Spark on Kubernetes

The Gap Between “It Works” and “It Works Well”

Damon Cortesi, Airbnb
KubeCon + CloudNativeCon North America 2026



@dacort

❯whoami

Staff Software Eng @ Airbnb

Data Platforms since 2010

Stats nerd since 1996

Open Source Contributor

Owner of an AI project named

“Forge”

Spark @ Airbnb



Aggregate Compute Capacity (p90) · 2025 → 2026

1.38PiB

Memory

was 1PiB · ▲ 38%

5.8k

Concurrent apps

was ~3k · ▲ ~2x

629k

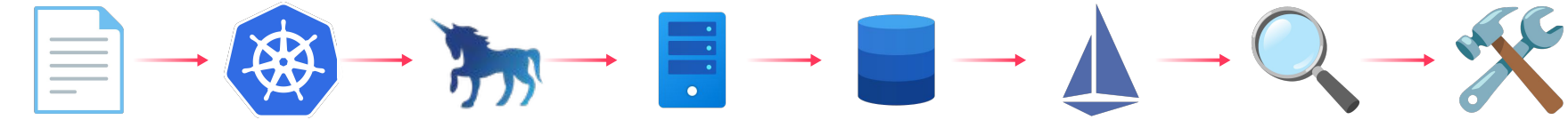
Daily jobs

was 696k · ≈ flat

Spark job submissions on Kubernetes



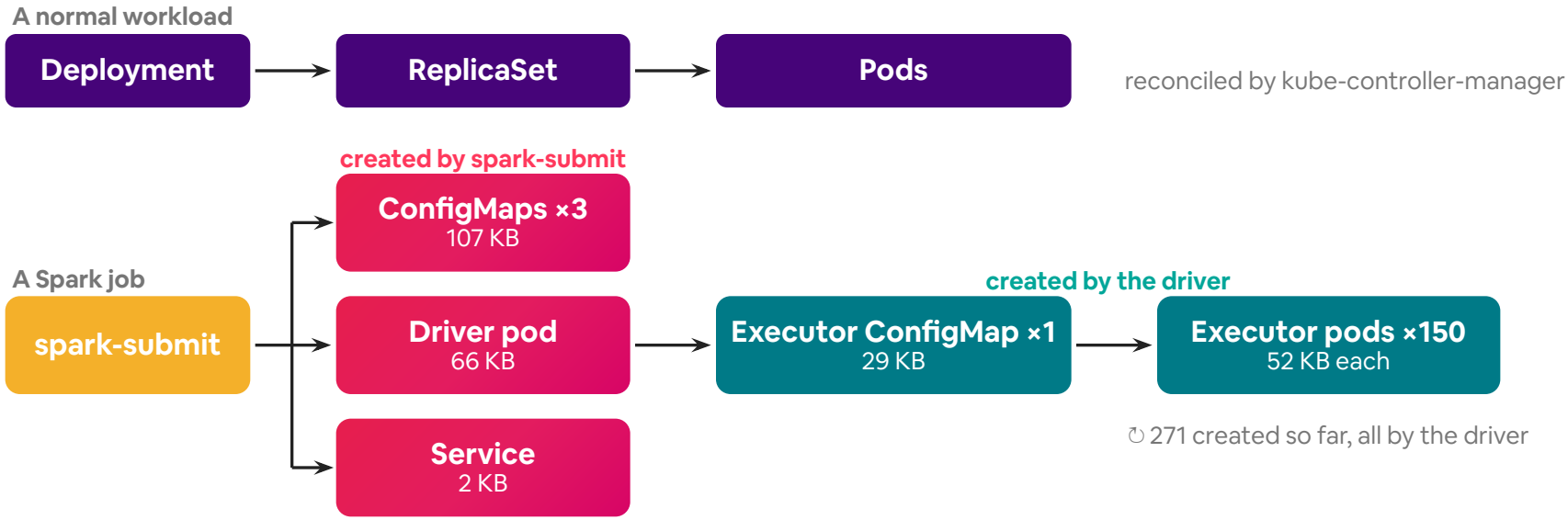
A day in the life of a Spark job



Submission → Control Plane → Scheduling → Node (Drivers & Executors) → Disks → Networking → Observability → Tooling



