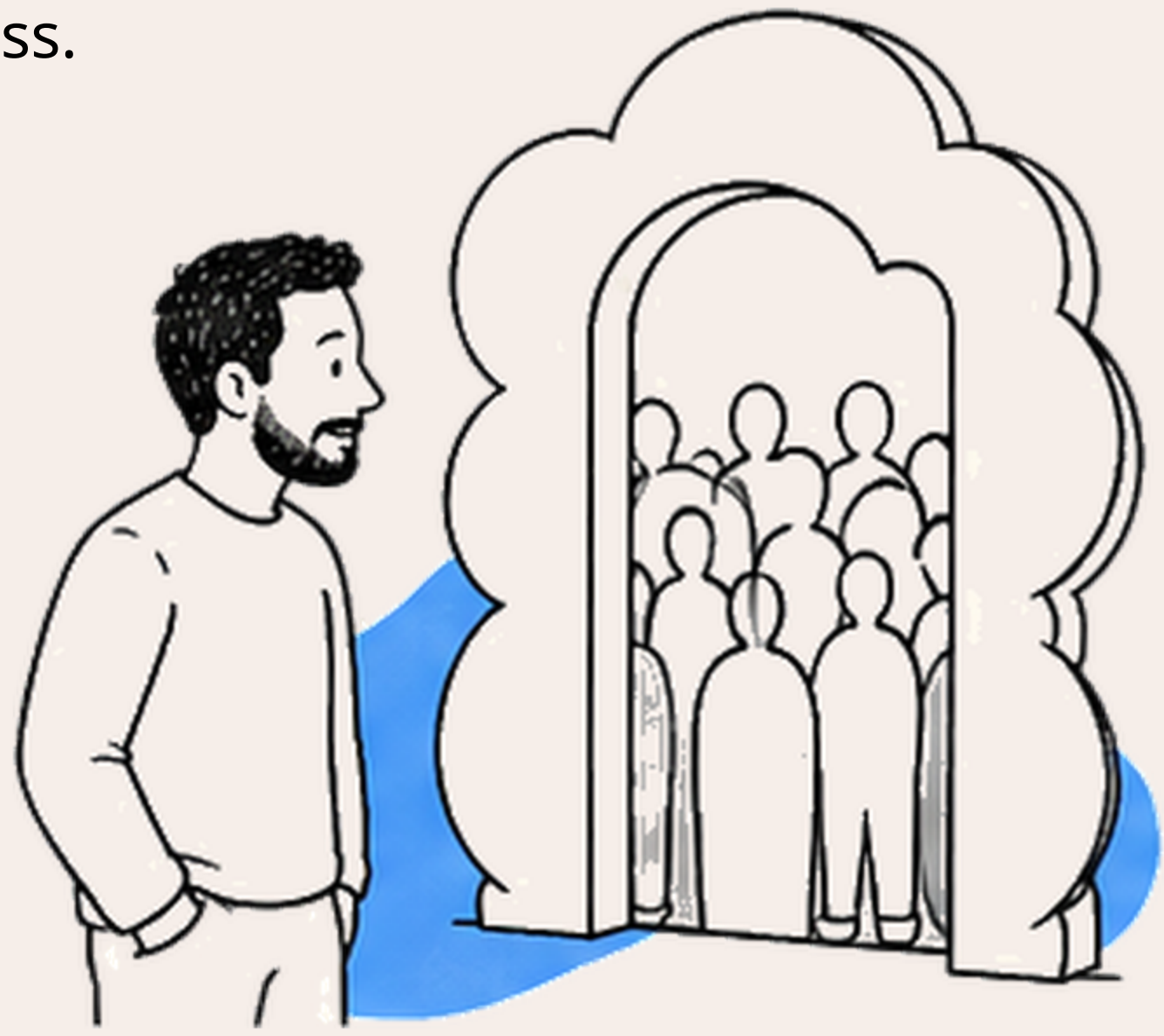
Be stubborn about the impact, not the solution.

Swipe ›››

We boldly challenged the status quo.

We challenged one assumption about how SaaS teams test software.

The result changed our cost structure, developer experience, delivery throughput, and AI readiness.



Shared cloud environments made growth expensive and slow.

Our product runs in the cloud. But as the team grew, shared test environments became an expensive bottleneck

More engineers meant:

- more contention
- more waiting
- lower throughput
- higher costs

Ephemeral environments moved only the scaling problem.

We developed ephemeral environments to isolate every change, but cloud compute still grows with engineering activity that grew with agentic engineering even more.

> Why test on an isolated machine in the cloud when you can test on an isolated machine locally?

Our cloud-native architecture made local testing look impossible.

