NATWEST × SYNTASSO

Applying Platform as a Product in a *300-year-old* bank

How one of the UK's largest banks applied industry leading ideas to turn undifferentiated heavy lifting into a self-service platform

THE INDUSTRY VIEW

Abby Bangser
Syntasso Founding Engineer
KubeCon Co-Chair and CNCF Ambassador

THE CASE STUDY

Joel King
NatWest Group Principal Engineer

Most engineering effort is non-differentiating

80%

Generic tooling
CI/CD pipelines, Kubernetes, monitoring stacks…

Duplicated heavy lifting
The same plumbing repeated across every team

Tech debt & maintenance
Keeping the lights on crowds out innovation

Business outcomes are suffering

Speed Safety Efficiency

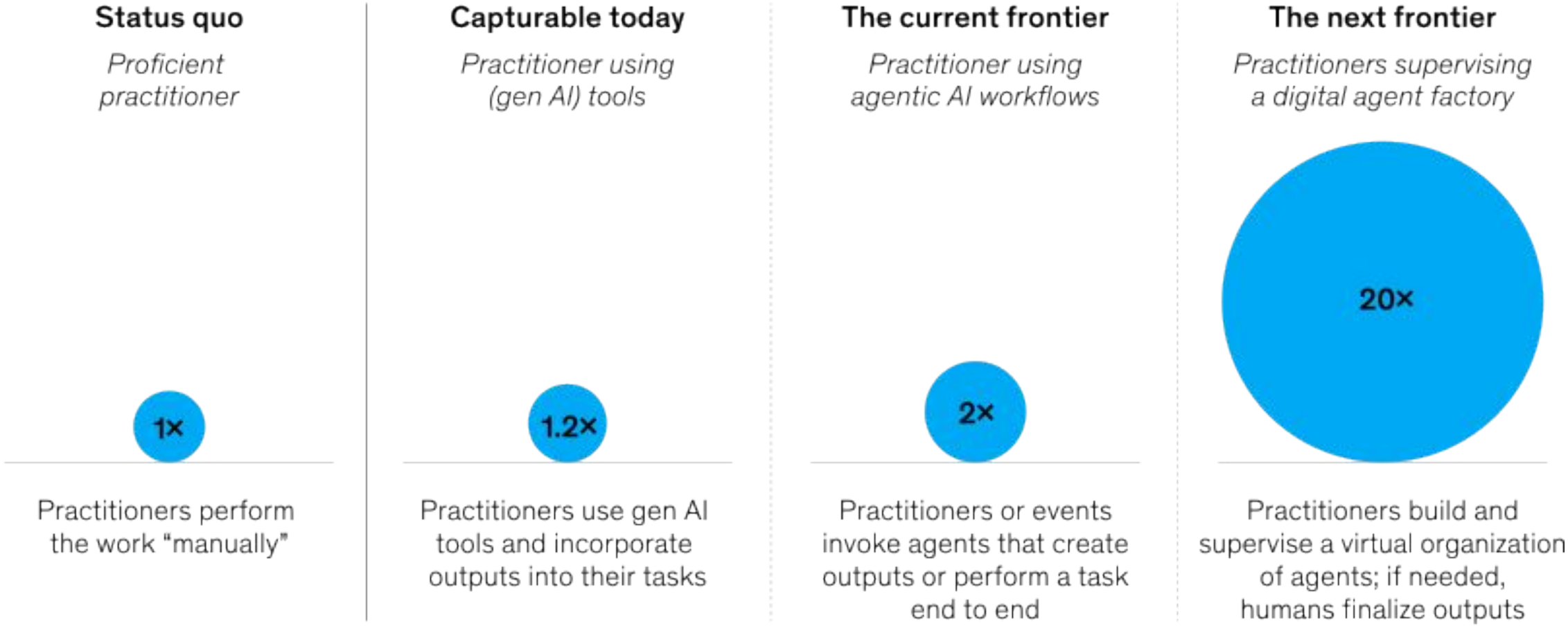
Shift-left shifted cognitive load, not outcomes

68% blocked meeting customer needs by existing tech

76% expect to grow tech headcount for agentic AI.

