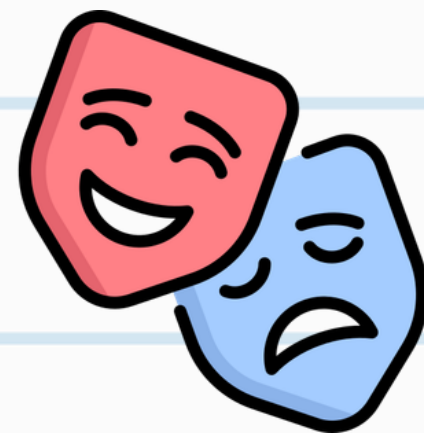
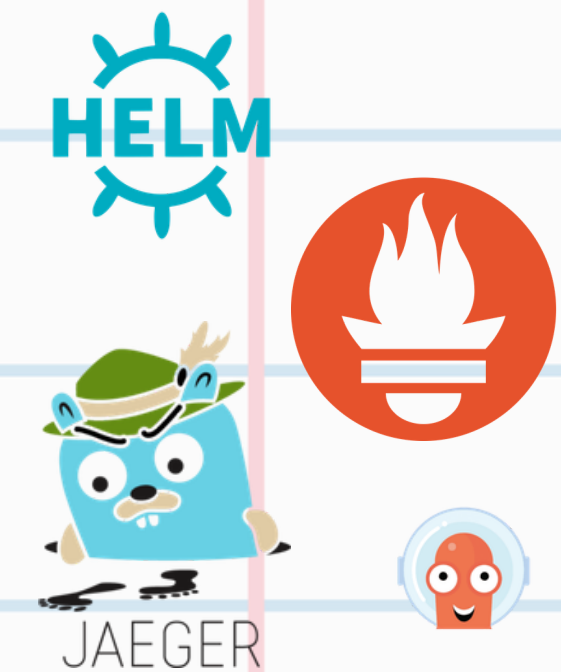




Kubernetes Native $\neq$ Cloud Native

Avoiding Architectural Theatre



Another exaggerated introduction slide

Senior TPM at **GROUPON**

Create Content on  

Like Sharing my tokens 

Love Dogs



is windyyy....yy

GROUPON[®]

The best things in life happen **offline**.

In a world engineered to keep people scrolling, **Groupon** takes the opposite position. Human experience, the kind that happens in the physical world, with real people, in real places, is **irreplaceable**.

Disclaimer

The views and opinions expressed in this presentation are solely those of the speaker and do not necessarily reflect the official position or policies of **Groupon, Inc. or any of its affiliates**. All architectural examples, scoring methodologies, and demo applications (including the Pac-Man variants) were created specifically for this talk to illustrate architectural differences — they are intentional simplifications, not production code.



Prerit Munjal • You
Platform Engineering Leader • Senior TPM at Groupon • ex-CTO...
[View my portfolio](#)
1yr •

Met the OG of Kubernetes at [The Linux Foundation #OSSummit](#) North America 2024: Kelsey Hightower

Got a new perspective on the Cloud-native ecosystem.

For all the folks reading this, it would be great if you could comment: **Why are you learning Kubernetes?**

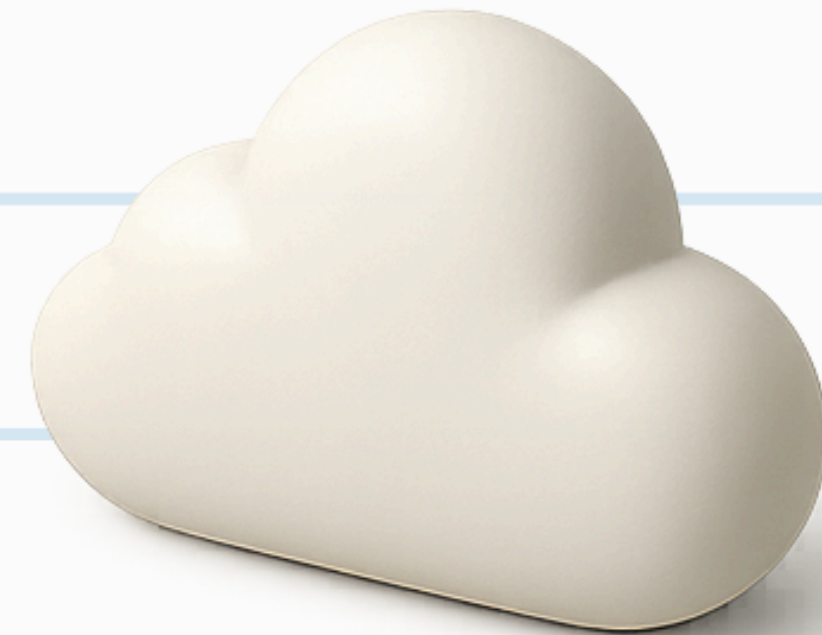
[#kubernetes](#) [#cloud](#) [#devops](#)



What is Cloud-Native?

Let's not define it. Let's question it.

CNCF says "Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments."



- **Externalized configuration:** config lives in the environment (env vars, config services), never hardcoded. The same artifact runs in dev, staging, and prod without a rebuild.
- **Observable by design:** structured logs, distributed traces, and metrics are built into the app from day one, not bolted on after an outage. You define SLOs and measure against them.
- **Graceful failure as a feature:** circuit breakers, retries with backoff, bulkheads, and health probes are part of the code. The app expects dependencies to fail and degrades instead of crashing.
- **Stateless and disposable:** processes can be killed and restarted at any time with zero data loss. State lives in backing services, not in local memory or disk.
- **Loosely coupled, independently deployable:** services own their own data and lifecycle. You can ship one service without coordinating a fleet-wide release.

What is Kubernetes-Native?

Your app speaks Kubernetes. That's it.



- **Custom Resources & Operators:** your app extends the Kubernetes API surface with CRDs and uses controllers/operators to reconcile desired state, turning K8s into a domain-specific control plane.
- **Pod lifecycle awareness:** probes (liveness, readiness, startup), init containers, sidecars, and preStop hooks are wired into the app. It cooperates with the scheduler, not just tolerates it.
- **Declarative everything:** the entire application topology (networking, scaling, secrets, RBAC, ingress) is expressed as versioned YAML manifests, managed through GitOps and reconciliation loops.
- **Platform-coupled by design:** the app leverages K8s primitives like ConfigMap hot-reload, service discovery via DNS, and leader election through the API. It assumes K8s is the runtime.
- **The tradeoff:** deep K8s integration gives you powerful operational leverage, but locks you to the platform. Portability is sacrificed for orchestrator-native capabilities.

