The $300 enterprise lab

Cameron Khorsandi & Derek Bowdle

Democratizing infrastructure skills with Raspberry Pis & AI agents.

Bill shock

Monday morning. The inbox is open.

This is the industry's ultimate deterrent. It stops juniors from exploring. It makes engineers tiptoe around architecture, provisioning the bare minimum because every button might cost money.

THE ULTIMATE DETERRENT

The fear that kills curiosity.

Training shouldn't require you to gamble next month's rent on a misclicked checkbox.

Learning on production is flying a 737

Public cloud is a commercial airliner with paying passengers.

You can't kill the left engine to feel how the airframe behaves. Engineering demands you cut the engine.

THE SANDBOX

Failure is a feature, not a bug.

A flight simulator lets you stall, spin, and crash — over and over — until the recovery is muscle memory. That's what this lab is.

Break things on purpose.

Two barriers stop everyone

BARRIER 01

The Cloud Bill Trap

Free Tiers are marketing funnels. The moment you build anything multi-node or highly available, the paywall hits. Skill development stops where the credit card begins.

BARRIER 02

The Mentorship Gap

Senior SRE time is the most expensive resource in IT. Juniors get cryptic Slack replies instead of pair sessions. Most learning happens alone, at 2am, in front of red logs.

Bill of materials

Everything that fits on a desk and costs less than a flight.

3x Raspberry Pi 5

Compute / control plane nodes — quorum of three

$240

NVMe SSDs + Hats

Fast distributed storage via Longhorn

$90

5-port unmanaged switch

Hardwired gigabit — no flaky Wi-Fi mesh

$10

MacBook Air M4

External AI brain — runs local LLM agent

(BYO)

CAPEX TOTAL ~ $325 + Pis + extras

ONE-TIME

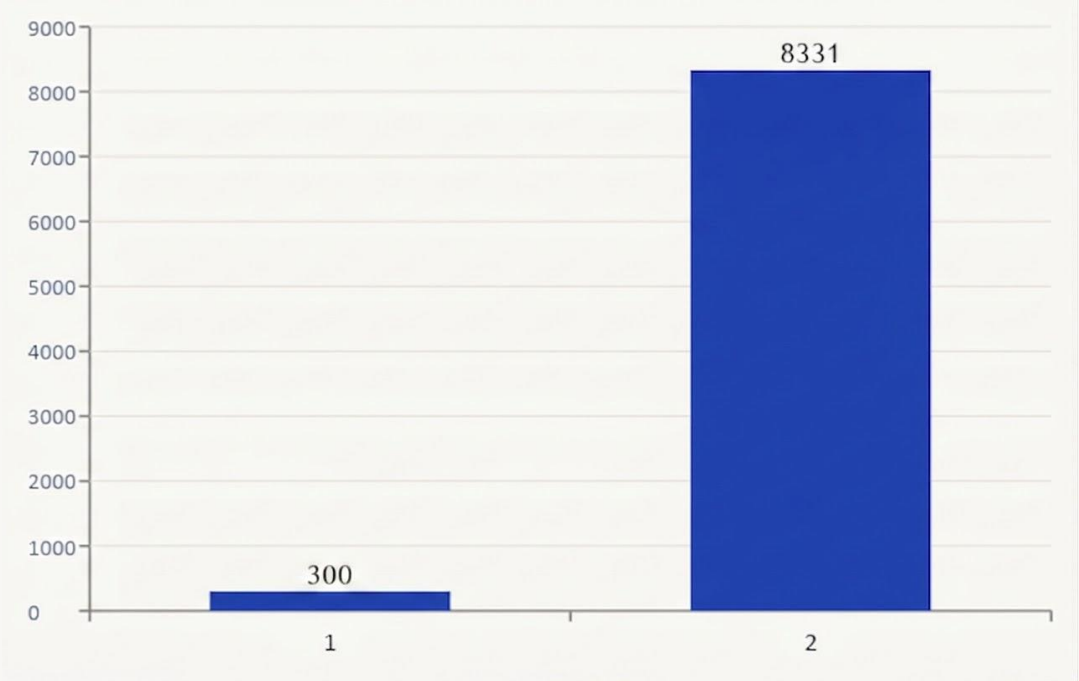
~$300

Sits on your desk forever.

No metered minutes.
No egress charges.
No surprises.

The math is not close

Total cost of ownership over a single year.