28% of IT spend reaches application development

And no, AI isn't improving this. AI is compounding it.



NatWest is a 300-year-old, 20-million-customer bank

A UK-focused bank serving over 20 million customers — at a scale that makes every platform decision compound.

20M+
customers served

#1
lender for UK infrastructure

~59k
staff worldwide

11.4M
mobile app users

1M
new customers added in 2025

700k+
new credit-card customers in 2025

81.8%
of retail banking customers bank entirely digitally

Every story needs the forest view and a tree example

ABBY · THE FOREST

The big picture

Tracking industry wide socio-technical patterns.

Codifying trends and supporting measured improvements.

JOEL · THE TREE

A real example

How a single 300-year-old bank actually built it.

The problem, the architecture, and next step scale.

OUR ROADMAP

Where we're going today

The Shift

Why platforms, why now

Building

From theory to outcomes

Scaling

Supporting future vision

At NatWest, scale and complexity collide

COMPLEXITY

135+

patterns in production — with **unsustainable growth** , ever-increasing complexity, and delivery that has slowed to a crawl.

SCALE

10s departments

100s teams

1000s staff

All moving to **containerisation** at once — every team solving the same infrastructure problems in parallel.

The cost: cognitive load, drift, and slow change

01 High cognitive load on development teams

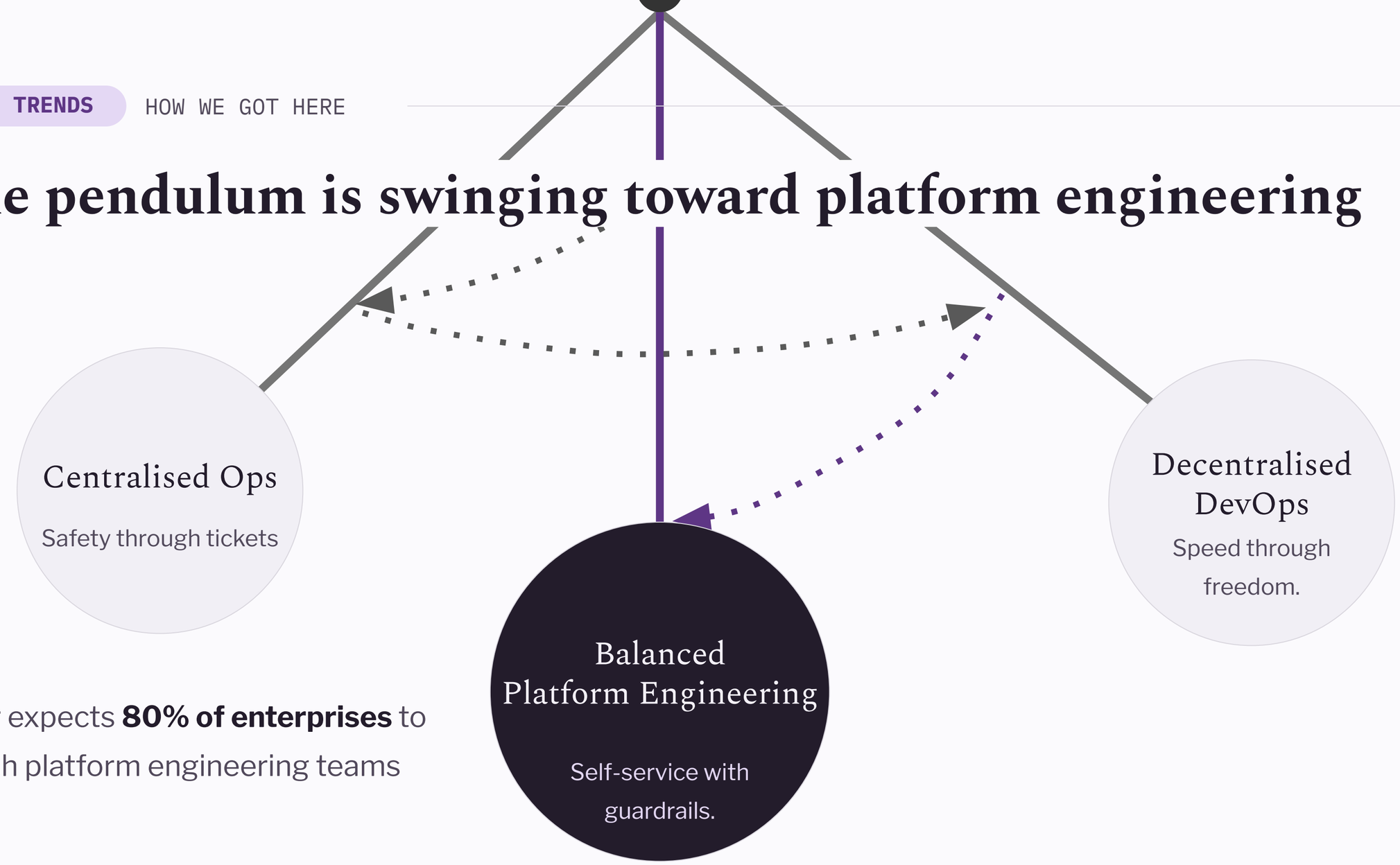
02 Duplication of effort across application teams and services

