KubeCon CloudNativeCon North America 2026

#KubeCon #CloudNativeCon

The Promises We Keep at Scale

A Contract-Driven Guide to Platform SLOs



Vasudev Bongale, Swati Shridhar Nair

Compute Infrastructure

LinkedIn





Our platform at scale

The Kubernetes platform behind LinkedIn's applications

Running stateless and stateful workloads across many clusters and regions

5M+
Pods



500K+
Hosts



3,000+
Applications



110K+
Weekly deployments

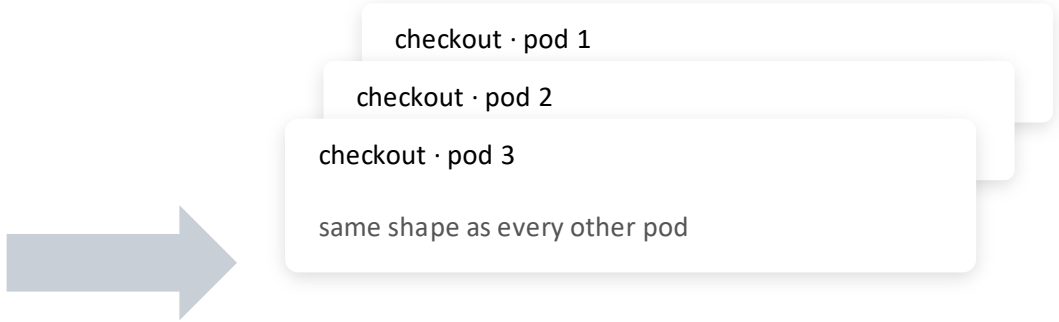


Deployment model

User writes ~10 lines

```
kind: LiDeployment
metadata:
  name: checkout
spec:
  replicas: 3
  versions:
    app: 2.2.5
  resources:
    cpu: 10
    memory: 32Gi
```

Platform owns pod spec and runs it



- Injected init containers
- Applied runtime contract
- Volumes, security, placement
- Policies preventing custom node selectors

What we'll answer today



What should the platform promise its users?



How do we know we are keeping those promises?

Our approach



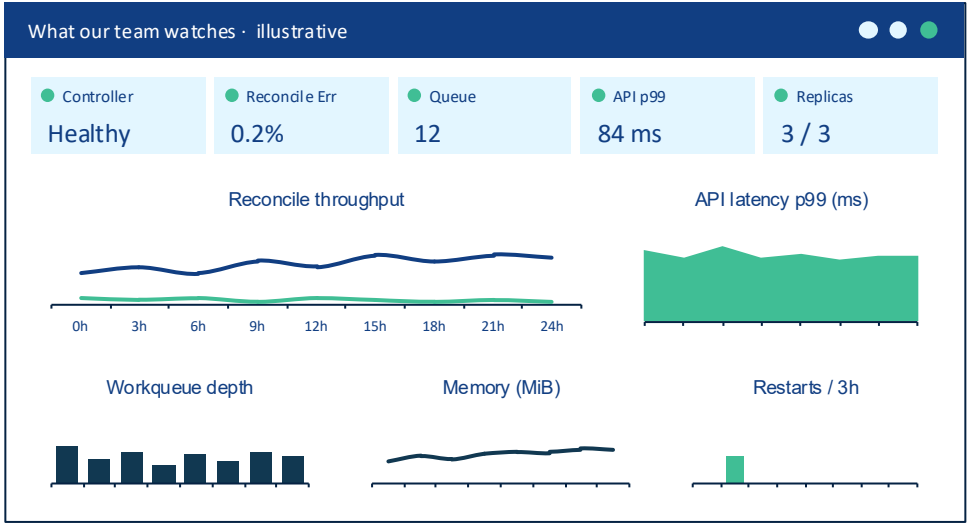
Understand what users care about

Define the promises

Measure what the platform owns

Test and evolve

What users care about



They expect one platform, even though many teams build it

Users experience the platform only through their workloads

What users ask at each stage



Rollout	Start	Run	Clean up
Did my change go out?	How long until my pods are up?	Is my app staying up?	Has my old pod let go?

Four promises

A promise is what users can expect from the platform

Rollout	Start	Run	Clean up
			
“Your rollout won’t fail because of the platform.”	“The platform won’t keep your pods waiting.”	“The platform won’t compromise your workload’s availability.”	“Your deleted pods are gone on time.”
Rollout success rate	Scheduling + preparation	Workload readiness	Overdue termination

From promise to SLO



"Your pods are scheduled quickly."