Running the full product meant fitting all this on one laptop:

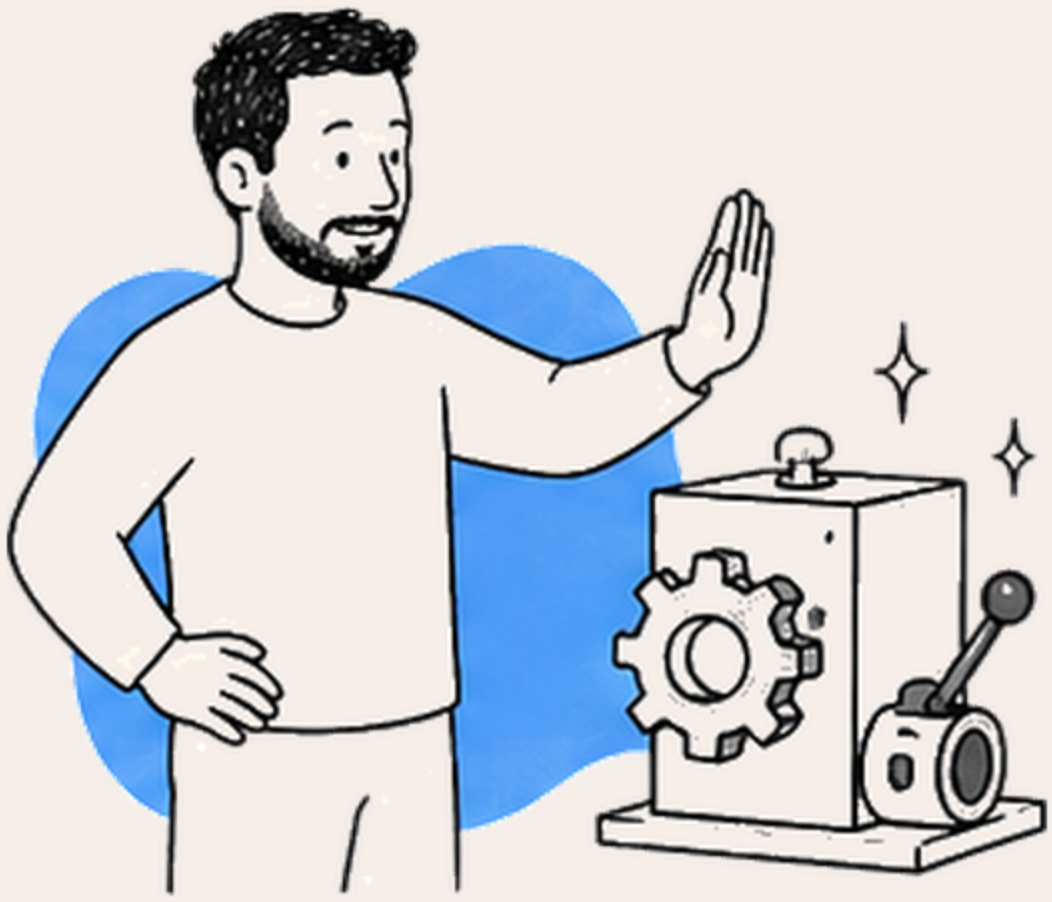
Kubernetes, many services, infrastructure dependencies, external integrations, and enough compute to run everything.

There are no big problems, just a lot of little problems.

.. as famously Henry Ford said.

We broke the problem into individual impediments and worked through them one by one.

- Some turned out to be assumptions.
- Others required engineering.
- A few were not worth solving.



Stop when further work optimizes the solution, not the impact.

It is easy to fall in love with the perfect solution instead of staying focused on the problem.

I regularly challenged myself asking:

> Would I sponsor the remaining work as a new initiative right now?

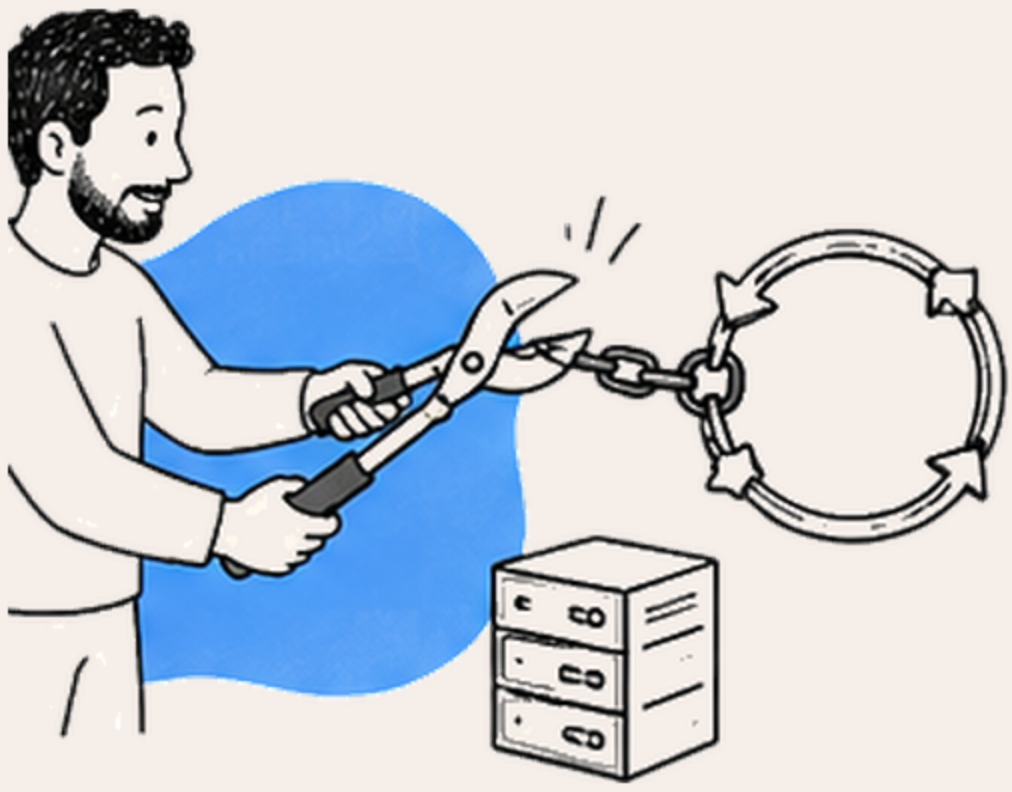
We paused when the answer was “No.”