03 Inconsistent environments and configuration drift

04 Slow, manual, and error-prone change processes

05 Limited visibility and control across multi-cluster fleets

The pendulum is swinging toward platform engineering



Gartner expects **80% of enterprises** to establish platform engineering teams

The hype-cycle needed grounding

A fragmented landscape: everyone agreed platforms mattered but not on what they were

SCEPTICISM

Dismissed as "new hype" or "rebranded ops".

TERMINOLOGY CHAOS

No industry-wide agreement on what this is.

UNCERTAINTY

Big projects with poor ROI.

TOOL-CENTRIC

Repurposing tools over business outcomes.

CNCF is documenting a proven path

WHAT

Platforms White Paper

Building a common language and defining platforms as **outcomes over outputs**.

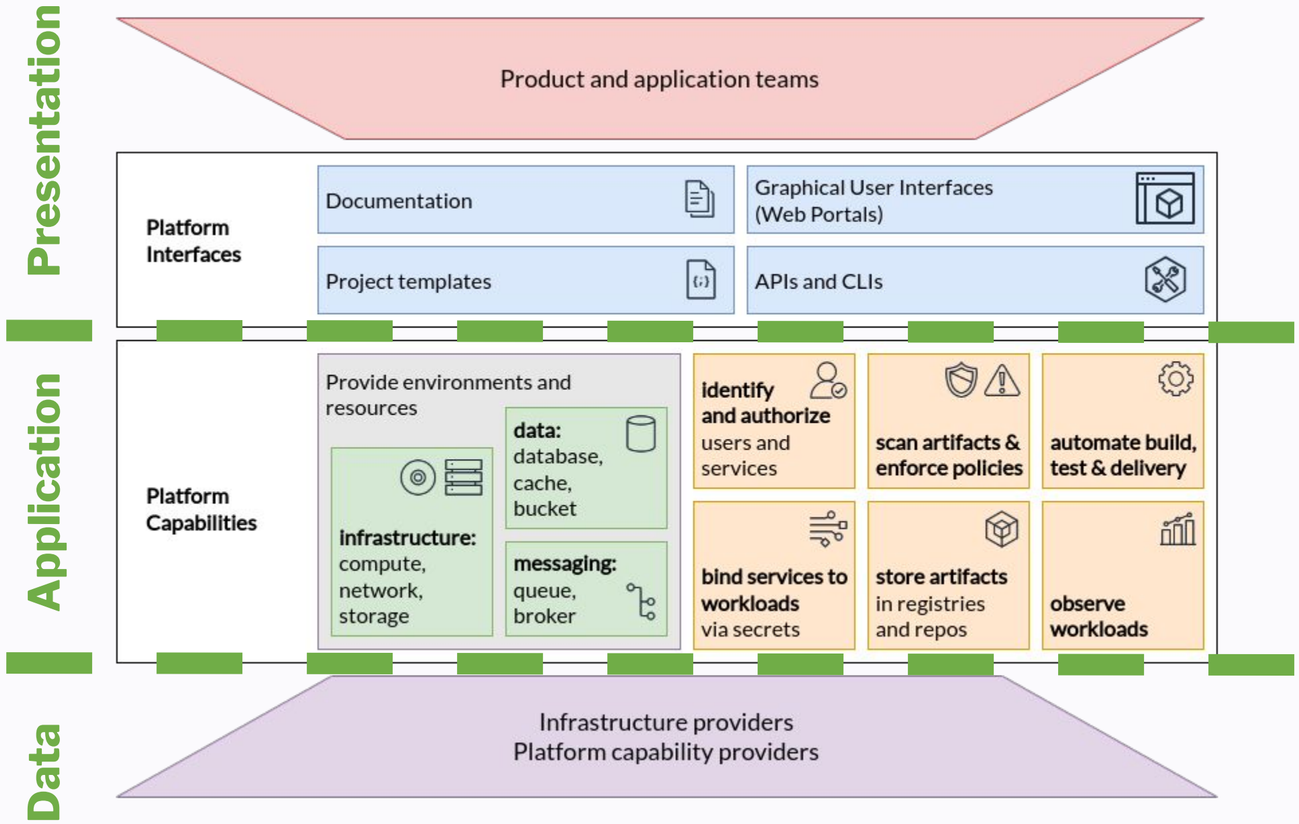
HOW

Maturity Model

Growth framework of **practices and processes** delivering white paper outcomes.

First application of the three-tier architecture

At their most basic, internal platforms provide **consistent experiences for acquiring and using individual services**...As they mature internal platforms also offer compositions of such services.

