A WALKTHROUGH · 4 SCENARIOS · 3 VIDEOS

Day-2 remediation patterns on the a *telco* GitOps stack.

GitOps + Keptn + Flagger — live, on a Kind cluster, judged by an automated harness.

PRESENTED BY

Martin E. Quintero

AI Solutions Architect · Cuemby

STACK

`flux · keptn-lt · flagger`

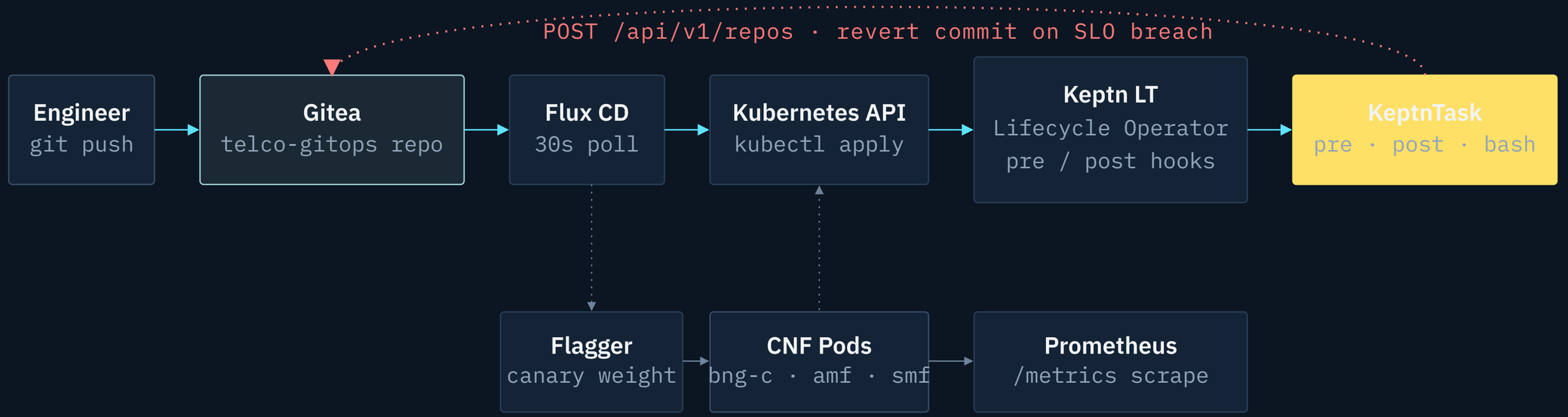
All CNCF. All declarative.

DATE

21 May 2026

Brussels, remote

The stack — all CNCF, all declarative.



Flux
you already run this

Keptn LT
Sandbox · pre/post SLO hooks

Flagger
Incubating · progressive delivery

The remediation loop
closes through Git

Four scenarios. Each ends with a passing judge.

#	SCENARIO	CNF	TRIGGER	REMEDIATION
01	Auto-rollback on SLO breach	cnf-bng-c	bad image deploy	**Git revert via Keptn post-task**
02	**Horizontal autoscale**	cnf-amf	traffic surge	HPA + Keptn resource bump
03	**Memory-leak rolling restart**	cnf-amf	memory > 250 MiB	CronJob → KeptnTask → rollout
04	**Canary failure rollback**	cnf-bng-c	bad canary metric	Flagger weight 20% → 0

Each scenario completes in < 5 min. Each ends with an automated judge verdict.

— SCENARIO 01 · SETUP

Auto-rollback on SLO breach.

THE STORY

Friday push of `v6.7.1` to BNG-c. Pre-prod was clean.

Production drops subscriber sessions within two minutes.

The platform self-heals — it detects the breach, reverts the Git commit, and rolls back. No pager.

THE WIRING

```
keptn.sh/pre-deployment-tasks: pre-deploy-check-bng-c
```

— the gate

```
keptn.sh/post-deployment-tasks: auto-rollback-bng-c
```

— the remediation

Two pod annotations. That's the integration.

WATCH FOR pre-deploy task fires · post-deploy detects breach · revert commit lands on Gitea

▶ VIDEO 01 / 03

Scenario 1 — auto-rollback.

01 Pre-deploy task fires — read the pod annotation

02 Post-deploy detects breach — PromQL + Grafana side-by-side

03 Revert commit lands in Gitea — Flux reconciles

— SCENARIO 01 · TAKEAWAY

What just happened.

01 A bad Git commit triggered the platform.

02 Keptn ran a **pre-deploy script** — the gate.

03 Keptn ran a **post-deploy script** — the remediation — which detected the SLO breach.

04 The script pushed a **revert commit** to Gitea — the *config update*.

05 Flux reconciled the revert. **Loop closed in under 5 minutes.**

Threshold lives in YAML. Operators change it through pull requests.

— SCENARIO 02 · SETUP

AMF horizontal autoscale on subscriber surge.

THE STORY

Major sports event ends. **4M UEs** hit the AMF in 90 seconds.

The control plane needs to scale — without a human in the loop.

THE MECHANIC

HPA reacts to CPU. Fast. K8s native.

Keptn watches a user-facing SLO — p95 attach latency. If HPA can't keep up, Keptn fires `bump-resources` to double CPU/memory requests.

WATCH FOR AMF instances **2 → 5** · p95 latency panel · parallel Keptn signal

▶ VIDEO 02 / 03

Scenario 2 — horizontal autoscale.

01 HPA replicas **2** → **5** — fast layer doing its job

02 Keptn `bump-resources` fires — slow layer compensates

— SCENARIO 02 · TAKEAWAY

Two-layer scaling.