Illustrative example

Scope

Which workloads

Long-running services, per region

SLI

What we measure

% of eligible pods bound to a node within a threshold

SLO

Good enough

≥ 99%

over 7 days

Workload readiness contract


```
$ kubectl get lideployments                    # simplified output

NAME        REPLICAS   READY
checkout    1/3        False
search      1/3        False
inventory   2/3        False
feed        3/3        True
```

Workload readiness contract

$ kubectl get lideployments

simplified output

NAME	REPLICAS	READY	CATEGORY	REASON
checkout	1/3	False	App	AppFailedToStart
search	1/3	False	Infra	InitializationFailure
inventory	2/3	False	Unknown	ProgressDeadlineExceeded
feed	3/3	True	-	-

▶ Making Application Rollouts Observable, Actionable and Boring, KubeCon NA 2025

Workload readiness contract

$ kubectl get lideployments

simplified output

NAME	REPLICAS	READY	CATEGORY	REASON	CONTRACT
checkout	1/3	False	App	AppFailedToStart	Met
search	1/3	False	Infra	InitializationFailure	Not met
inventory	2/3	False	Unknown	ProgressDeadlineExceeded	Not met
feed	3/3	True	-	-	Met

Workload readiness SLO

KubeCon CloudNativeCon North America 2026

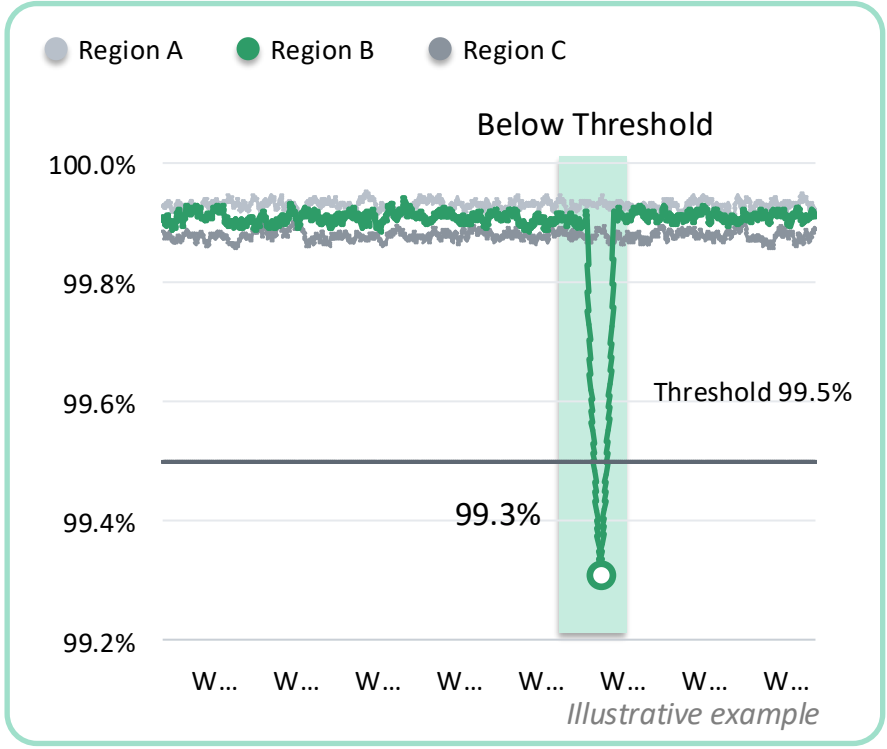
SLI

$$\frac{\text{workloads meeting the contract}}{\text{eligible workloads}}$$

