Hope Is Not a Rollout Strategy

Balancing (AI) Quality, Cost, and Risk Through Experimentation

Todd Baert



Todd Baert

- OpenFeature maintainer
- Principal Software Engineer, Dynatrace

AI-powered applications

We use models for fuzzy, complex tasks; that’s where the new problems come from



Non-deterministic output
same question, different answers



Tokens, per prompt
one line prompt change can explode cost



Prompt injection
user input is part of the instructions



Lots of knobs
model, prompt, temperature, guardrails; each one trades quality, cost and risk



Everyone tests in production

Some do it **on purpose**.

So how do we test in prod safely?



Control exposure

who gets what



Measure

what's actually happening



React

change it without a deploy

From ideas to tools



control exposure

OpenFeature + flagd

evaluation context, targeting rules, fractional rollouts



measure

OpenTelemetry

feature flag and GenAI semantic conventions



react

OpenFeature Operator

change a flag, live; no deploy

Feature flags

The maturity model

0
The ENV_VAR swamp
change = redeploy

1
Dynamic configuration
change without a deploy; centralize configuration

2
Dynamic evaluation
evaluation logic extracted from code; observable

3
Operationalized
rollouts and experiments drive decisions

Control exposure

OpenFeature primitives we use

```
// provider: flagd evaluates the targeting rules
await OpenFeature.setProviderAndWait(new FlagdProvider());

// hooks: emit feature_flag.* telemetry on every evaluation
OpenFeature.addHooks(new SpanHook(), new MetricsHook());

// evaluation context: who's asking
const context = { targetingKey: user, plan };

// evaluation API: which model does this user get?
const model = await client.getStringDetails('llm-model', 'qwen2.5:1.5b', context);
```

Measure

OpenTelemetry semantic conventions

Feature flags

```
feature_flag.key             # which flag
feature_flag.result.variant  # what was served
feature_flag.context.id      # to whom
```

GenAI

```
gen_ai.request.model              # which model
gen_ai.usage.input_tokens         # tokens in
gen_ai.usage.output_tokens        # tokens out
gen_ai.client.operation.duration  # how long
```

both have status "Development" (Nov 2026)

React

OpenFeature Operator: flags are Kubernetes resources

```yaml
# k8s/flags.yaml
kind: FeatureFlag
spec:
  flagSpec:
    flags:
      input-guardrail:
        state: ENABLED
        defaultVariant: "off"
        variants:
          "on": true
          "off": false
```

```yaml
# the operator injects a flagd sidecar
annotations:
  openfeature.dev/enabled: "true"
  openfeature.dev/featureflagsource: flag-source

# edit, then apply: live in seconds, no deploy
kubectl apply -f k8s/flags.yaml
```

The demo



A real AI workload
real models, real prompts, real tokens



Objective results
every answer is right or wrong; quality is measurable



Real failure modes
wrong answers, token cost, prompt injection



A real cluster
Kubernetes (Kind), OpenFeature Operator, flagd, Ollama, Grafana LGTM

Test in prod, on purpose

Control exposure	Measure	React
OpenFeature + flagd	OpenTelemetry	OpenFeature Operator



Flags + telemetry = test setup + assertions
targeting picks who; dashboards decide roll out or roll back



SRE tools are dev tools now
all CNCF, all open source; build it next sprint

Hope is not a rollout strategy

Thank you!

Feedback

Repo

github.com/toddbaert/hope-is-not-a-rollout-strategy

openfeature.dev

CNCF Slack #openfeature