KubeCon CloudNativeCon North America 2026

#KubeCon #CloudNativeCon

Progressive Delivery for Trust

How agents earn the right to merge

Darian DeFalco · Senior Staff SRE · Transfix



Instruments let scientists observe what they couldn't otherwise.

A miscalibrated one still produces results, so calibrating it and noticing drift is vital.

REVIEW BOTTLENECK

Review is the bottleneck



1,642 agent-written PRs in 90 days

Faster to write than to review.

Trust attaches to the work, not the worker.

bound → supervise → correct the recipe

grant

observe

THE LENS · DECISION BY DECISION

Where does each decision sit?

ONE TASK, FIXING A VULNERABLE DEPENDENCY · THE DECISIONS INSIDE IT

CODE DECIDE → JUDGMENT DECIDES

a fixed rule	a rule with judgment inside it	judgment checked by code	open judgment
which recipe runs whether it may merge	how to make the change	what to exercise in the test	none

Separate question: how reliably can we tell it worked?

Another: how costly would a mistake be?

Plugin versioning: code makes the decision

YAML

```yaml
on:
push:
branches: [main]
paths: ['plugins/tf/**']

# decision
if version already moved: skip
else: bump patch,
open a PR
```

Simple decision – no need for inference

README MAINTENANCE · HIGH INFERENCE · MINIMAL RISK

README: merges on its own, audited daily

01

Evals

the generator, before any PR

02

Real proposals

owner review while it earned it

03

Correction

skill corrected; scope narrowed

A daily audit reads every bot merge. A kill switch pauses the lane; a wrong edit is a revert.

diff size | README files only | verifier re-run at the merge commit | merged by the bot

SUPERVISED WORK · ONE TICKET, TWO PULL REQUESTS

An error reveals a condition the recipe missed

First PR

pyproject.toml +2 · poetry.lock

pins a transitive dependency directly

branch ...-fresh · no labels

closed

routing: inferred → scripted

+ 12 tests

Same ticket, next PR

poetry.lock only

follows the recipe

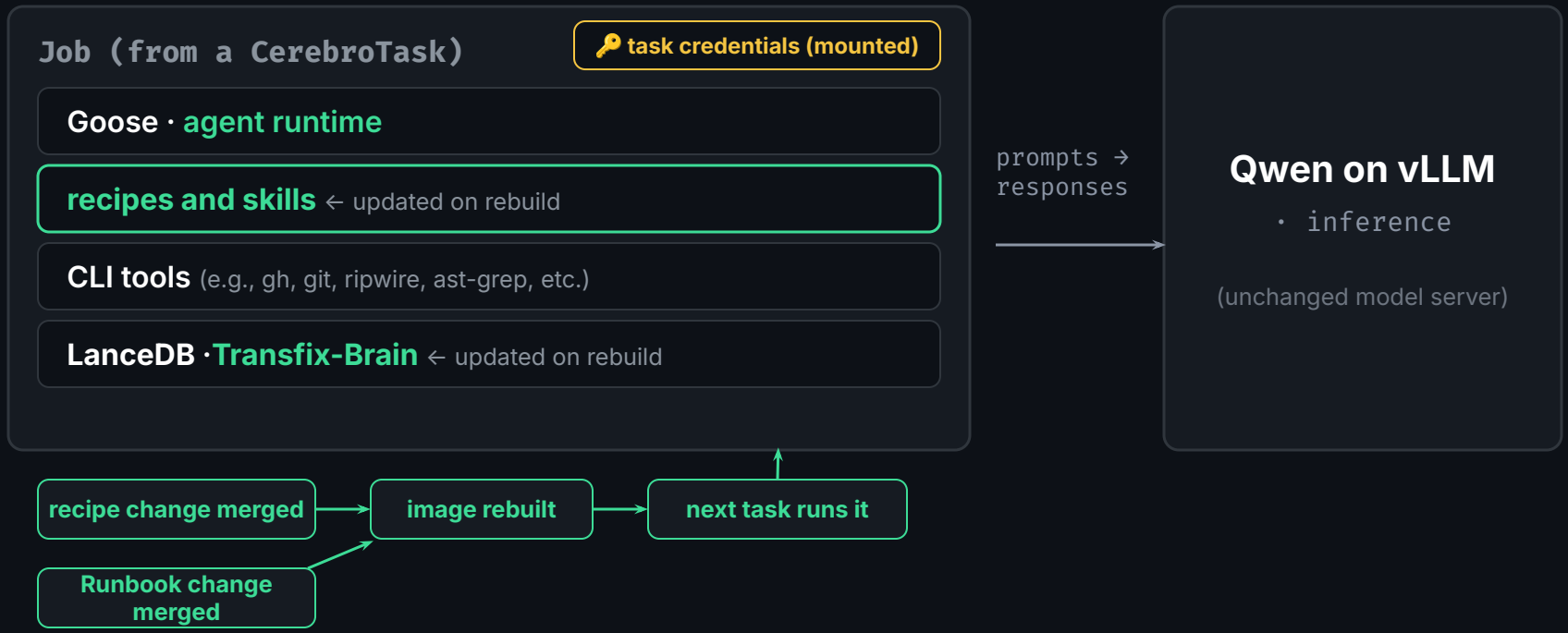
branch ...-fix-cve-...

approved by a person · merged

We fixed the condition, not the PR.

