DOJO

LEADDEV LONDON · 2026

From chaos to centralisation.

Taming extreme tech sprawl — how 350 engineers, 90 Kubernetes clusters and 11 database technologies became one Golden Path.

SPEAKER

Ell Sullivan

ROLE

Platform & Staff Engineering, Dojo

HELLO, GREENWICH.

I only travelled *9 miles* to get here today.

I grew up just down the road in Erith — which, for those visiting London, gives a completely different meaning to the phrase "three nines."

A CONFESSION

I helped build one of those silos.

I started in Core Payments. We were moving fast — admittedly, faster than the central Platform team. It took fresh eyes, and a step out of my Tribe, to see what was actually happening.

SECTION 01

01

The sprawl.

An engineering organisation of 350 — running, somehow, on every deployment methodology invented in the last decade.

THE REALITY, MAY 2024

90

Separate Kubernetes clusters across the organisation.

11

Different database technologies in active production use.

4

Distinct, completely incompatible software development lifecycles.

"

You build it. *You run it.*

THE MANDATE THAT PRODUCED 90 CLUSTERS

CONWAY'S LAW, LEARNED THE HARD WAY

Give a siloed tribe total autonomy without a paved road — *they will build their own road.*

Six tribes built six roads. None of them connected. To talk across cluster boundaries, engineers were forced into architectural decisions they knew they shouldn't be making — just to ship a feature.

THE COST OF UNCHECKED AUTONOMY

Autonomy had become anarchy.

01

Disparate security & reliability

02

Severe developer friction

03

No operational leverage

SECTION 02

02

Discovery and architecture.

Before we asked leadership for a mandate, we needed to know exactly what the future looked like.

THE DISCOVERY

A museum of every deployment method invented in the last decade.

- Pure TicketOps
- Ephemeral cloud environments
- Pulumi IaC living inside app code
- Full Terraform with strict app/infra split
- "Infrastructure from code"
- A shared Helm chart everyone agreed was terrible

TWO NON-NEGOTIABLE PRINCIPLES

What our discovery locked in.

01 · The mandate from the top

"Product teams need to spend their time writing business logic — not wrestling with infrastructure."

— SVP of Technology

02 · The signal from the bottom

Product teams had absolutely zero appetite to take on Terraform.

— Every team we interviewed

PLATFORM-AS-A-PRODUCT

One platform, multiple products.

Orchestra

The workload spec and CLI. One YAML, any environment.

Managed Compute

Multi-tenant, domain-tiered Kubernetes with auto-provisioning.

Managed Databases

Mongo, Spanner, Redis, Postgres — tiered, isolated, never shared.

Managed Ingress

Ingress, certs and DNS — end-to-end, zero human interactions.

Secrets Manager

IAM and secret management baked into the platform by default.

Managed Messaging

PubSub today, Kafka next. Topics, subscriptions, schemas.

THE LENS FOR EVERY DECISION

Dual-scalability.

Does this scale for our product workloads — *and* does it scale for the platform team supporting it?

If we couldn't answer yes to both, we didn't ship it.

THE MOST IMPORTANT ARCHITECTURAL CHOICE WE MADE

Squads own domains, *not infrastructure.*

Tribes change. Re-orgs happen. Domains rarely disappear. Domain-based namespacing made our IAM, our isolation boundaries and our FinOps completely immune to organisational churn.

TIERING — EVERY DOMAIN GETS A CLASS

From compute hardware down to FinOps, tier drives everything.

Tier 01

Always-on, multi-region

High-performance iron. Critical payments path. The infrastructure that cannot blink.

Tier 02

Standard production

Where the bulk of our workloads live. The build-out started here — heavy concentration of domains.

Tier 03

Spot & preemptible

Batch, dev, and non-critical workloads. Cheapest possible compute, automatically.

FROM THE FOUNDATIONS UP

Managed Compute.