LATER

aws

Azure



RESET IT ALL?
LOL IDK

? Helm?
? Istio?

? Istio?



Kubernetes?

Microservices ?

??

AWS, Azure, Google Cloud

? NODES ??

API Gateway ???

Auth?

Auth?

WTF?

Serverless
Lite™

Random
LAMDDAS?

Auth??

WEIRD CONTAINERS!?

Cloud? IDK!

?? DAIBKES?
SomeBrands?

Growlfore

CRON?

? CRON ?

??? CLOUD STORAGE?

?? NTIRES?

ConfigMap? Secrets???

?? DATABASES?
RDS?



THE PLAYBILL

What Cloud-Native Sold Us ...for a decade

Portability Write once, run anywhere (sound familiar, Java folks?)

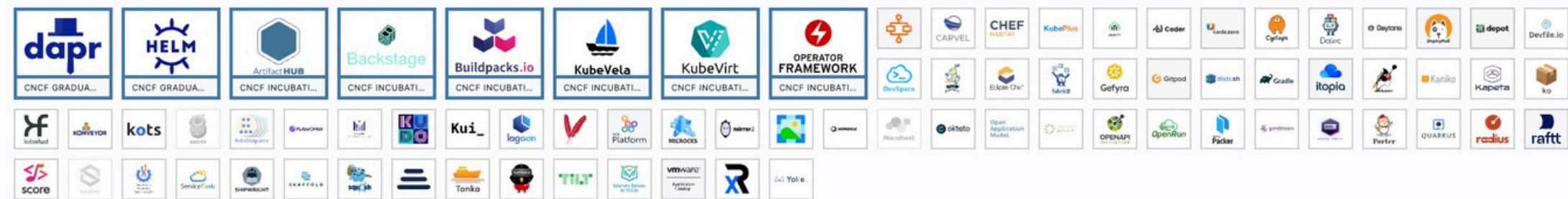
Scalability Infinite scale at the push of a button

Resilience Self-healing everything

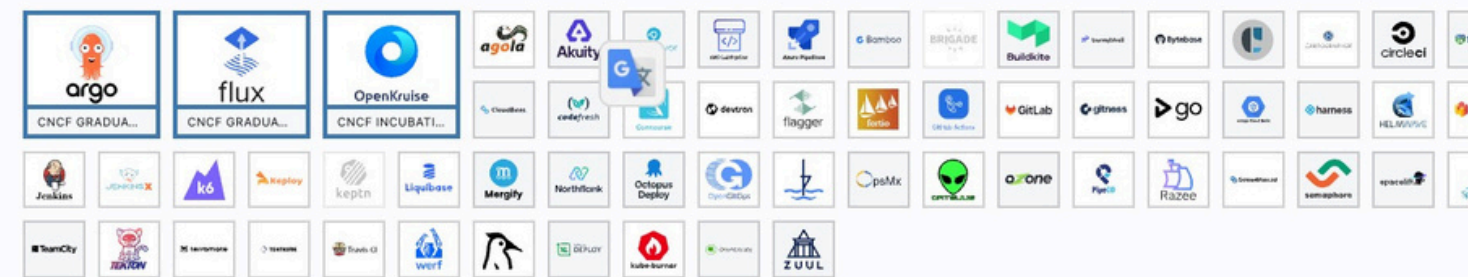
Developer Velocity Ship faster, break less