sampled every minute

A minute is good if SLI ≥ 99.5%

SLO: ≥ 99% of minutes good, 7 days



Per-minute, sampled · workloads meeting the contract

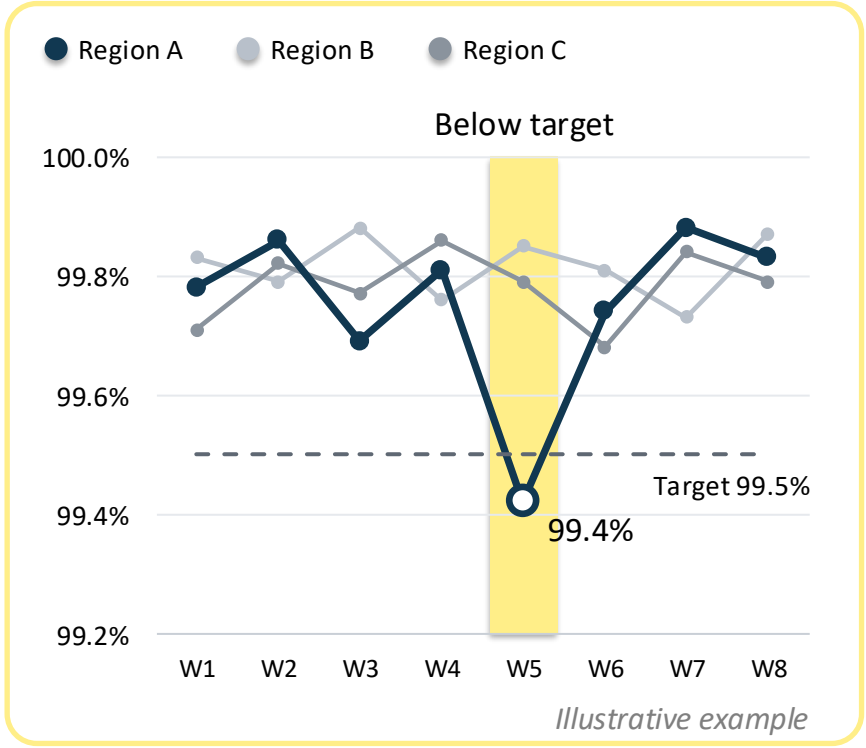
Rollout success SLO

SLI

$$\frac{\text{rollouts without a platform-caused failure}}{\text{rollouts that ended in the window}}$$

SLO: ≥ 99.5% over 7 days

Needs a minimum number of rollouts per window

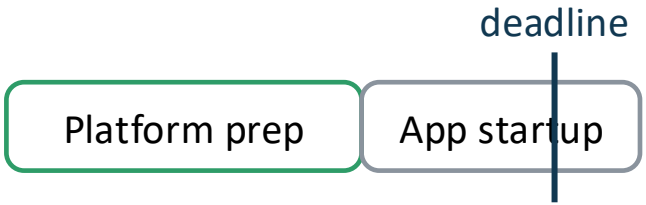


Weekly · rollouts that ended in each 7-day window

Attribution changed the platform

Before

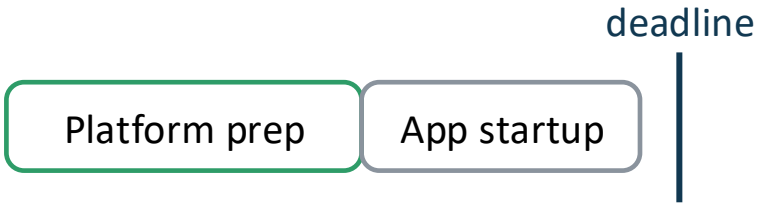
One deadline for everything



✗ Healthy app rollout failed

After

Platform work gets its own time



✓ Healthy app rollout didn’t fail

✗ Crashing app still fails early

How long does the platform make you wait?

Three promises about time



Scheduling



Platform prep



Overdue termination

Measuring the scheduling wait

Scheduling wait

Eligible pod → bound to a node

Platform prep

Bound → app starts

App startup

not counted

Eligible means the pod is ready for us to place, with nothing outstanding on the workload's side.

SLI: % of eligible pods bound to a node within threshold

SLO: ≥ 99% over 7 days

The threshold varies by workload class.

Where do pods wait to be scheduled?



