Member-only story

This Tiny Open-Source Project Is Doing What Kafka Couldn't

Turns out, the real bottleneck wasn't Kafka. It was us pretending to understand it.

Dax Follow 5 min read · Nov 6, 2025

1. The Night Kafka Broke Me

It was 2:17 a.m.
My Slack was on fire. Grafana dashboards were red.
Kafka's consumer lag graph looked like an EKG right before flatline.

I remember muttering, "It's just another offset issue."
It wasn't.

For months, our team had been scaling Kafka clusters like we were playing whack-a-mole with partitions. More brokers. More topics. More YAML. Every fix bought us maybe a week of peace — until one day, a minor schema change brought the whole thing down.

Honestly? I used to love Kafka. The elegance of streams, the sense of power in distributed messaging — it felt like *real engineering*. But somewhere along the line, the complexity started to feel like a full-time job.

You know that moment when you open `server.properties` and just… sigh? Yeah.

2. When I Stumbled on a "Tiny" Alternative

One weekend, out of frustration, I Googled:

> "Kafka alternative that doesn't require a PhD in ZooKeeper."

That's when I found **Redpanda**.

It sounded too good to be true:
Kafka-API compatible, but no JVM, no ZooKeeper, written in C++. Promised better latency and fewer moving parts.

I rolled my eyes. *Marketing fluff,* I thought.
But curiosity won.

```
$ docker run -it --rm -p 9092:9092 docker.redpanda.com/vectorized/redpanda start
```

It started in three seconds.
Three.

No cluster config. No external dependencies.

It just… worked.

And honestly? I didn’t see it coming.

3. The Benchmark That Made Me Swear Out Loud

I wrote a quick Go script to test throughput.

```
package main
import (
    "fmt"
    "github.com/segmentio/kafka-go"
    "time"
)

func main() {
    w := &kafka.Writer{
        Addr:     kafka.TCP("localhost:9092"),
        Topic:    "load-test",
        Balancer: &kafka.LeastBytes{},
    }
    start := time.Now()
    for i := 0; i < 100000; i++ {
        _ = w.WriteMessages(nil, kafka.Message{Value: []byte(fmt.Sprintf("msg-%d
    }
    fmt.Println("Duration:", time.Since(start))
}
```

Kafka (on the same laptop):

```
Duration: 7.3s
```

Redpanda:

```
Duration: 3.1s
```

No JVM tuning. No GC pauses. No brokers syncing partitions like gossiping teenagers.

I actually laughed out loud.
Then I reran it three times, just to be sure.
Still faster.

That's when it clicked.
Or maybe snapped.

4. Kafka's Real Problem Isn't Speed — It's Sanity

The irony is, Kafka isn't "bad." It's brilliant at scale. But for 90% of teams, *we don't need Google-scale pipelines*. We just need something reliable that won't eat a weekend every time a broker dies.

Kafka makes you feel like a distributed systems engineer.
Redpanda makes you feel like a developer again.

That difference matters.

With Redpanda, the setup was literally one binary. No ZooKeeper, no JVM, no tuning dark magic. I ran it on my laptop, on a t2.micro, even inside WSL — same results every time.

Here's the thing nobody tells you:
Kafka's complexity isn't a feature. It's inertia disguised as architecture.

I get why it happened. Kafka grew up in the Hadoop era. It was built for a world where disk was slow, clusters were manual, and ops teams had time. Redpanda grew up in a world of SSDs, Docker, and CI/CD pipelines where patience is a scarce resource.

And you feel that difference in every command you run.

```
# creating a topic in Kafka
$ kafka-topics.sh --create --topic orders --bootstrap-server localhost:9092 --pa

# creating a topic in Redpanda
$ rpk topic create orders
```

That's it.
No ceremony. No anxiety. Just create and go.

5. What "Tiny" Actually Means

When I first read "tiny open-source project," I assumed it meant hobby-level. Turns out, Redpanda's codebase is *lean*, not *fragile*.

The team behind it built a custom thread-per-core scheduler that maps perfectly onto modern CPUs. No garbage collector, no context switching overhead.

Their benchmark numbers?
1.7× faster on average for the same workloads, with *lower tail latency* —

meaning fewer nasty surprises at 3 a.m.

But the best part? Kafka clients don’t even know the difference.
You can drop Redpanda in and your existing producer/consumer code keeps working.

That blew my mind.