CNCF Landscape: 50 → 1,500+ projects. We bought the brochure

Application Definition & Image Build



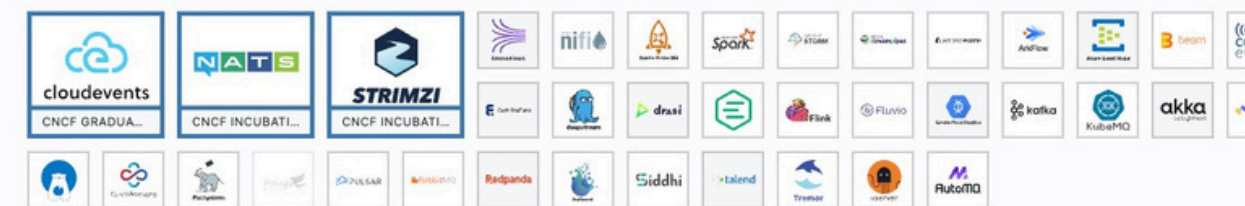
Continuous Integration & Delivery



Database



Streaming & Messaging



Scheduling & Orchestration



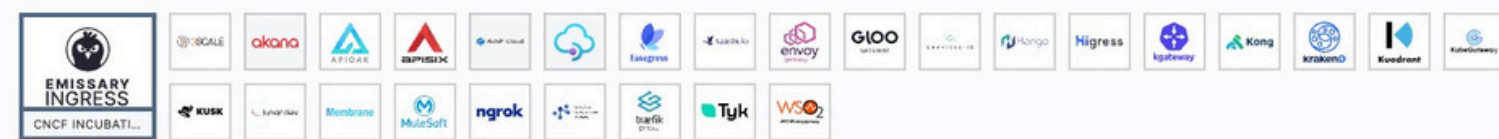
Service Proxy



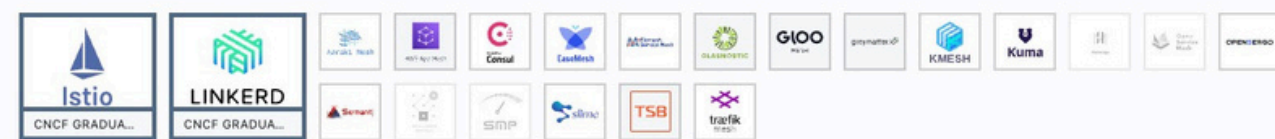
Remote Procedure Call



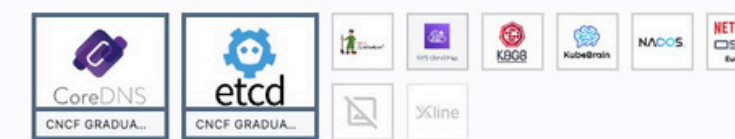
API Gateway



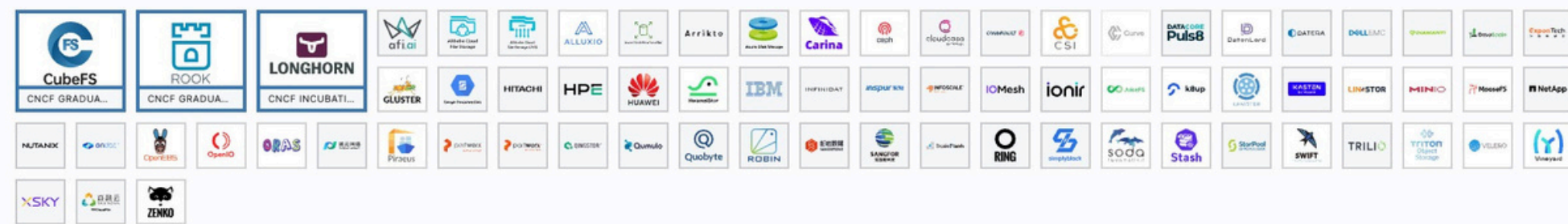
Service Mesh



Coordination & Service Discovery



Cloud Native Storage



Container Runtime



Cloud Native Network



Security & Compliance



Container Regi...



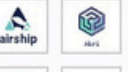
Automation & Configuration

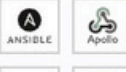

KubeEdge
CNCF GRADUA...

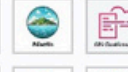

Cloud Custodian