Fleet maintenance & remediation

Rolling node drains or automated repairs temporarily reduce local capacity



Cluster capacity

Delayed autoscaling during bursty scale-outs of workloads



Scheduler plugin constraints

Strict topology spread constraint for a workload reduced the eligible nodes

Measuring platform prep latency

Scheduling wait

Eligible → bound

Platform prep

Bound → app starts

App startup

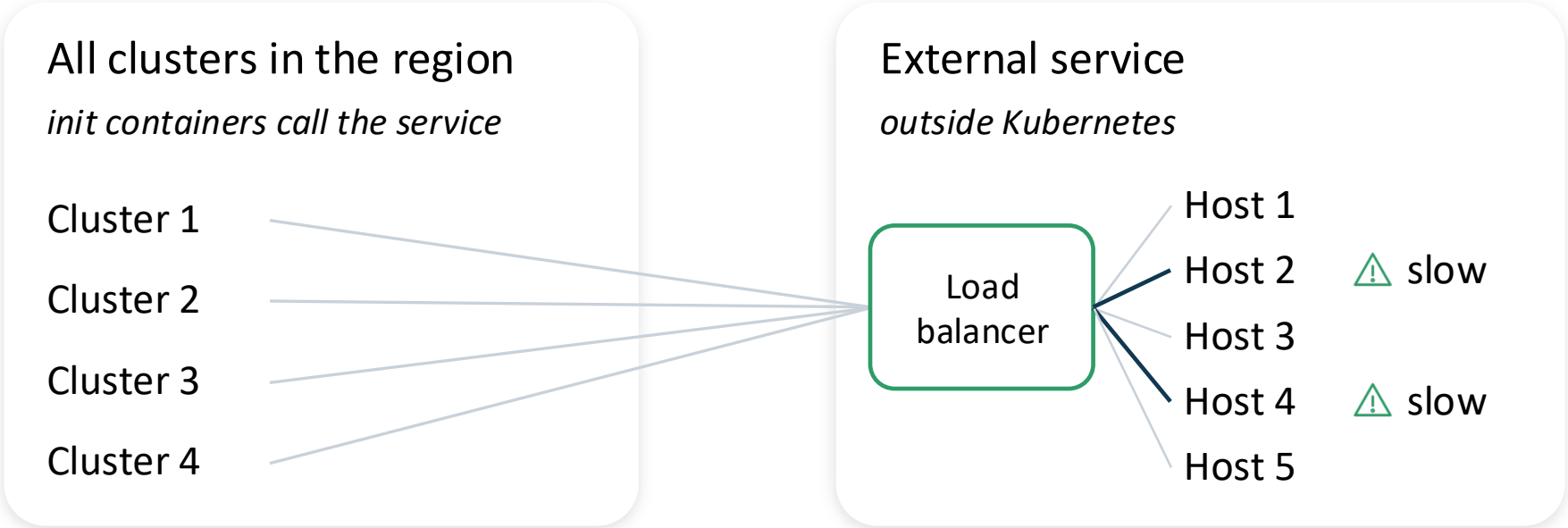
not counted

Only our work counts: infra init containers
Excludes app's own init containers

SLI: % of pods that finish platform prep within threshold

SLO: ≥ 99% over 7 days

An external service slowed pod startup



Platform promises require explicit contracts with dependent services

Their aggregate SLO was green. Our workload promise breached.

Measuring overdue termination

App shutdown

grace period, not counted

Platform cleanup

Deletion timestamp → pod gone

What we own directly: releasing our finalizers.

Cleanup also needs a healthy node: kubelet, runtime, CNI, CSI drivers.

A stuck pod can hold up the next rollout and node maintenance.

SLI: % of pods in terminating state beyond a threshold

SLO: < 0.1% over 7 days

Pod cleanup stalled on healthy nodes

- Node Ready
- Kubelet healthy
- Finalizers clear
- No maintenance

Yet pods were stuck in Terminating

A containerd bug that dropped container-stopped events during high pod churn.

Upstream: containerd issue #12344, fixed in 2.0.7

Put your SLO to the test



Proven

What known platform failures have you tested it against?



Accurate

Did it capture the impact, and the recovery, correctly?



Early

Would an alert on this SLI fire before the SLO is breached?



Owned

Who responds, and what action can they take?

Takeaways

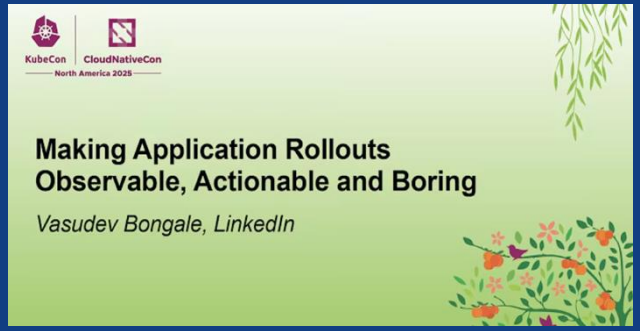
- ☐ Start with promises, not metrics
  Write down what workloads need, then decide what to measure.
- ☐ One metric, one clear question
  If a metric starts answering more than one question, split it.
- ☐ Attribute before you aggregate
  Separate what the platform caused from what the app caused, then roll it up.
- ☐ Close the feedback loop
  Test your SLO with known failures, then improve the platform and the SLO.

The value of an SLO isn’t the number it reports. It’s the promise it helps you keep.

KubeCon CloudNativeCon
North America 2026



▶ Watch the talk



▶ Watch the talk

Thank you!

Q & A

Scan to share your feedback →

The Promises We Keep at Scale: A Contract-Driven Guide to Platform SLOs