THANK YOU

Now go deploy on a Friday.

Karishma Verma
Engineering Manager

READ THE PAPER

proceedings.raptors.dev/verma-intelligent-ci-cd-at

DOI 10.68495/rc.2026.C032