CNCF INCUBATI...

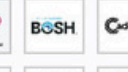

CNCF INCUBATI...


OpenYurt
CNCF INCUBATI...


airship


aks


ANSIBLE


Apollo

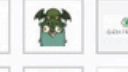

Arctic


ARK Distribution

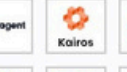

BOSH

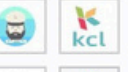

Cadence

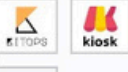

CDK8s


CFEngine

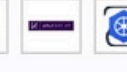

CHEF


CLOUDIFY

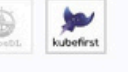

Cooler


CockroachDB


CROSSIN


CROSSIN

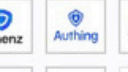

CROSSIN

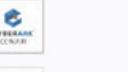

CROSSIN


CROSSIN


CROSSIN


CROSSIN


CROSSIN


CROSSIN


CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

CROSSIN

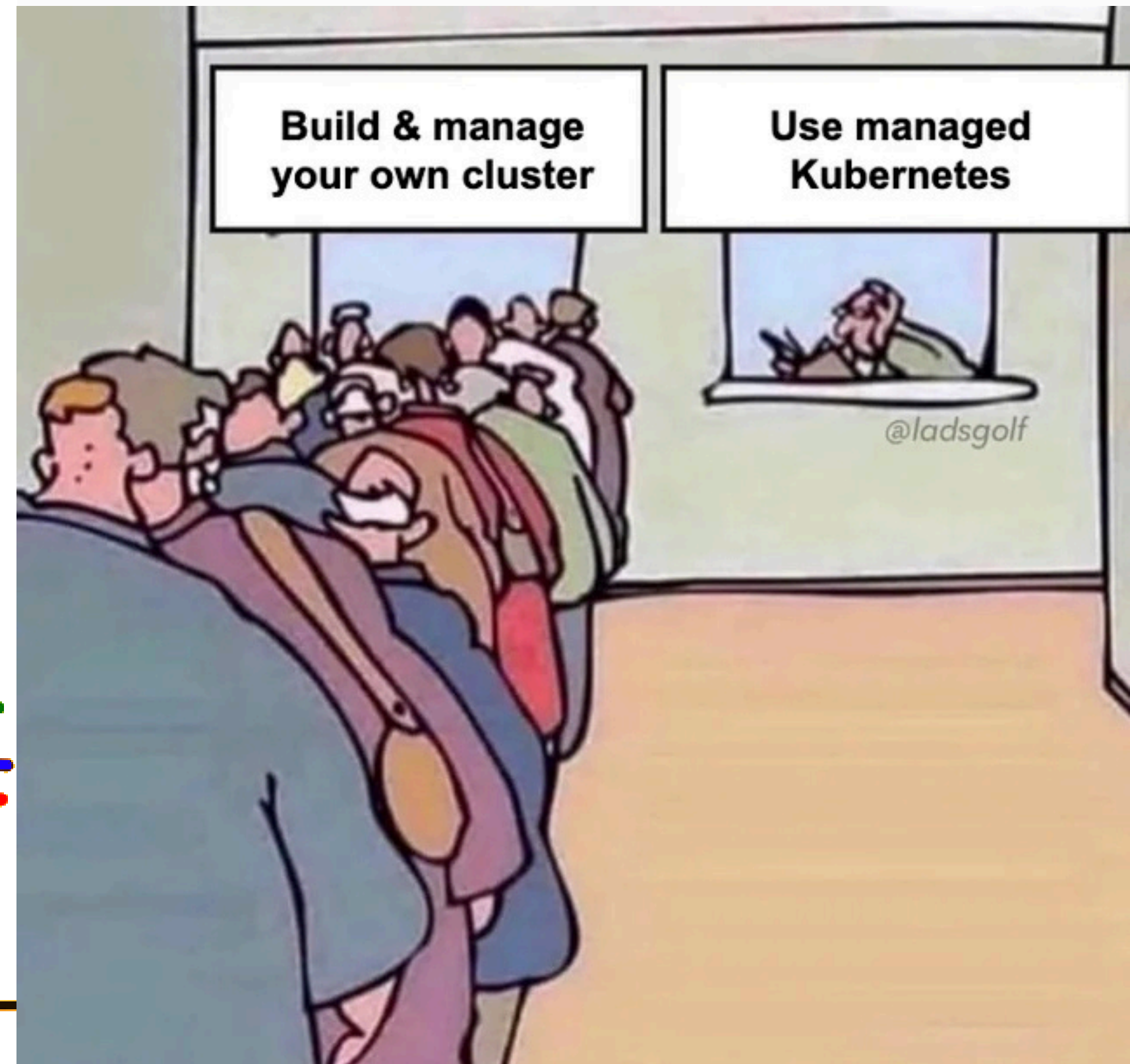
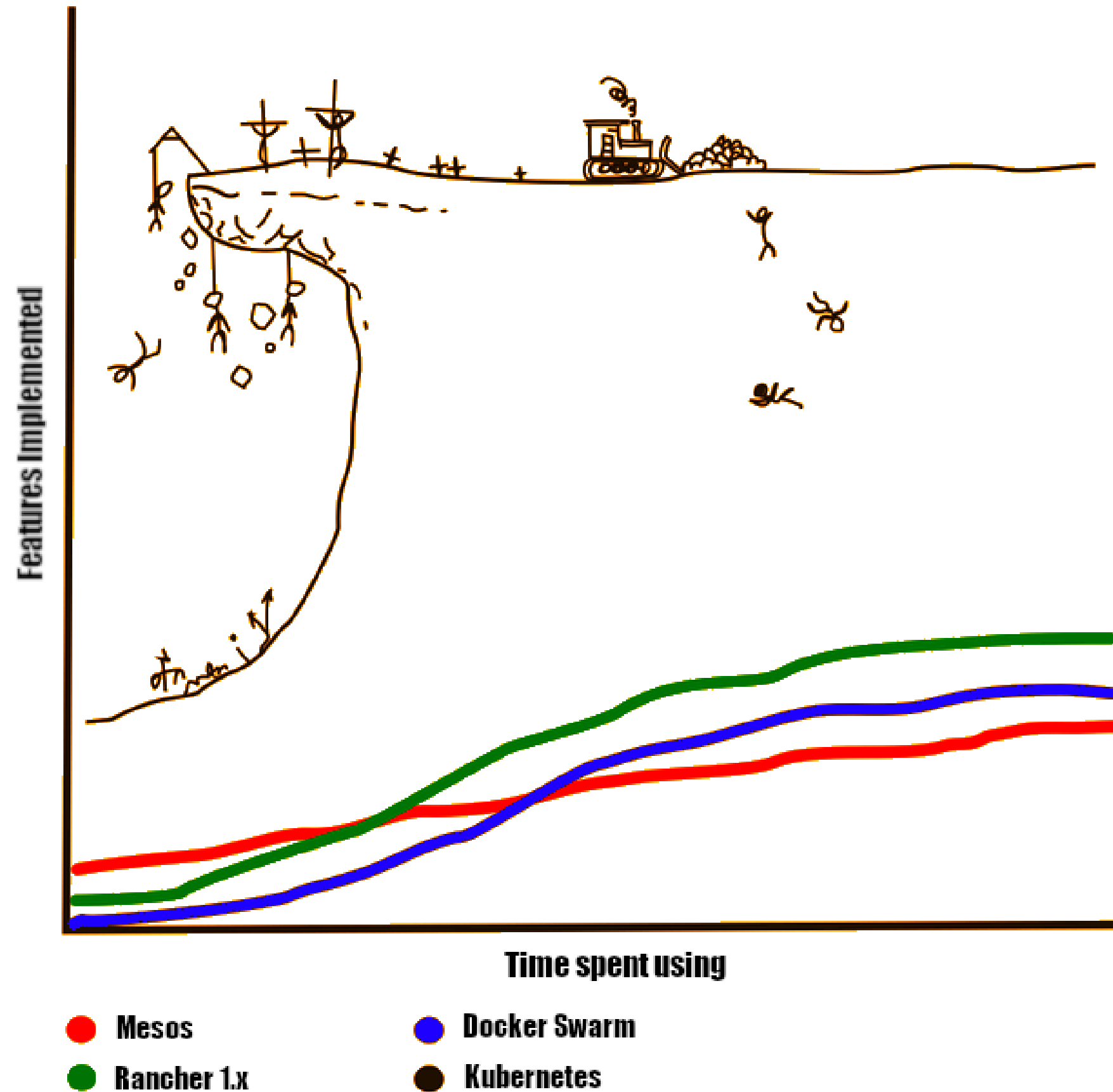
CROSSIN