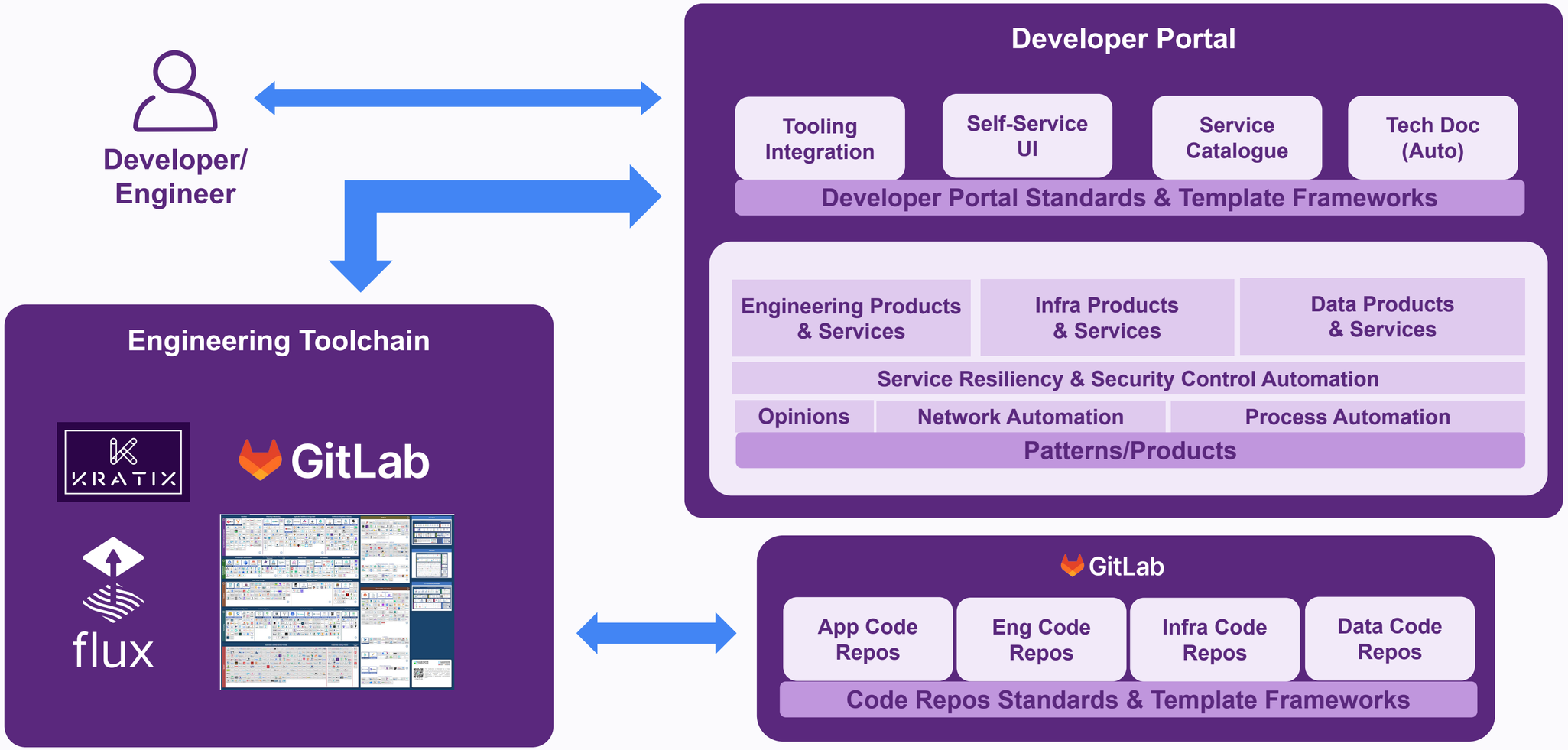


Five dimensions codify organisational trade-offs

Built through end-user, consultant, and vendor collaboration bringing diverse perspectives, real use cases.

Investment	How are staff and funds allocated to platform capabilities?
Adoption	Why and how do users discover and use internal platforms?
Interfaces	How do users interact with and consume platform capabilities?
Operations	How are platforms planned, prioritised, developed, and maintained?
Measurement	How is feedback gathered, incorporated, and learned from?

Platform as a Product, end to end



Kubernetes and GitOps put the platform on rails

01 · SOURCE OF TRUTH

All platform and infra definitions version-controlled in Git, reducing cognitive load.

02 · GOLDEN PATHS

Reusable, declarative templates eliminate duplication and drive consistency.

03 · RECONCILIATION

Flux keeps clusters aligned with desired state — no configuration drift.