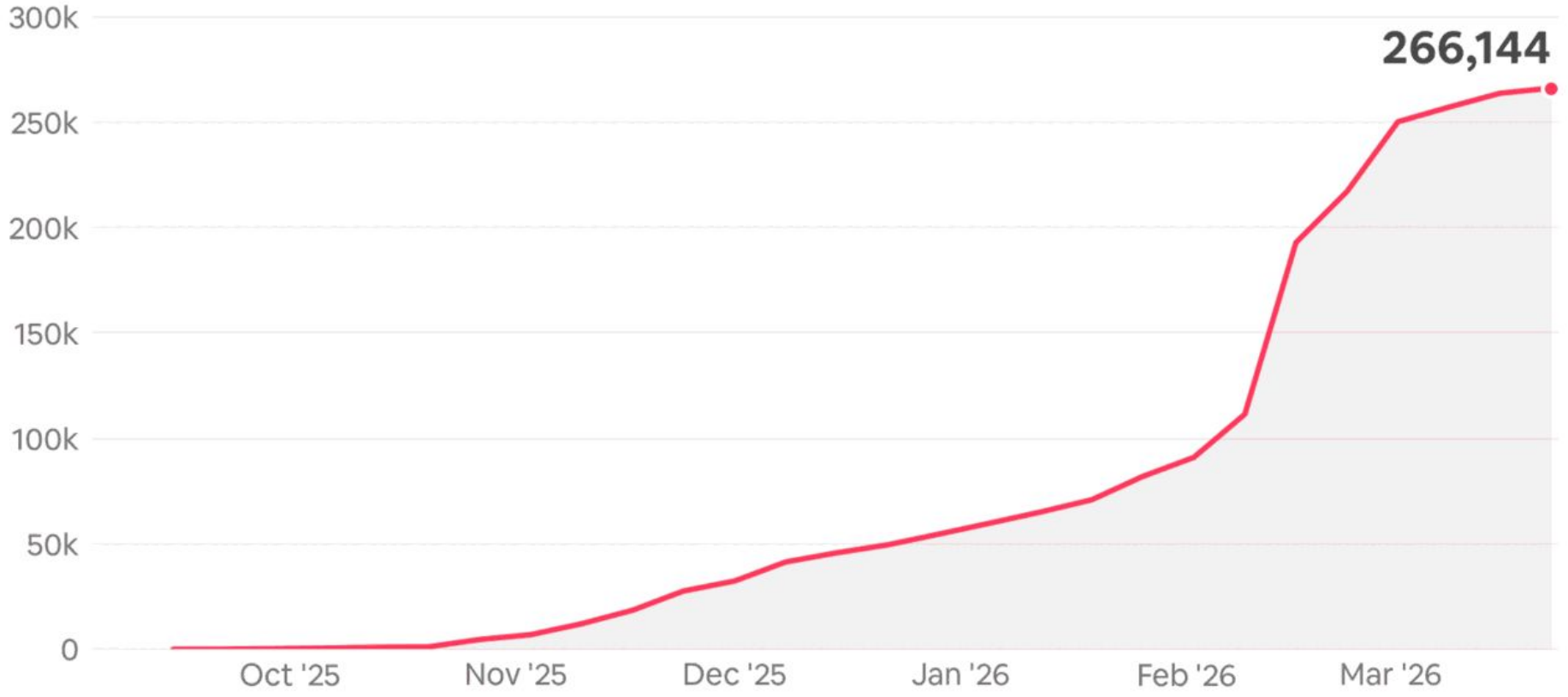
One job, many objects



≈ 8 MB of API objects for one job

One running job, 150 executors. JSON size incl. managedFields, measured 2026-10-15.

“I think there might be a leak…”



Check your config.
Stress test.

Control plane

API server · etcd · Argo CD

etcd tuning



events TTL

1 hour → 10 minutes

p99 latency: ~0.5s → <25ms

Object count: ~2M → ~400K

ArgoCD toYAML tooMuch

App CRs ~500 → 20MiB

etcd: 1.8 → 0.8 → 0.28GB

“lowest I've seen etcd in like a year”