CROSSIN

CROSSIN

CROSSIN

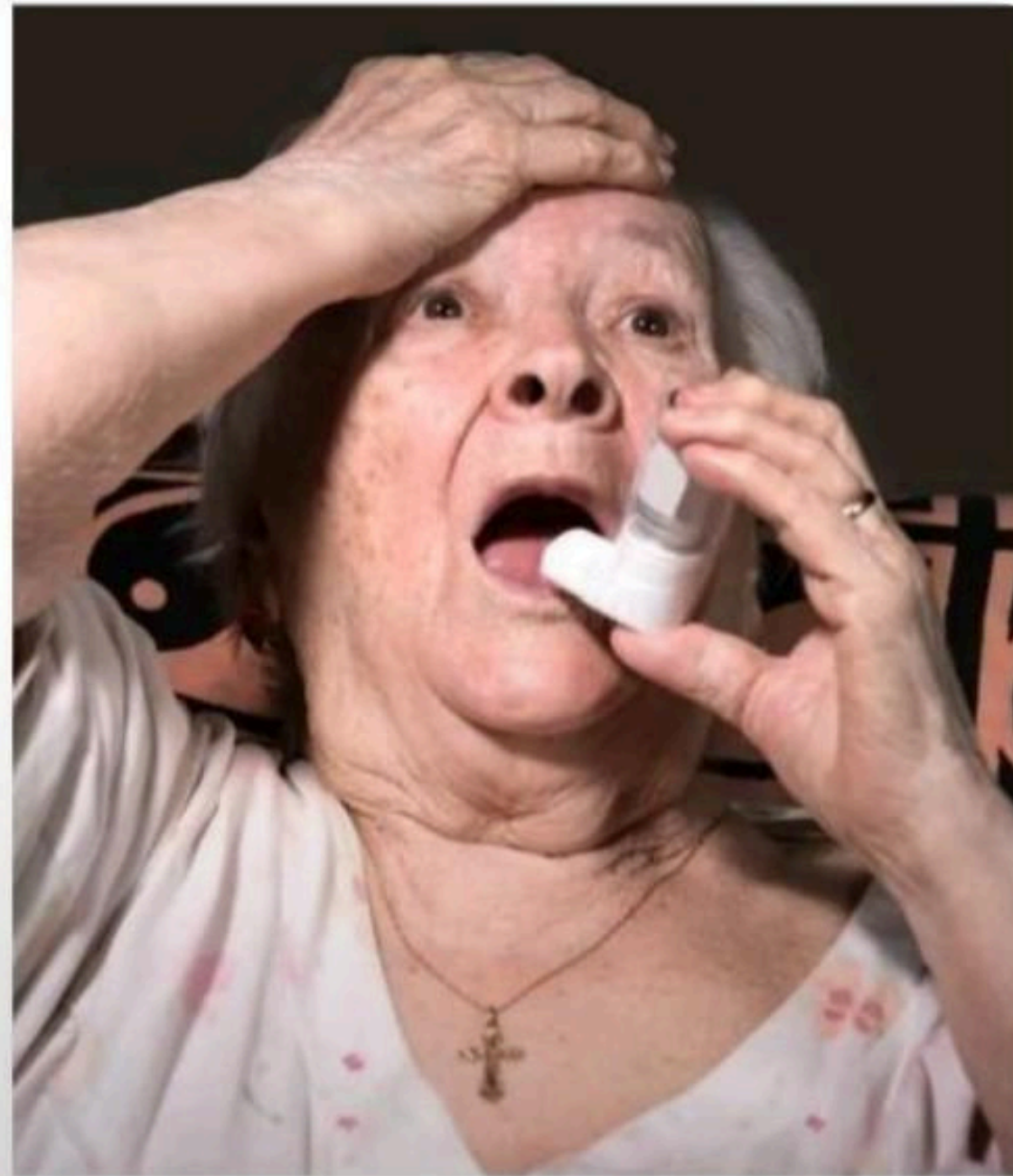
Learning curves of some Container Orchestration Engines