The cloud savings paid for the new laptops in one month.

Local testing required expensive hardware upgrades for our engineers.

At first, that investment was difficult to justify. Then we compared the one-off hardware costs with recurring cloud infrastructure costs.

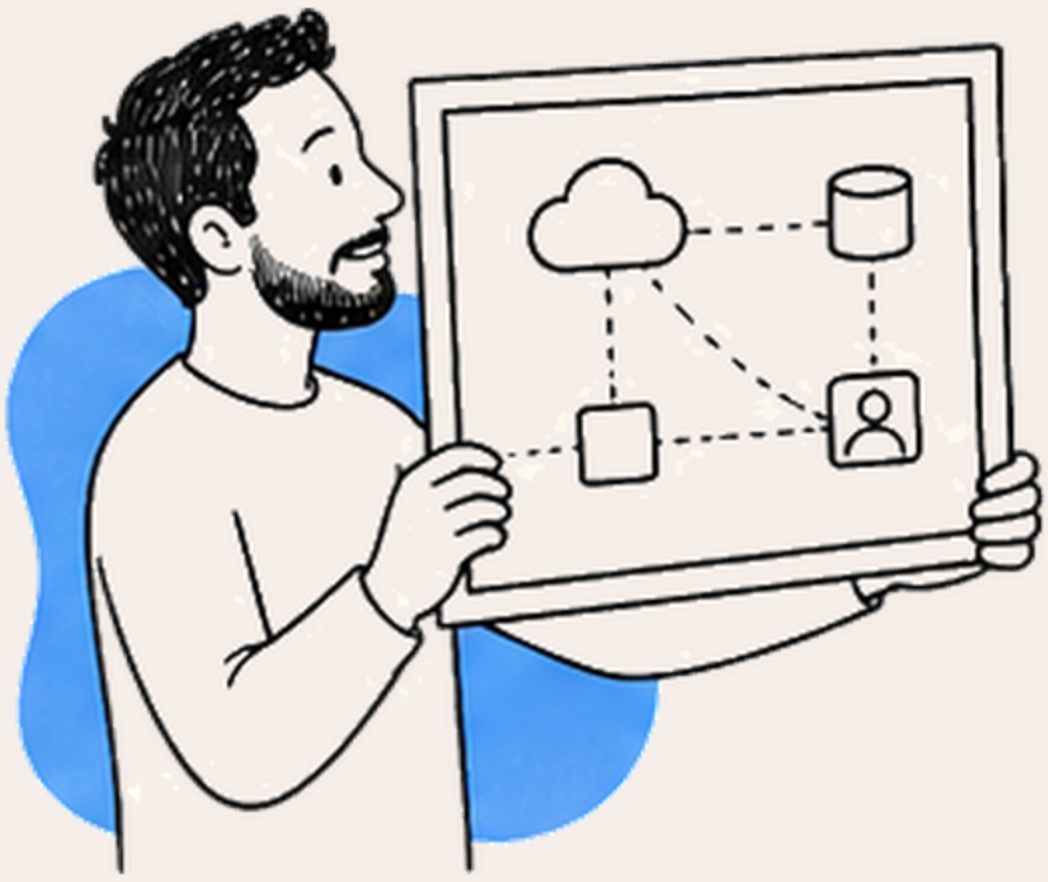
Spending more in one place can make the whole system cheaper.



Saving 50% on infra turned out being only secondary benefit

The effects compounded across the development loop:

- no queueing for testing
- more autonomy for agents
- faster inner loop feedback
- higher throughput



The biggest opportunities sit between the parts we optimize separately.

We could have kept tweaking the existing system with more environments, cheaper compute, and similar improvements.

Instead, we stepped back, treated it as one system, and questioned its assumptions from first principles.

Hello, I’m MJ.

I’m an engineering leader and mentor hyperfocused on developing people and building organizational capabilities. I draw on 17 years of leadership experience across companies of all sizes.

I’m always happy to share my experience.
Get in touch!