It’s like someone replaced your car’s engine overnight and it suddenly runs smoother, cheaper, and doesn’t catch fire every third trip.

6. The Moment I Realized I Was the Bottleneck

Here’s the embarrassing part.

For weeks, I blamed Kafka for our pipeline failures — timeouts, backpressure, offsets stuck in limbo. But when Redpanda ran flawlessly on the same workloads, I had to face an uncomfortable truth:
The problem wasn’t Kafka alone.
It was how *we* were using it.

We over-partitioned. We under-monitored. We treated it like a magical queue instead of the distributed log it really is.

So yeah, Redpanda didn’t just outperform Kafka. It exposed our illusions of competence. And that’s uncomfortable.
But also freeing.

7. The Day It Hit Production

When we finally rolled Redpanda into staging, I half-expected something to explode.

But logs stayed quiet. Lag stayed low. Metrics were… boring.

And boring is beautiful.

```
$ rpk cluster health
Healthy: true
Under-replicated partitions: 0
Offline replicas: 0
```

That line — “Healthy: true” — felt like therapy.
After months of wrestling with config files and GC logs, *silence* was the sweetest alert.

8. Lessons I Didn’t Expect to Learn

1. Complexity isn’t sophistication.
Kafka made me feel smart. Redpanda reminded me that smart code should feel simple.

2. Tools shape behavior.
When setup takes hours, we postpone experimentation. When it takes seconds, we explore more.

3. Boring tech wins.
If your system needs constant attention, it’s not a system — it’s a pet. And I’m done being a zookeeper.

4. “Tiny” projects can reshape giants.
Don’t underestimate a small team with focus. The best ideas often start as rebellion.

9. What’s Next for Kafka (and for Us)

I don’t think Kafka is going anywhere. It’ll remain the backbone of massive enterprises for years.
But I do think Redpanda — and projects like it — are rewriting what “developer-friendly infrastructure” means.

Because, let’s be honest, we’ve romanticized complexity for too long.
We equate scale with credibility, verbosity with control.
And we forget that most of us just want a message queue that doesn’t collapse under its own weight.

Maybe that’s what open-source is supposed to do — question the giants it once worshipped.

10. The Quiet Ending

A few weeks after the migration, I opened our dashboards out of habit. No alerts. No chaos. Just a steady stream of green.

I paused.
The CI didn’t.

And that’s when I realized:
We didn’t fix Kafka.
We fixed *ourselves*.

> **“Most outages aren’t technical. They’re emotional.”**

Red Panda | Kafka | Open Source | Cloud Infrastructure | Distributed Systems

Written by Dax

255 followers · 15 following

Dropping thoughts into the void of the internet's memory palace. Future archaeologists will thank me.

Follow

Responses (6)

David B Chase

What are your thoughts?

Jeffrey M. Birnbaum
4 days ago

You should check out AMPS at https://tinyurl.com/5n6c77rc In a nutshell it is simpler, lower latency and supports content filtering. Redpanda certainly relieves some of the large operational cost compared to Kafka but AMPS is even more "developer friendly".

10 Reply

Enzo Lombardi
Nov 6

If you don't need strict durability requirements Blink might be an even better fit: https://github.com/Cleafy/Blink

It's not RedPanda but it has a quite good niche of usages: it starts in milliseconds.

5 Reply

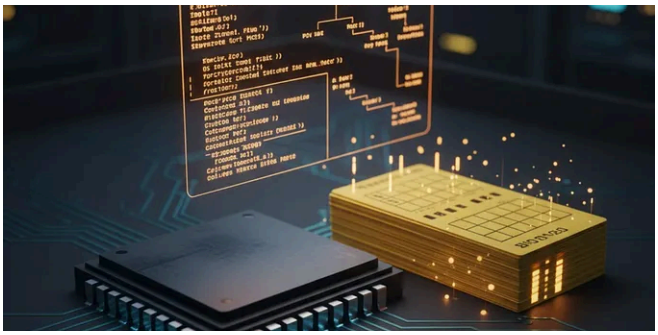
Oisin Grehan
21 hours ago

Did the same, except migrated to azure event hubs with kafka endpoint. Confluence cloud bill went from $25k/year to zero. Event hubs cost < $10 a month.

3 Reply

See all responses

More from Dax



Dax

Fortran Outsmarted Our Billion-Dollar AI Chips

We spent millions optimizing CUDA kernels—then a 1970s Fortran loop quietly ran faster.