**Jennifer
Aniston at
51 years old**



**Kubernetes
Engineers
in their 30s**





Start with a



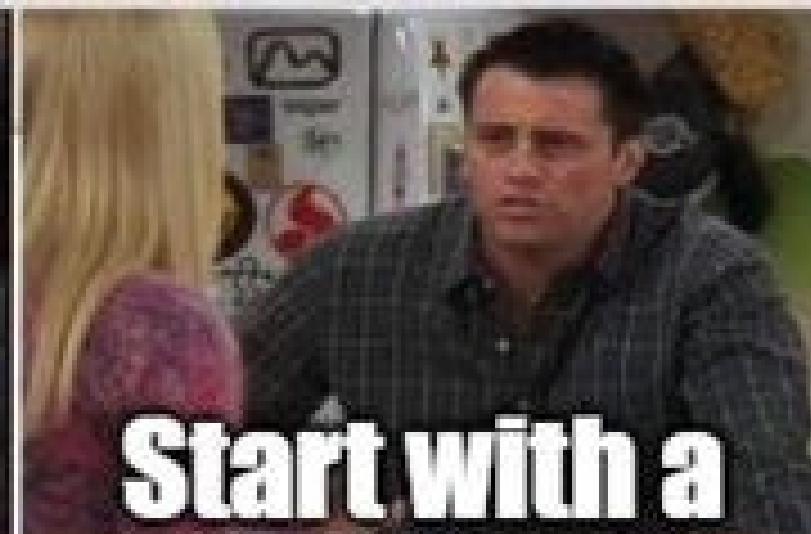
Simple



**Kubernetes
cluster**



**Start with a simple
Kubernetes
cluster!**



Start with a



Simple



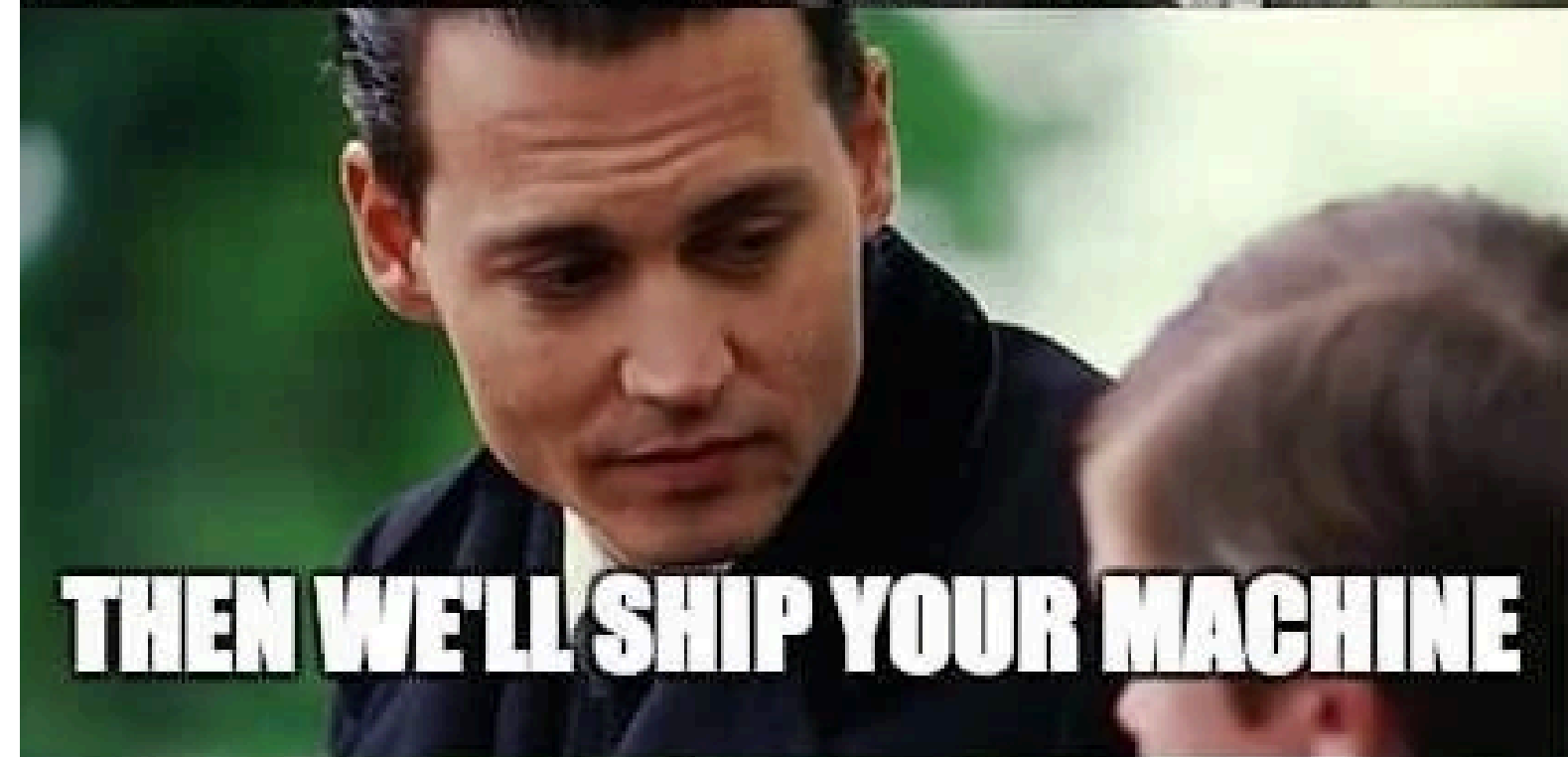
**Kubernetes
cluster**



**Helm, Istio, Jaeger
Terraform, Prometheus,
Envoy, Argo CD,
Cert-Manager, Grafana**



IT WORKS ON MY MACHINE



THEN WE'LL SHIP YOUR MACHINE



AND THAT IS HOW DOCKER WAS BORN

What We Imagined

- ✓ Self-healing infrastructure
- ✓ Portable across clouds
- ✓ Observable by default
- ✓ Developer self-service
- ✓ Fast, safe deployments

What We Got

- ✗ YAML hell at 3am
- ✗ Vendor lock-in with extra steps
- ✗ Prometheus: installed, unread
- ✗ Platform team as bottleneck
- ✗ 20-minute deploy pipelines

Which side of the stage is your team on?

82%

run K8s
in production
CNCF 2025

25%

are truly
cloud-native

What we really want

Your CEO doesn't care about your CRDs. They care if the app is up.

01

Availability

Is the app up? Can customers transact?

02

Scalability

Can it handle Black Friday?

03

Fast Deploys

Can we ship a fix in under 30 minutes?

04

Low Cost

Are we paying for value or vanity?

ACT 1: STARTUP

3 engineers, 2 services
Running: K8s + Istio + ArgoCD
+ OPA + custom operators
"Resume-driven development"

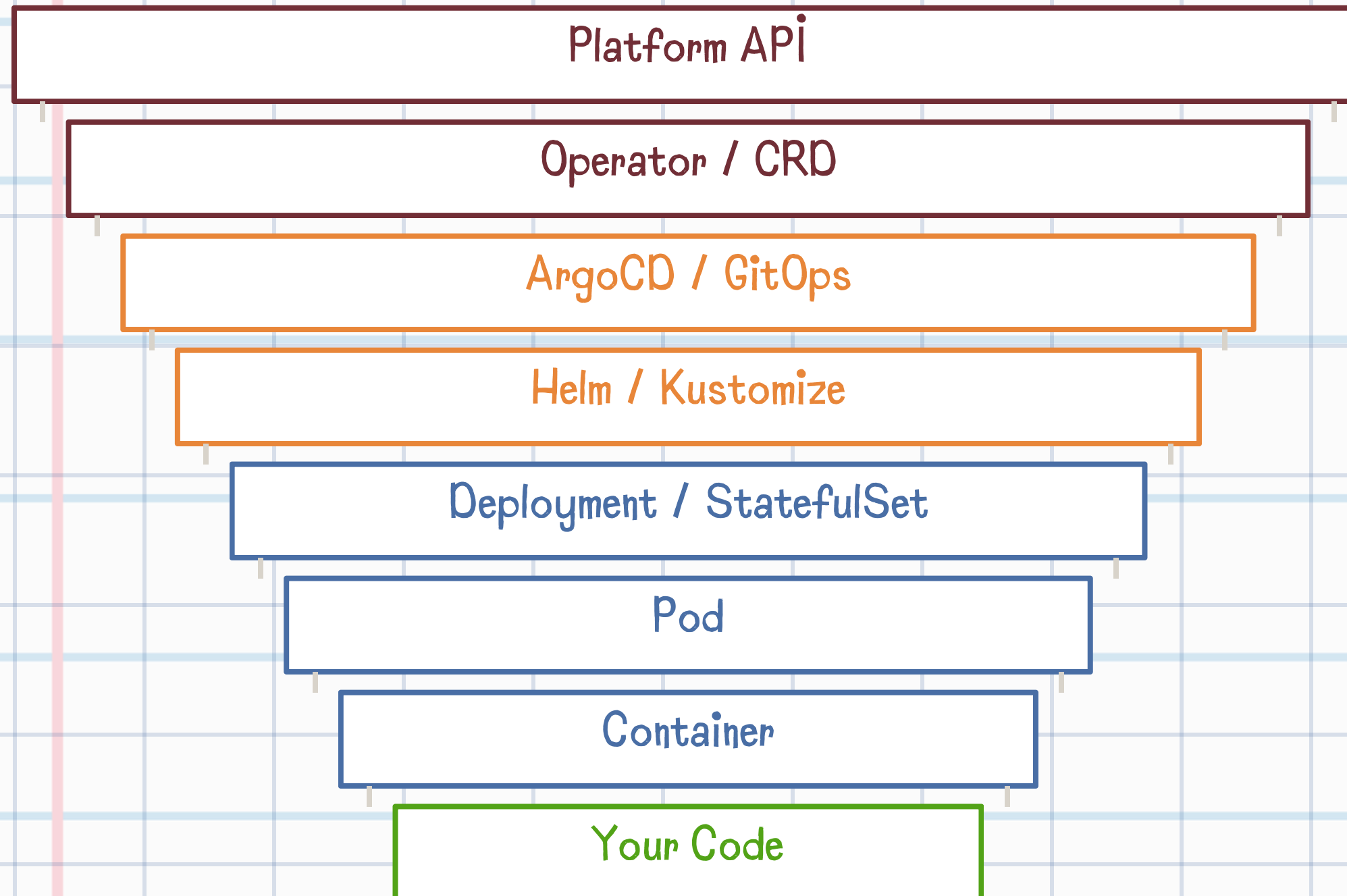
ACT 2: ENTERPRISE

200 engineers, 500 services
Still can't deploy on Friday
"Different problems,
same theater"