AWS / YEAR

$8,331

EKS · 3x m5.large · gateway · egress

PI LAB / YEAR 1

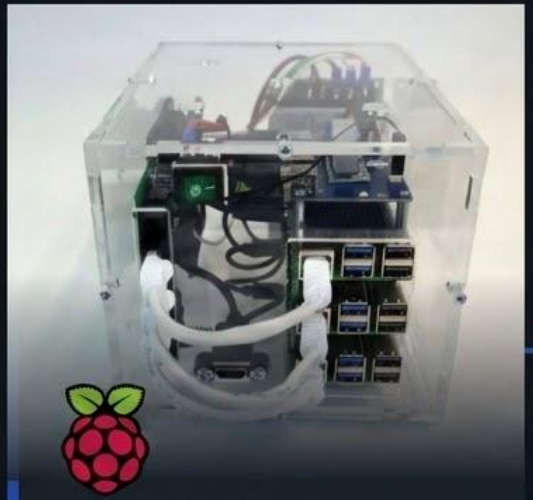
$300

Then ~$0 in years 2, 3, 4…

That is a 13x multiple — every year — for the same topology.

The answer is Architecture Parity.

We aren't trying to match the raw power of a hyperscale data center; we are matching the topology.

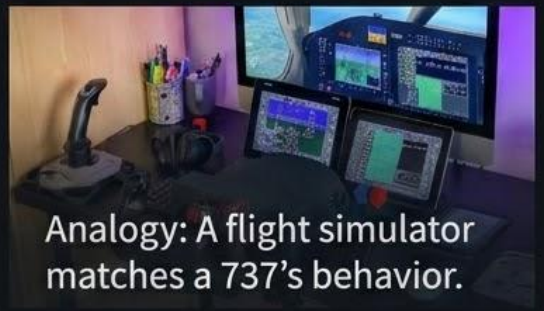


Matching the topology, not the power.

The same operational logic.

Load Balancing · Distributed Storage · GitOps Workflow.

EKS · Multi-region · Highly available.



Analogy: A flight simulator matches a 737's behavior.

The open-source engine

Six components. Zero license fees. Full architectural parity.

k3s

Lightweight Kubernetes — three control-plane nodes for proper quorum

MetalLB

Bare-metal load balancer — gives your cluster real LoadBalancer IPs

Longhorn

Self-hosted distributed storage — block volumes mirrored across nodes

Forgejo

Self-hosted Git — your private GitHub running on the cluster itself

OpenCode + LM Studio

Local AI agent + local LLM host — your private senior SRE

kube-ops-view

Real-time graphical heartbeat — every pod, every node, every failure

Your local AI senior architect

Why local matters more than which model.

Air-gapped privacy

Proprietary logs, kubeconfig secrets, API keys — none of it leaves your laptop. No prompt-injection risk from someone else's RAG pipeline.

Zero token cost

A single crash-loop debug session can burn thousands of tokens. Locally, it's free. Iterate as long as the lesson takes — not as long as the budget allows.

Offline reliability

Hotel Wi-Fi, conference network, a flaky home connection — none of it matters. The model and the cluster are both on your desk.

The mentorship workflow

A recipe for how to collaborate with your coding agent

01

Role setting

Prompt the agent as a senior platform architect with Pi-hardware context.

02

Recipe design

Ask for a YAML plan grounded in your cluster — not someone's blog post.

03

Execution

You apply the manifest. The human stays in the loop, hands on the keyboard.

04

Feedback loop

Pipe raw kubectl logs straight back into the agent — full context, no paraphrase.

05

Debug & verify

Cryptic error streams become plain-English fixes and a written rationale.

Every fix is questioned. Every error becomes a lesson. The loop forces engagement — not copy-paste.

Live demo

Five things, in five minutes, on three Pis.

01 Summarize the cluster

Agent inspects topology and explains it back in plain English.

02 Ping the hosted service

Confirm the production-style website is live on a MetalLB IP.

03 Open the dashboard

kube-ops-view: every pod and node, animated in real time.

04 Pull up Forgejo + the site

Self-hosted Git and the deployed app — both running on the lab.

05 Pull the plug

Kill a node mid-demo. Let the agent walk us through recovery.

Chaos engineering: the plug-pull test

Yank the power cord. Mid-traffic.

While the cluster is serving live traffic, we physically pull the plug on one of the three nodes. k3s panics. Quorum wobbles. Pods get rescheduled. The website either stays up — or it doesn't.

In a corporate job, this gets you fired.

On a $300 lab, it's the graduation ceremony. You stop being afraid of the terminal because you've already seen the worst case sitting on your desk.

FAILURE CASCADE

- → Node goes NotReady
- → Quorum drops to 2/3
- → Longhorn reattaches PVs
- → MetalLB re-elects speaker
- → Pods rescheduled in seconds
- → AI agent narrates the recovery

CLOSING THOUGHT

The gatekeepers of infrastructure have been bypassed.

You don't need a corporate mistake budget. You need a switch, three Pis, a local model, and the courage to pull the plug.

Go build your lab. Pull the plug. See what happens.