THE TASK · ONE KUBERNETES JOB

Where the learning is kept



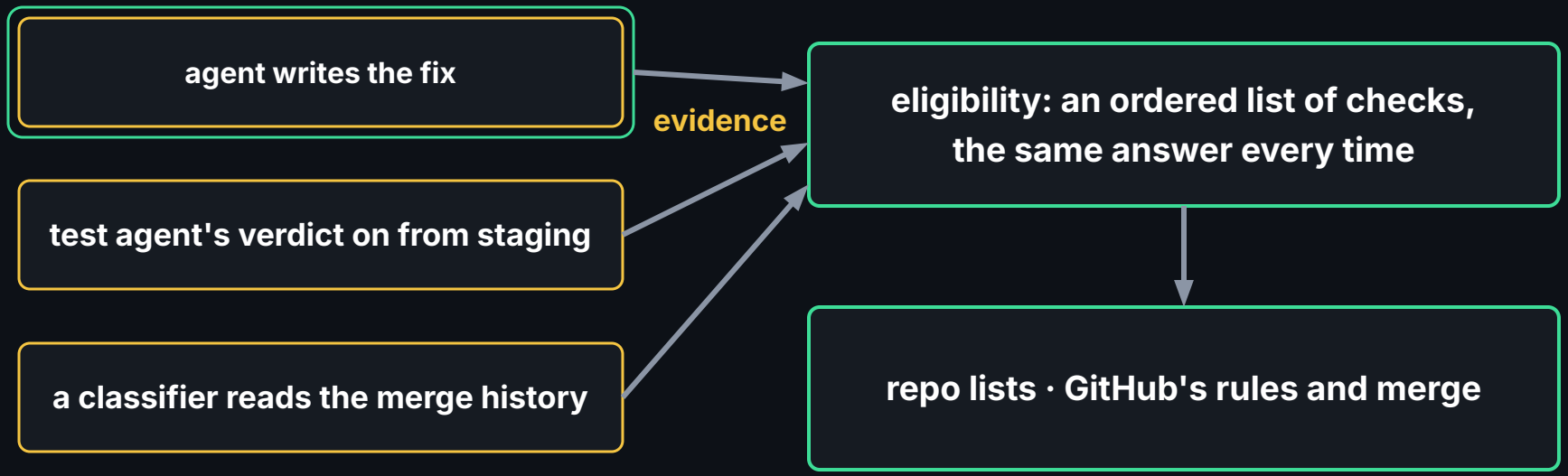
The model didn't change. The procedure around it did.

THE SPECTRUM OF DETERMINISM · IN THE PIPELINE

Who infers, and who decides

a model infers · fixed code decides

inside scripted rules: routing, lockfile checks



Models produce evidence. Code decides authority.

A rule can be wrong too.

THE MODEL · PROGRESSIVE DELIVERY FOR TRUST

Progressive delivery, authority as the payload

Partial confidence. Bound exposure · watch · keep a way to respond.

canary — analysis — success condition — promotion — abort

1. Canary step
↓
Shadow mode: propose, decide, withhold
`CerebroTask → Job`

2. AnalysisTemplate
↓
grade the work against outcomes
`ApplicationSet + CronJob`

3. Success condition
↓
An eligibility policy
`policy in the evaluator image`

4. Promotion
↓
A reviewed permission change
`a PR to the CronJob env`

5. Abort
↓
Revoke and clean up
`the same PR, reversed`

WHERE IT BREAKS

Revoking permission leaves completed merges in place

What counts as a track record

the judgment · written to the ticket

transfix#24906 · net-imap · ruby

```
repo: transfix pr: 24906 package: net-imap eco: ruby
merged_at: 2026-05-06T21:30:45Z
head_sha_at_merge: 459052a8 ...
review_proof: { approver: <human>,
commit_id: 459052a8 ... } # same head
ci_proof: { observed_checks: 66, all SUCCESS,
head: 459052a8 ... }
outcome_proof: { outcome: clean-merge,
material_failure: false,
observed_until: 2026-05-07T21:30:45Z
```

A merged remediation counts if:

- ✔ checks green on the merged head
- ✔ no one else committed to it
- ✔ no revert or follow-up in the window after
- ✔ 30 consecutive clean merges in the same ecosystem, pooled across repos

Approval told us who clicked. The clean window tells us whether it held.