compaction interval

5min → 2.5min

peak etcd from 9.5 → 5.4 GiB

p50 job runtime = 2.5min

Turn failures into waits

APF back on

restart ≠ outage

a YuniKorn restart can't swamp the API server

APF was off; two schedulers' full LISTs did

reject → queue

429 → wait

a job waits instead of failing mid-creation

cost: some latency + timeouts

our own reconciler

QPS → 100 / burst 200

cleanup keeps pace with ~400K pods/hr

~6,000 zombie pods cleared

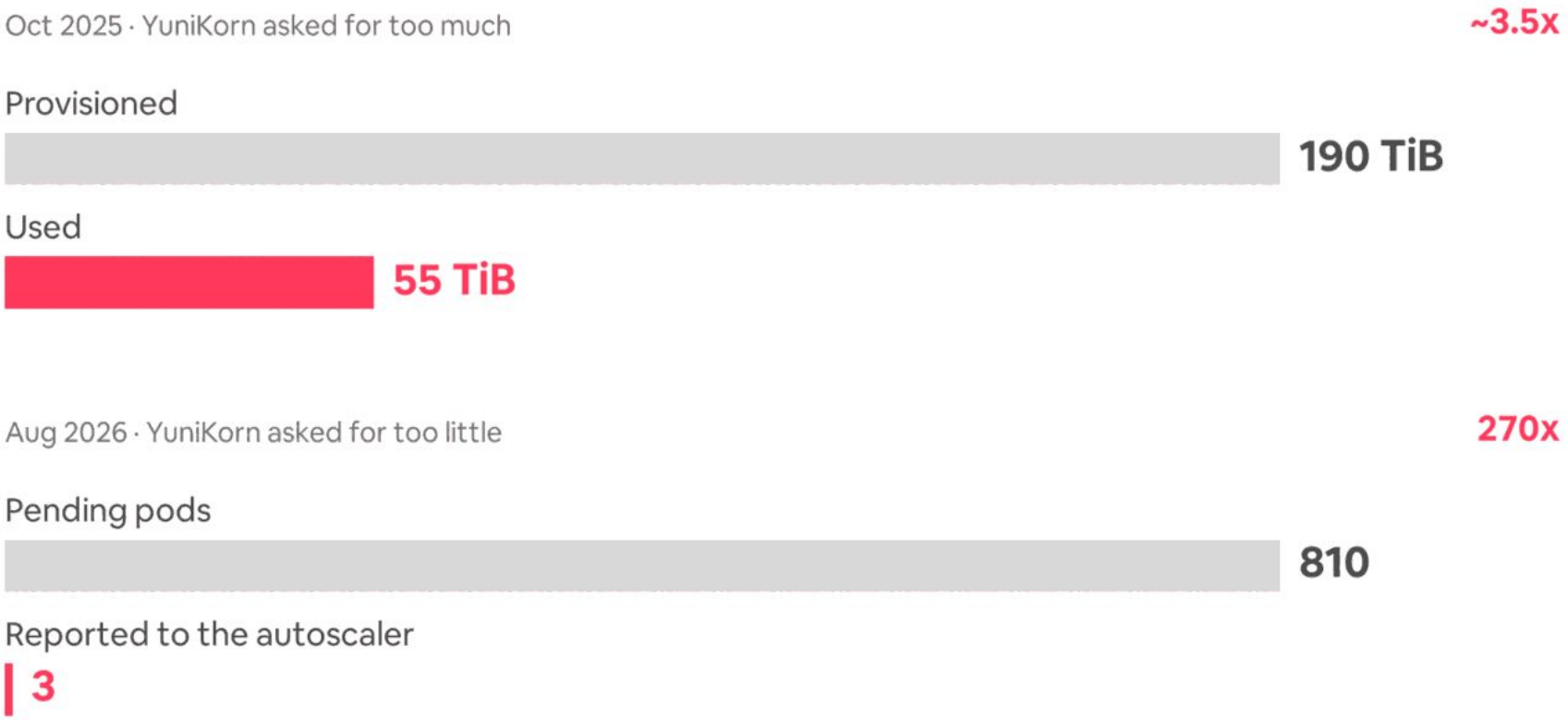
Nothing broke.
Everything was just slow.

Scheduling



YuniKorn · Karpenter

Too much, then too little



YUNIKORN-3148



YuniKorn App Sorting

1. FIFO based on the **Driver** time
2. Earlier “apps” can block others with Dynamic Allocation




sort.policy: fifo
sort.policy: fair

Use KWOK to simulate

kind + KWOK + Karpenter + YuniKorn: replay a scheduler bug on a laptop

Nodes

Is there a bug? Or is it just physics...

Something something thundering herd...



Four bugs, one kill path

All four ended the same way: node marked dead → force-deleted, PDBs ignored

Mar	May	Jun	Jun
NPD log rotation	CRI-O panic cri-o#9663	NPD OOM-kills its own probe	Kubelet dead-container GC

Our health checks were killing healthy nodes.

Disk

throughput · ephemeral storage

250 MB/s
disk default



750 MB/s
actually works

Provision for the 99%.
Let the 1% ask.

Ephemeral disk

97.3%

of jobs use <1 GB

0.1%

use >50 GB

2.1%

of apps get an on-demand PVC

Network

Cilium · the mesh

~400K

pods created every hour across our Spark clusters

1.76M

Cilium identities

65k

the limit

exportTo: ~

one annotation kept Spark's churn out of the online mesh

Spark

the fork we carry

In Spark, the driver is its own controller.

SPARK-34948 · ConfigMaps outlived their jobs

SPARK-35174 · every submit opened a pod watch

SPARK-40379 · evictions didn't count as failures

SPARK-36052 · cap pending pods to spare etcd

SPARK-35714 · executors hung on shutdown

SPARK-41210 · executor failures never failed the app

32 upstream fixes backported in 15 PRs

300+

Kubernetes tickets closed for Spark 4.0

maxNumFailures



Observability

my favorite

~4M

delete markers in the event-log bucket. Then listings started failing.