A multi-tenant Kubernetes environment with strict isolation and auto-provisioning, using GKE Node Auto-Provisioning. Domain model as the boundary. Tier as the node pool — Tier 1 gets the iron, Tier 3 gets the spot. Reliability and cost-control became automatic.

STATE, RUTHLESSLY CURATED

11 database technologies *down to four.*

MongoDB

via Atlas. The default document store for most application workloads.

GCP Spanner

For workloads that need globally consistent relational state at scale.

Redis

Caching, ephemeral state, lightweight pub/sub.

Postgres

Mostly to support open-source apps like Airflow.

Hard rule: no applications sharing databases. Total isolation. Enough choice for developers — small enough for a platform team to actually secure and maintain.

SECTION 03

03

The pitch.

Translating engineering pain into business risk — and how Nando's funded a platform leap.

THE NEGOTIATION TACTIC

Extra-hot peri-peri is the key to *unlocking infrastructure funding.*

Getting time with our CTPO is notoriously challenging. So we booked a lunch meeting and ordered Nando's. He couldn't refuse.

EVEN OUR BEST SDLC — PAYMENTS — LOOKED LIKE THIS

To ship one feature, one developer:

3
REPOSITORIES TO NAVIGATE

5
SEPARATE PULL REQUESTS

6
HUMAN REVIEWS TO WAIT ON

9+
FLOW DISRUPTIONS

9
MANUAL STEPS FOR INGRESS ALONE

A flow disruption is a moment when work completely stops to wait for a handoff, a security ticket, or an infrastructure approval.

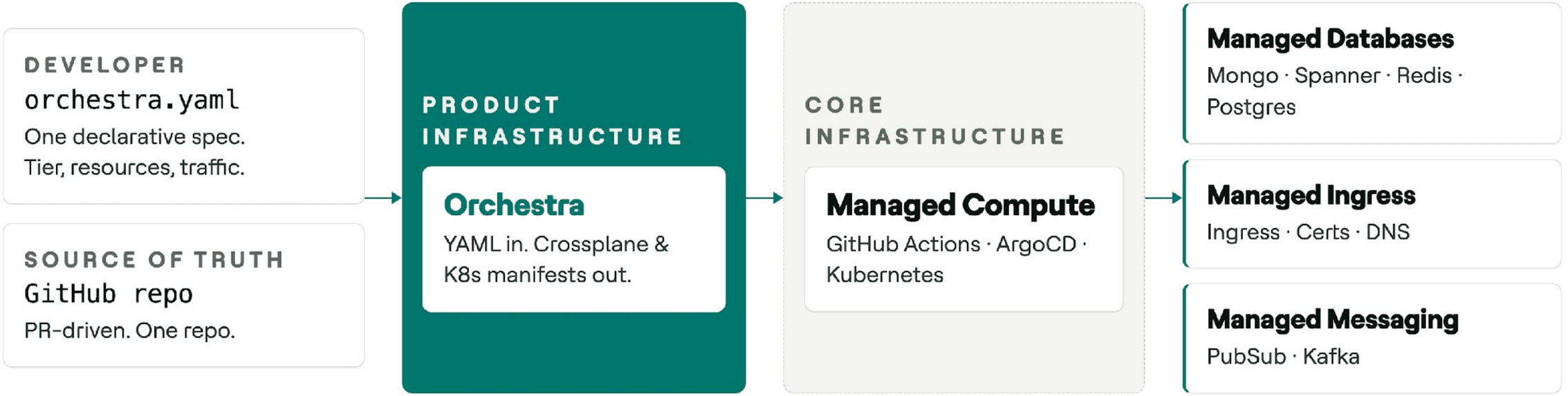
TRANSLATION LAYER FOR THE C-SUITE

Engineering pain becomes business risk.

Multi-step PRs & key-person dependencies	→	**Operational risk.** Multi-region rollouts taking months, not days.
Manual KMS & fragmented pipelines	→	**Reputational & security risk.** High incident rate, inconsistent compliance posture.
Unlabelled, bespoke resources	→	**FinOps failure.** No accurate cost-per-domain. Flying blind on cloud spend.
Tribe-specific SDLCs	→	**Ramp-up penalty.** Every team move destroys engineer productivity for weeks.