A deploy per proposal

transfix-pr-staging-applicationsets.yaml (trimmed)

```
generators:
  - pullRequest:
      github:
        owner: transfixio
        repo: transfix
        labels: [staging-beast-mode]          # the switch
      requeueAfterSeconds: 60
  template:
    spec:
      source:
        kustomize:
          namespace: transfix-{{ .branch_slug }}   # one per PR
          images:
            - transfix_svc_stage:{{ .head_sha | substr 0 12 }}
      syncPolicy:
        automated: { prune: true, selfHeal: true }
        syncOptions: [CreateNamespace=true]
```

ships inside every preview

```
kind: RoleBinding
roleRef:
  kind: ClusterRole
  name: cerebro-pr-namespace-exec
subjects:
- kind: ServiceAccount
  name: cerebro
  namespace: cerebro
```

pods: get, list · pods/exec: create. This namespace only, for as long as it exists.

this exact commit

label off → it's gone

No provisioning code. No cleanup code. Reconciliation.

EPHEMERAL STAGING · ONE FIX, TESTED

What the test actually does

crass 1.0.7 → 1.0.8 · one line in Gemfile.lock

- `script` Plan: under the HTML sanitizer, no direct callers → smoke-test the app
- `script` Staging healthy · kubectl exec: crass 1.0.8 loaded
- `agent` Logs in, reads, writes, posts its evidence

Not checked, and said so: the CSS bug itself

5 PRs at once

a namespace each
verdicts in 2–22 min

Code builds the plan and checks merge eligibility. The agent’s report is evidence.

SUCCESS CONDITION · WHY THIS GRANT

What made the grant reasonable

- Supervised PRs, reviewed and merged by people
- Errors investigated; recipes and skills revised
- 30 consecutive clean merges
- A passing test on the exact commit for every proposal
- One narrow class, granted per repo and ecosystem

→ **approve and arm auto-merge**

BEFORE
a person reviews every PR

→

AFTER, FOR THIS CLASS
people review the recipes, the policy, the exceptions and the grant

PROMOTION · ARGO #4544

Permission lives in reviewed config

vuln-evaluate-cronjob.yaml

```
schedule: "0 * * * *"
env:
  VULN_APPROVAL_MIN_COMPARABLES: "30"
-   VULN_APPROVE_SHADOW: "true"
+   VULN_APPROVE_SHADOW: "false"
-   VULN_AUTOMERGE_ENABLED: "false"
+   VULN_AUTOMERGE_ENABLED: "true"
  VULN_AUTOMERGE_REPOS:
"transfix-brain,claude-config,core-platform,tms,transfix,broker-platform-svc"
```

now one grant per repo and ecosystem, e.g. rogers:npm

The code shipped first. Configuration held it.

Same arming function as the plugin workflow.

Granting authority was a PR to this file, reconciled by Argo CD. Revoking it is the same PR, reversed.

PLAN-DETECTIVE · STORAGE

Where the record lives

GitHub — PR, review, check and merge facts · read fresh every run, never cached as a second copy

Jira — the durable decision · a comment for each changed shadow decision, and one before every live action

Loki — one event per decision and per run · identifiers in the body, not in labels

Mimir — counts only: decisions, evidence per ecosystem, Jira-versus-evaluator parity

Grafana — the Vulnerability Lane dashboard · an alert when Jira and the evaluator disagree

The pod — reports and caches for one run · gone when the pod exits

PROMOTION · WHO DOES WHAT

Four roles, four responsibilities

Proposes

`Cerebro · Job`

can't approve its own PR

→

Tests

`QA agent · Job`

exec in one namespace, while it exists

→

Approves and arms

`evaluator · CronJob`

writes to Jira first, or doesn't act

→

Merges

`GitHub`

enforces the rulesets

Kubernetes RBAC: what each can touch in the cluster

Repository rules: what can happen in the repo

ABORT · VULNERABILITY REMEDIATION · REVOCATION

Withdrawal has four parts

01 — exists

Stop new actions

(a flag; a hold label for the Friday sweep)

02 — exists

Disarm pending auto-merge requests

(only ours; a human's is left alone)

03 — exists

Dismiss approvals already submitted

(only ours)

04 — always human

Completed merges

(an ordinary revert)

A merge is a merge.

automatic: A revert automatically suspends the grant

TAKE-HOME

Five questions for one workflow

Plugin · arms its own PR

Vulnerability · merges one narrow class

README · merges, audited daily

1. What work would you delegate, and which of its decisions could deterministic code make?

2. What makes a result good enough, and who can tell?

3. What has supervision taught you, and where is that written down?

4. What evidence would justify the next specific permission?

5. What would make you narrow or withdraw it?

Thank you!

Reliance doesn't require the end of inquiry.

> It needs grounds to proceed, and a willingness to revise them.

Darian DeFalco · Transfix



Scan to submit feedback