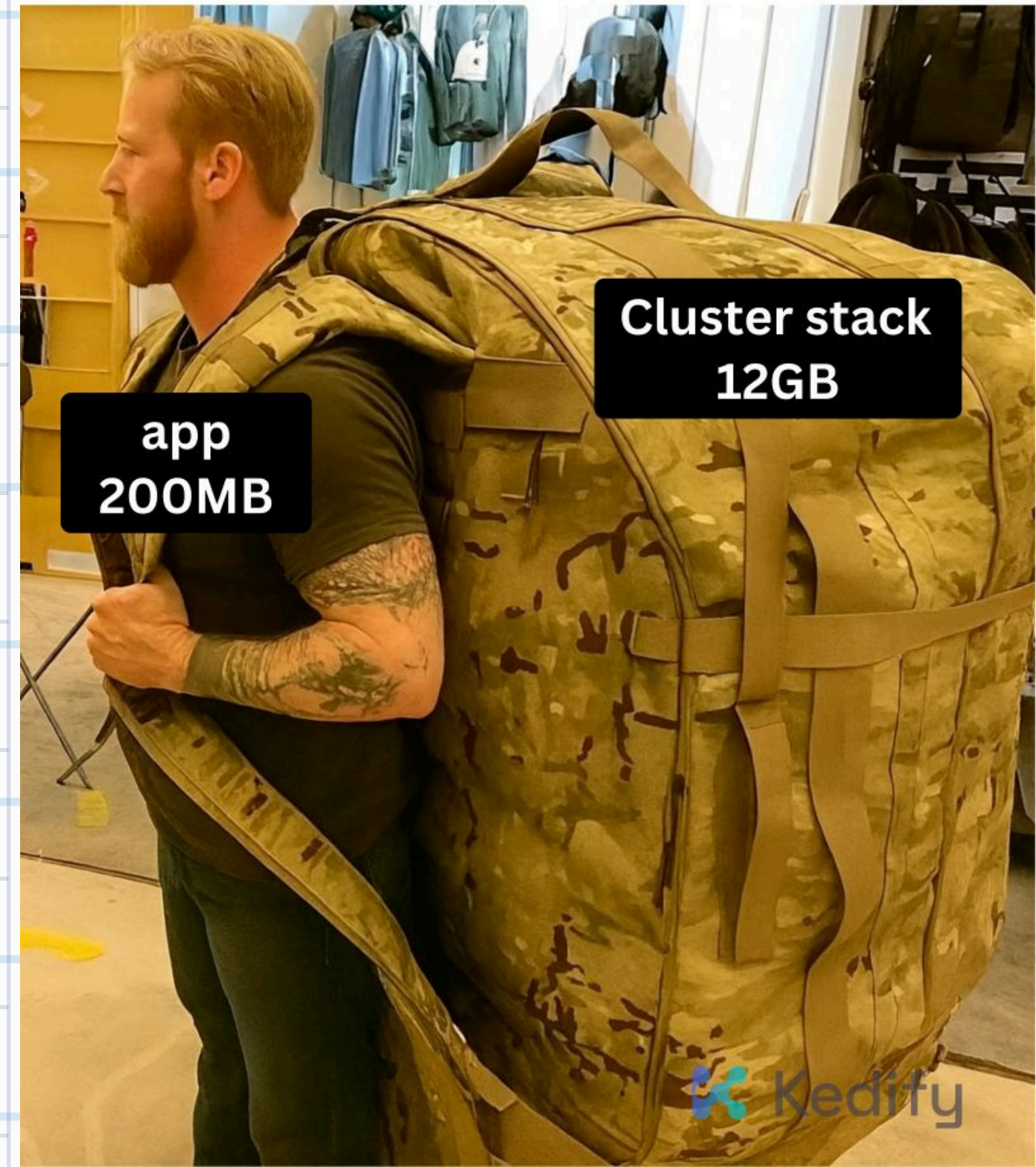
Each layer was supposed to simplify. Together they terrify.