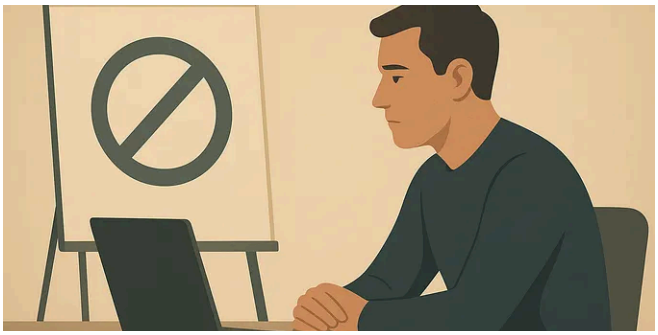
Oct 31 878 32

Dax

The Secret Language Powering macOS: Why Apple’s Kernel Team...

For decades, C was the beating heart of Apple’s operating system. Now, deep inside...

Oct 28 232 13

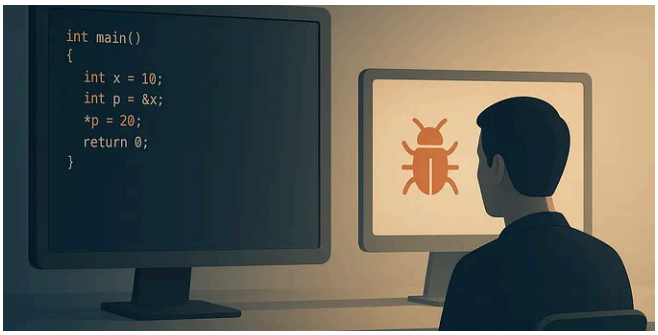


Dax

Scrum Masters Are the Most Useless Role in Tech

I once worked on a team where the Scrum Master's biggest contribution was reminding...

Aug 23 510 63



Dax

How Zig Found a Bug That GCC Ignored for Years

The night a "toy compiler" embarrassed a titan

Sep 18 434 15

See all from Dax

Recommended from Medium

The CS Engineer

Forget JSON — The Future of Fast APIs Is Already Here

JSON is killing throughput and your developer patience.

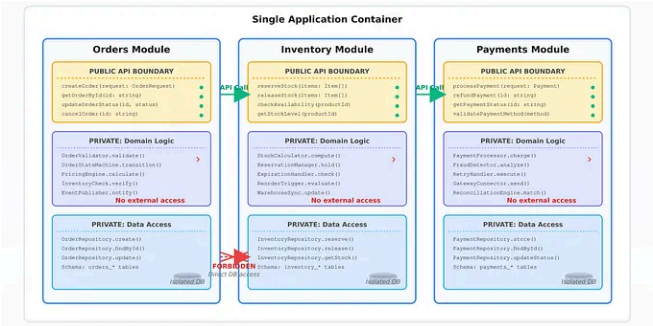
Oct 30 217 10

In Medialesson by Marius Schröder

JSON vs TOON — A new era of structured input?

Why structure matters more than ever

Nov 3 739 36



Is It Vritra - SDE I

My Client Asked Why I Charge $180/Hour When Bubble.io is Fre...

So my client sends me this Slack message —

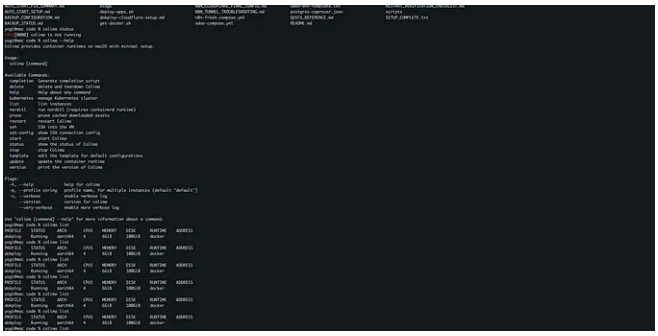
Oct 31 839 15

The Atomic Architect

Architecture Patterns That Actually Scale In 2025: The Only...

Last month, a company I advised shut down.

Nov 7 808 27



Yogeshwar Tanwar

The $699 Mac Mini Server That Replaced My AWS Bill: The Devil's...

Three months in, and I'm still waiting for the disaster. The catastrophic failure that would...

Zudonu Osomudeya

Don't Containerize That Database, Just Don't

The reality check every DevOps team needs before diving into the deep end

Oct 13 391 10 Oct 26 243 16

See more recommendations