04 · FLEET CONTROL

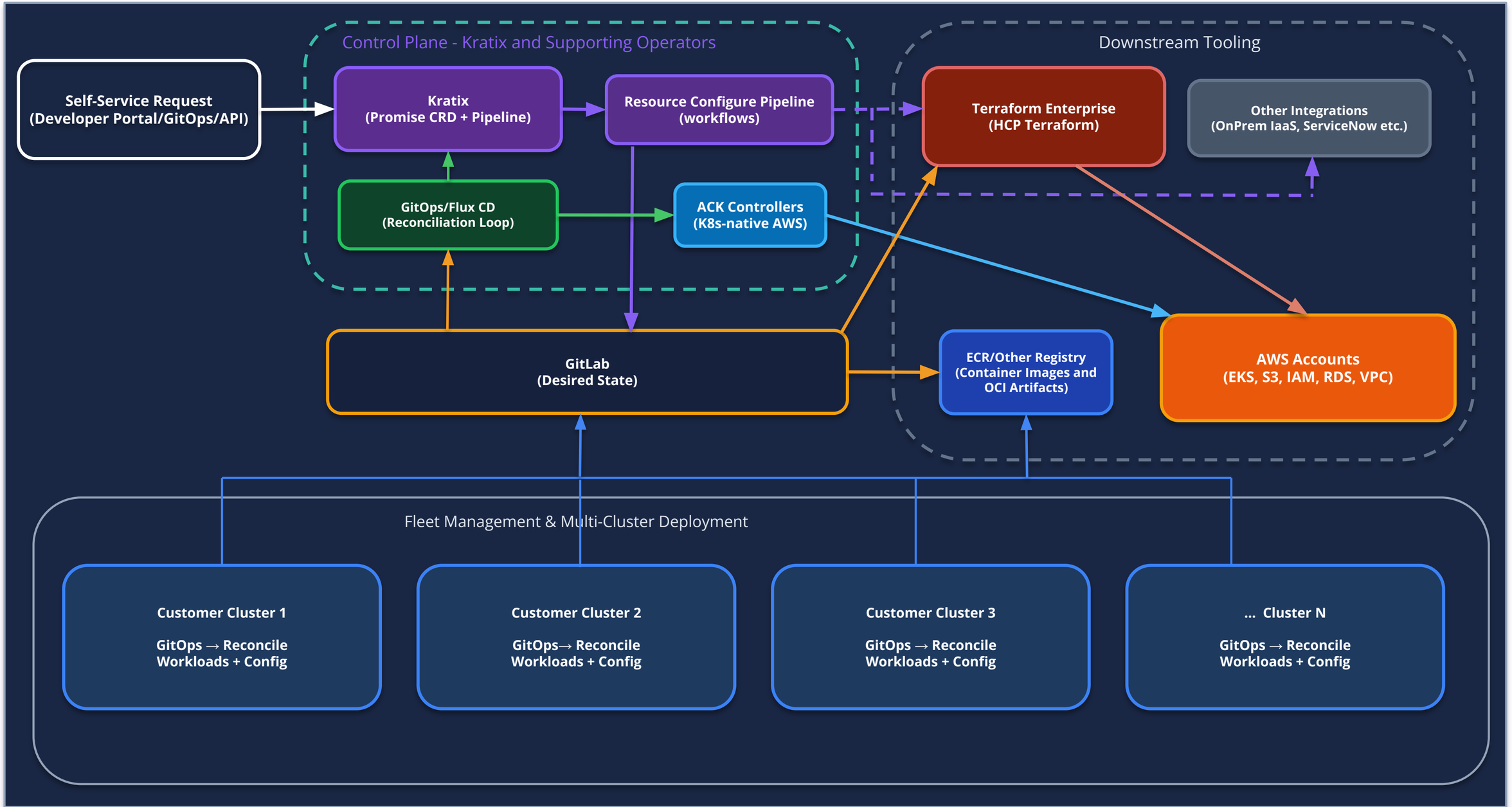
Multi-cluster GitOps enables scalable, consistent rollout of capabilities.

05 · SAFE CHANGES

Every change is reviewed, approved, and traceable through Git.

06 · DEVELOPER EXPERIENCE

Developers consume capabilities as curated products — not bespoke setups.


Control Plane - Kratix and Supporting Operators
Downstream Tooling
Self-Service Request
(Developer Portal/GitOps/API)
Kratix
(Promise CRD + Pipeline)
Resource Configure Pipeline
(workflows)
Terraform Enterprise
(HCP Terraform)
Other Integrations
(OnPrem IaaS, ServiceNow etc.)
GitOps/Flux CD
(Reconciliation Loop)
ACK Controllers
(K8s-native AWS)
GitLab
(Desired State)
ECR/Other Registry
(Container Images and
OCI Artifacts)
AWS Accounts
(EKS, S3, IAM, RDS, VPC)
Fleet Management & Multi-Cluster Deployment
Customer Cluster 1
GitOps → Reconcile
Workloads + Config
Customer Cluster 2
GitOps→ Reconcile
Workloads + Config
Customer Cluster 3
GitOps → Reconcile
Workloads + Config
... Cluster N
GitOps → Reconcile
Workloads + Config

Innovators are building composable marketplace platforms

EXPERIENCES

"Platform capabilities are transparently integrated into the tools and processes that teams already use."