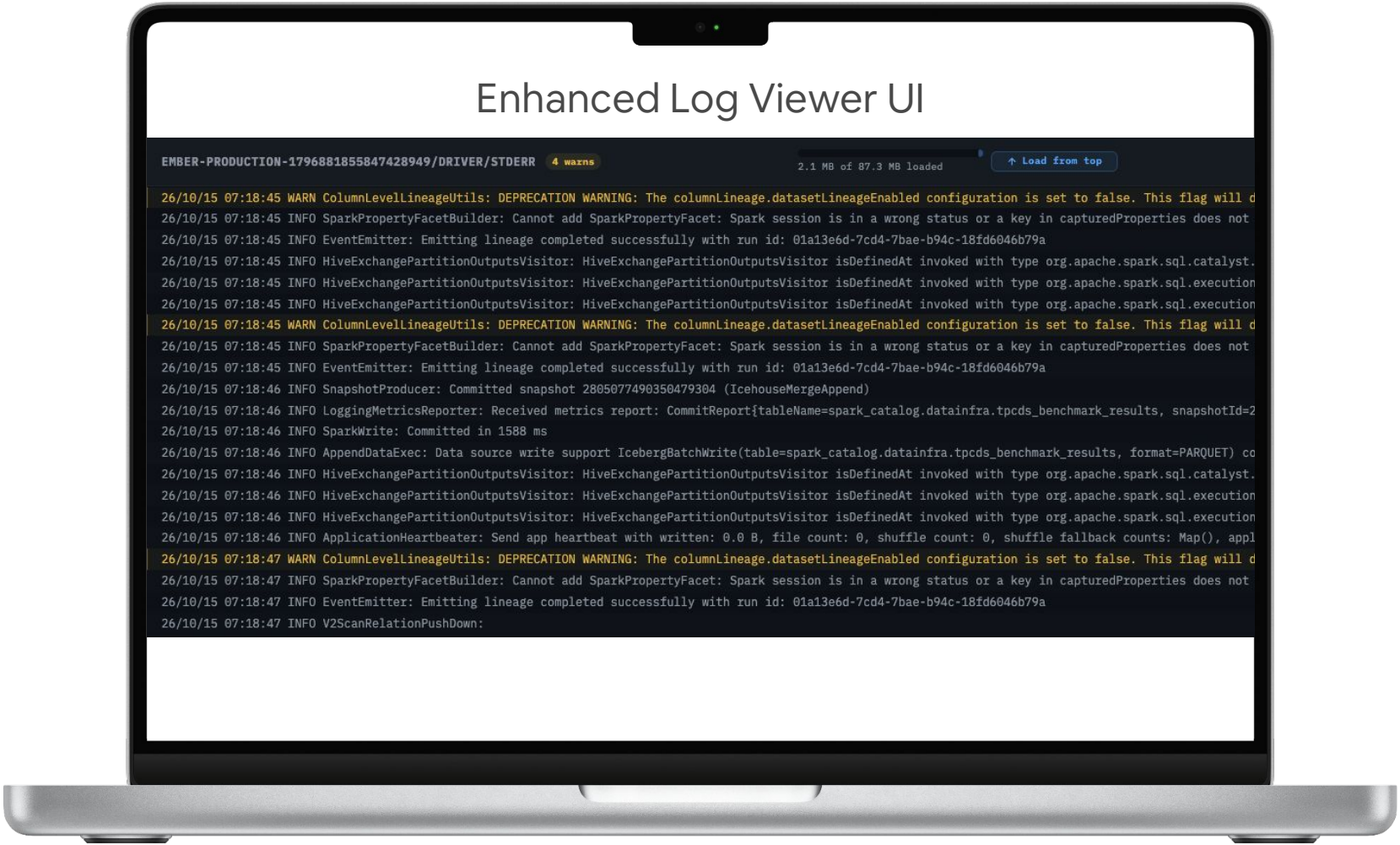
Losing a file is annoying. Losing one line is a mystery.

Fluent Bit's line buffer silently dropped the big ones

A 64 MB init container set the whole pod's disk limit.

The main container had no limit, so 64 MB became the pod's. Celeborn hit it first.

Enhanced Log Viewer UI



Tooling

when off-the-shelf ran out

k8s-cleaner

Analyze, then clean

spark-scout

Any ID in, answers out



Agentic skills

Debugging runbooks that run

eBPF tracer

Allowlists from what jobs actually touch

18 of 34

fixes are ≤10 lines. Finding them took time.

Check your config. Stress test.
Repeat.

Thank you!

@dacort everywhere