SLO LAYER	**Keptn · bump-resources**	slow · seconds–tens · user-facing SLO	p95 attach latency > SLO

REACTS TO A METRIC SUBSCRIBERS FEEL

INFRA LAYER	**HPA · CPU target**	fast · seconds · K8s native	CPU > 70%

Pattern transfers to **SMF** (PDU-session-setup rate) and **UPF** (bandwidth/pod). Same KeptnTaskDefinition — only the PromQL changes.

SCENARIO 03 · SETUP

AMF memory-leak rolling restart.

THE STORY

A bad AMF release leaks **200 MiB / min**.

In prod that's a **4 a.m. pager** 40 min from now. Caught in 2.

THE MECHANIC

A CronJob `memleak-watchdog` ticks every **60s**.

Reads `KeptnMetric` for `pod-memory-usage`. If **> 250 MiB**, creates a `KeptnTask restart-on-leak-<ts>` and triggers a rollout.

01
CronJob
60s tick

→

02
KeptnMetric query
pod-memory-usage

→

03
> 250 MiB ?
threshold gate

→

04
KeptnTask + rollout
restart-on-leak-<ts>

▶ VIDEO 03 / 03

Scenario 3 — memory-leak restart.

01 Grafana sawtooth — the loop in steady state

02 KeptnTask history — post-mortem evidence

— SCENARIO 03 · TAKEAWAY

Observability driving action, not Slack.

01 The sawtooth *is* the loop closing in steady state.

02 **No custom controller.** 30 lines of bash on a CronJob — the entire mechanism.

03 KeptnTask history *is* the post-mortem. Every restart is a row.

04 In prod: page after **N consecutive restarts** in a window. The loop buys the on-call engineer time.

— SCENARIO 04 · PROPOSED

NOT RECORDED — PROPOSED PATTERN

BNG-c canary failure with progressive rollback.

THE STORY

Promote `v6.7.2` of the BNG.

Flagger shifts **10%** → **20%** of traffic, evaluates success rate.

Bad metrics → abort, roll back to primary.

THE MECHANIC

`Canary` CR on the Deployment + `MetricTemplate` against Prometheus.

Threshold: **3 failed checks** → **halt**.

Max bad-traffic exposure = stepWeight at abort.

WHY IT MATTERS

In telco — BNG, AMF, SMF — a bad rollout drops live subscriber sessions. **Flagger bounds the blast radius** to step-weight %, before scenario 1 has to clean up.

○ PROPOSED · NOT YET RECORDED

Why scenario 4 belongs on the roadmap.

01 **Routine promotions are the common case** — most prod changes in a telco are version bumps, not new features. Scenario 1 is the safety net; scenario 4 is the everyday path.

02 **Subscriber-facing CNFs can't take 100% blast radius** — BNG, AMF, SMF carry live sessions. Flagger bounds exposure to step-weight %; bad releases never reach all traffic.

03 **It composes with scenario 1** — Flagger guards 0 → 100%; Keptn guards after 100%. The slow-burn degradations that only surface post-rollout get caught by the rollback loop.

04 **Same wiring as scenarios 1–3** — a `Canary` CR on the Deployment plus a `MetricTemplate` against Prometheus. No new tooling. Roughly half a day per CNF on top of what's already shown.

— SCENARIO 04 · TAKEAWAY

Two layers of safety.

Flagger

routine canary releases

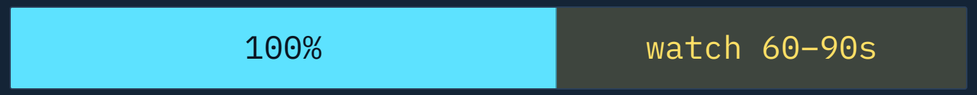
guards 0 → 100%

Bounds bad traffic to **stepWeight%**.

+

Keptn auto-rollback

slow-burn degradations



guards **after** 100%

Catches what only surfaces post-rollout.

Composition: release passes Flagger → 100% → Keptn watches 60–90s → if still bad, revert commit. You don't pick one.

Three questions, three answers.

Q1	What are `ngntelco` and the realtime visualisation tool?	**Out of scope.** Those are terms from your prototype. Happy to discuss your specific tools after this.
Q2	Effort towards Keptn vs. the rest of the prototype?	**Keptn ≈ ~1 day per CNF** — 2 task definitions, 2 pod annotations, 1 analysis definition. **No custom controllers.** The rest is Flux + Prometheus + Flagger — all CNCF, all replaceable.
Q3	What triggers the *config-update remediation* in Keptn?	**The `keptn.sh/post-deployment-tasks` pod annotation.** Flux applies a new `keptn.sh/version` → Keptn creates a `KeptnWorkloadVersion` → post-deploy phase runs the task → bash container queries Prometheus → on breach, pushes a revert commit to Gitea. `You watched this in Video 2 — paused twice.`

— 04 · HONEST CLOSING

What works · what's still open.

WHAT YOU SAW WORKS

- ✓ 4 day-2 patterns. All CNCF. All declarative.
- ✓ GitOps audit trail end-to-end.
- ✓ Workload-agnostic — same code across your CNFs.
- ✓ The patterns compose — S1 + S4 = two layers of safety.

WHAT'S STILL OPEN

- ! Keptn LT is **CNCF Sandbox** — not Incubating / Graduated.
- ! Multi-cluster orchestration not shown — single Kind cluster.
- ! Data-plane CNFs (UPF) need multus/SR-IOV — out of scope.
- ! Keptn ↔ Flagger has a rough edge (`D-013`) you'll hit.

That's the prototype.

SHOWN	Scenarios 1–3 — auto-rollback, autoscale, leak restart. Live, judged.
PROPOSED	Scenario 4 — canary rollback with Flagger. Same primitives, same wiring.
PATTERN	GitOps audit trail end-to-end · workload-agnostic · CNCF only.

$ Questions.