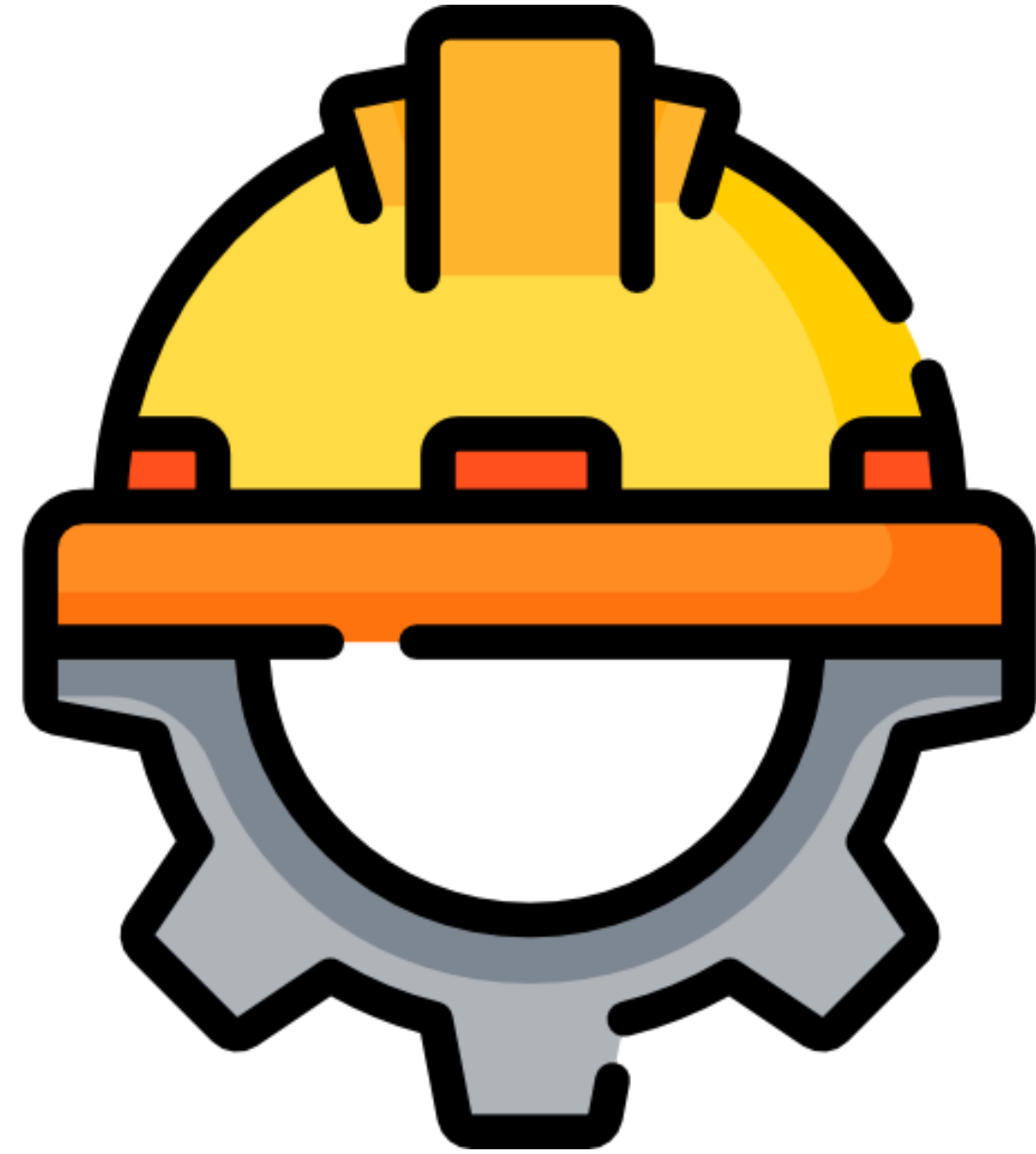
The Set Design

Layers of Abstraction



Kubernetes Engineering





DX → **KX**

Abstractions

The Monologue

Building One App on K8s — every vertical you'll touch:

Code

Dockerfile

Base Images

XXXXXX

Helm Charts

Probes

Secrets Mgmt

RBAC

Ingress & TLS

Observability

Identity & Auth

Cost Allocation

Failure Domains

Resource Tuning

YAML Sprawl

Container Registry

We solved deployment. We created 15 new problems

Almost Done!

Learning
container

Almost!

Docker

Oh yes!

Understand
kubernetes

Oh shit!

Cloud
Native

Breaking Changes & The Upgrade Tax

K8s Deprecation Cycles

API removals every 3 releases. Your YAML expires.

CRD Version Migrations

v1alpha → v1beta → v1. Each step can break installs.

Helm Chart Drift

Upstream charts break silently. Pin a version? Miss security patches.

Buildpacks & Base Images

Build today, CVE tomorrow. Rebuild everything, test nothing.

Your platform is always 1 upgrade away from broken.

**I JUST WROTE A KUBERNETES
BLOG**

AAAAAND IT'S OUT OF DATE

The Illusion

Portability — ever actually done a migration?

Migrating a K8s app between clouds $\approx$ same effort as switching to a completely different orchestrator. — McKinsey

IAM & Identity

AWS roles $\neq$ GCP IAM $\neq$ Azure RBAC

Managed Databases

Cloud Spanner $\neq$ Cosmos DB $\neq$ Aurora

Networking

ALB $\neq$ Cloud Load Balancer $\neq$ App Gateway

K8s gave us the illusion of portability. The invoice says otherwise.

Did somebody say Seeeeev1.



~80%

of K8s incidents are caused by operational complexity,
not infrastructure failure

CASE STUDY: MONZO OUTAGE

K8s bug + etcd reconfiguration + Linkerd incompatibility = cascading failure.
"Self-healing" systems that need a human to understand 4 abstraction layers at 3am.

More layers = more failure modes = exponential complexity

The Comedy

Developer Productivity: The Broken Promise

S O L V E D

"Works on my machine" → Containers

Picked 2 problems from devs
Turned them into 12 for platform
eng

C R E A T E D

"Works on my cluster... sometimes"

CNCF 2024: #1 challenge shifted from technical → organizational

The Box Office

ROI: Are We Getting Value?

Infrastructure

Compute, networking, storage, managed K8s fees

Platform Team

Engineers maintaining the platform, not building product

Training

Upskilling devs on K8s, Helm, observability tooling

Incident Response

Time debugging platform vs. application issues

Cognitive Load

Context-switching between 10+ tools per deploy

If your platform team is larger than your feature team, the platform isn't serving the product.

Cut the Props

You Don't Need Excess

Service mesh for 3 services?



HTTP health checks work fine.

Custom operator for a CronJob?



Use the CronJob.

GitOps for a monolith?



CI/CD pipeline is enough.

Custom CRD for config?



ConfigMap + CI pipeline.

Every tool you add is a tool someone has to debug at 3am.

The Script

Principles That Actually Matter

01 **Own Your SLOs**
If you can't define it, you can't deliver it

02 **Contract Your APIs**
Versioned, owned, documented — no exceptions

03 **Observe Everything**
Structured logs, distributed traces, SLOs > uptime %

04 **Fail Gracefully**
Circuit breakers, blast radius, graceful degradation

05 **Keep Blast Radius Small**
If one thing breaks, only one thing should break

These work with or without Kubernetes.

We haven't solved the current complexity — and we're adding non-deterministic workloads on top.

The Sequel

Agentic Infrastructure — coming to a cluster near you

40%+

of agentic AI projects
canceled by 2027
— Gartner

11%

have agentic AI
in production today

Bursty, unpredictable workloads

Agents don't fit 9-to-5 usage patterns

No mature control plane

Scheduling, isolation, cost attribution — all primitive

Non-deterministic behavior

Agents plan, reason, act — and fail unpredictably

Security & isolation

Code-executing agents need kernel-level sandboxing

The Evidence

12-Factor App Compliance • 4 Pac-Man Architectures Scored

38% Non-Cloud
45/120

67% Cloud-Native
80/120

67% K8s-Native
80/120

42% AI-Powered
50/120

12-Factor	Non-Cloud		Cloud		K8s		AI	
Codebase	5		0		5		5	
Dependencies	10		10		10		10	
Config	0		10		5		0	
Backing Services	0		10		10		0	
Build/Release	0		5		5		5	
Stateless	10		0		0		0	
Port Binding	10		10		10		10	
Concurrency	5		10		10		5	
Disposability	5		5		5		5	
Dev/Prod Parity	0		10		10		10	
Logs	0		10		10		0	
Admin Procs	0		0		0		0	

Non-Cloud (38%) → Cloud-Native (67%): biggest jump. Cloud → K8s: zero improvement. K8s adds ops, not app quality

Let's keep cloud native moving

CURTAIN CALL

1. Cloud-native is a set of principles, not a CNCF landscape bingo card
2. Kubernetes is a tool, not an identity
3. Complexity is a liability, not a feature, simplify ruthlessly
4. The next wave (agentic) will punish those who haven't simplified



Kubernetes-Native ≠ Cloud-Native: Avoiding Architecture Theater



Prerit Munjal • preritmunjhal.me