"Platform teams are most responsible for the interfaces and user experiences."

CAPABILITIES

"Capability providers take on the brunt of responsibility for maintenance."

"A platform should be composable — enabling product teams to provide and manage their own capabilities."

Defining the multi-player friendly interface

PROVIDERS

- Security
- Data
- Finance
- Vendor

Independent, managed, composable, and testable.

→

PROVIDER INTERFACE

- API
- Support
- Standards
- Tools

Design for coherency over blind compliance.

→

CONSUMERS INTERFACE

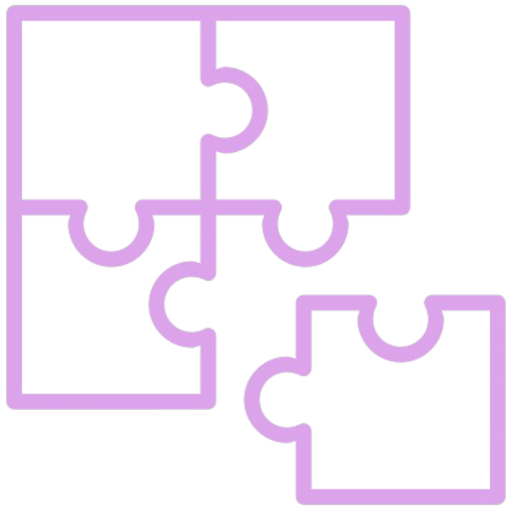
Anyone can consume **any capability** through prefered tools via **consistent user APIs.**

In other words, you need to build a ever flexible puzzle

Always with an eye on delivering the ability to ***adapt quickly***, ***build confidently***, and ***operate sustainably***

Platform wide principles:

- Extensible
- Composable
- Declarative
- Governable
- Observable
- Evolvable

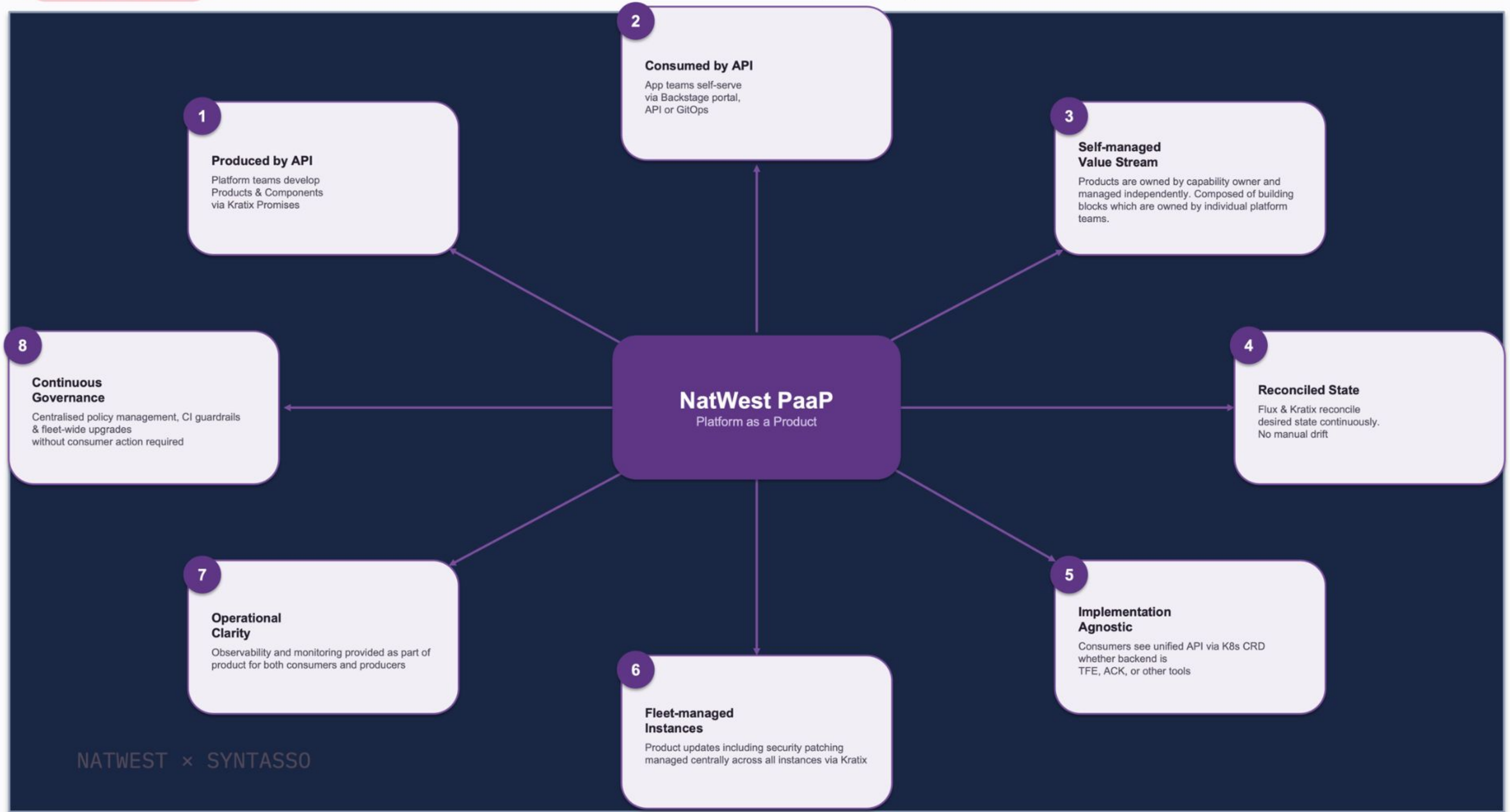


Component specific principles:

- Product Scoped
- Self-Serve
- Operationally Owned

Initial 8 factors that support an effective interface

1. **Produced by API**
2. **Consumed by API**
3. **Self-managed value stream**
4. **Reconciled state**
5. **Implementation agnostic**
6. **Fleet-managed instances**
7. **Operational clarity**
8. **Continuous governance**


2
Consumed by API
App teams self-serve via Backstage portal, API or GitOps
1
Produced by API
Platform teams develop Products & Components via Kratix Promises
3
Self-managed Value Stream
Products are owned by capability owner and managed independently. Composed of building blocks which are owned by individual platform teams.
8
Continuous Governance
Centralised policy management, CI guardrails & fleet-wide upgrades without consumer action required
NatWest PaaP
Platform as a Product
4
Reconciled State
Flux & Kratix reconcile desired state continuously. No manual drift
7
Operational Clarity
Observability and monitoring provided as part of product for both consumers and producers
5
Implementation Agnostic
Consumers see unified API via K8s CRD whether backend is TFE, ACK, or other tools
6
Fleet-managed Instances
Product updates including security patching managed centrally across all instances via Kratix
NATWEST × SYNTASSO

Outcomes of applying Platform as a Product

01 Faster time to value

02 Higher Autonomy

03 Improved Security and Compliance

04 Standardisation at scale

05 Clear ownership boundaries

THE RECAP

Where we've been today

The Shift

It happened, what does it mean

- Platforms fill the gap of what is unique to your business but common to your teams
- It is worth building once, but accept not everyone will fit

Initial Build

Product over project approach

- Success socio-technical both in investment and outcomes
- Early customers and contributors validate the path

Extended Innovation

Sustainability breeds growth

- The only guarantee is change, reducing bottlenecks is key
- Clear ownership boundaries and outcomes drives engagement

THE TAKEAWAY

Outcomes *over* outputs demands flexibility

The industry is moving toward composable, self-service platforms. NatWest shows it's buildable today with golden paths, reconciled state, and capabilities consumed as products.

THANK YOU AND QUESTIONS?

THE FOREST
Abby

THE TREE
Joel