THE ARCHITECTURE, END-TO-END

From one YAML to provisioned cloud — through GitOps.



THE INTERFACE

Orchestra.

1
file

1
CLI

∞
environments

No Helm charts. No Terraform for developers. GitOps-native. The same workload spec runs on a laptop that runs in CI that runs in production.

THE WHOLE INTERFACE FOR A DEVELOPER

This is what they write. Orchestra does the rest.

```yaml
apiVersion: platform.dojo.tech/v1alpha1
kind: App
metadata:
  domain: developer-experience
  system: innovation
  name: orchestra-demo-app
runtime: go

containers:
  - name: orchestra-demo-app
    type: application
    imageName: orchestra-demo-app

instruments:
  message:
    type: ManagedDatabase/SpannerDatabaseV1
    instanceName: mt
    migrations: db/migrations
  kafkaTopic:
    type: ManagedMessaging/KafkaTopicV1
    topicName: kafka-topic-v1
```

"

This is amazing. Why isn't *everyone* using this already?

CTPO · AFTER WE SHOWED HIM THE BILLING SQUAD'S APP AND INFRA, SIDE-BY-SIDE, ON ONE SCREEN.

SECTION 04

04

The migration.

Building the platform was the easy part. Moving everyone onto it was something else entirely.

THE HONEST PART

The friction wasn't technical. *It was cultural.*

New apps on Orchestra were a breeze. Migrating existing ones was harder than we ever anticipated. Teams were working and testing in such different ways that adjusting to a standardised Golden Path took real time. The platform had to adjust too — it was a two-way street.

YOU CAN'T SELL EVERY TEAM THE SAME THING

Two pitches, two audiences.

For teams without IaC

Sell on security, compliance, and best-practice out of the box. Easy yes.

For mature, Terraformed teams

Sell on developer experience and zero ramp-up between squads. Earn the trust to take their infra.

A massive shoutout to the Billing team — many of whom are sitting in this room. They took a bet on Orchestra early. The volume of feedback they gave us was invaluable.

WHAT REAL OPERATIONAL LEVERAGE LOOKS LIKE

The Data org migrated *their entire stack* in one quarter.

BEFORE

21+

Disparate Airflow instances scattered across squads.

AFTER

1

Unified, centralised model on the Golden Path.

SECTION 05

05

What's next.

By solving sprawl, we accidentally unlocked something massive.

AN HONEST TAKE ON AI AGENTS

Agents fail in chaos. *They thrive in standardisation.*

In a bespoke tech sprawl

Agents hallucinate. They break things. There's no consistent shape to the world they can reason about.

Inside Orchestra

One PR. One YAML. Constrained, predictable, automated. The perfect, safe sandbox for agentic development.

WHAT WE'RE INVESTING IN NEXT

Spec-driven development.

Standardisation today is the prerequisite for safe AI adoption tomorrow. Same headcount, radically more output.

SECTION 06

06

Takeaways.

Four things to take back to work tomorrow.

IF YOU TAKE A SCREENSHOT, TAKE THIS ONE

Four things to take back to work tomorrow.

01

Quantify the chaos.

Don't pitch a refactor. Map the developer workflow and translate engineering pain into business risk.

02

Abstract to adopt.

You cannot mandate platform adoption. Make the right way the absolute easiest way — one PR, one YAML.

03

Enforce by design.

A Golden Path isn't a wiki page. The guardrails are baked into the abstraction, not into committee meetings.

04

Standardise to scale.

Break the linear scaling of platform overhead. It's also the prerequisite for safe Agentic AI tomorrow.

THE WRAP-UP

We traded 90 bespoke clusters for one Golden Path.

Thank you

Ell Sullivan

Platform & Staff Engineering · Dojo