
KubeCon
CloudNativeCon
Europe 2025
LONDON BUS

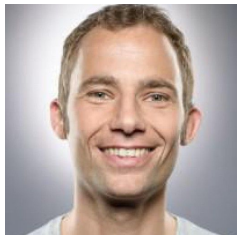
KubeCon | CloudNativeCon
Europe 2025

Platform Engineering for Architects
Crafting Platforms as a Product

Hilliary Lipsig, RedHat
Max Körbächer, Liquid Reply
Andreas Grabner, Dynatrace (....)

Hillary Lipsig
Senior Principal SRE
RedHat

Max Körbächer
Cloud Native Advisor & Managing Director
Liquid Reply



Andreas Grabner
DevRel & CNCF Ambassador
Dynatrace

KubeCon | CloudNativeCon
Europe 2025

Internal Development Platform “Projects” tend to fail.

- Adoption
- Covering the needs and demands of its end users
- Management expectations
- Missing to meet promised value

Those failure is often caused by people, culture and processes

- No defined purpose
- Missing a vision
- Technical debts
- Over engineering
- No listening/measurement of acceptance & effectiveness
- “Randomeering”

AND a **infrastructure-first thinking**.

How infrastructure-first thinking kills your platform ambitions.

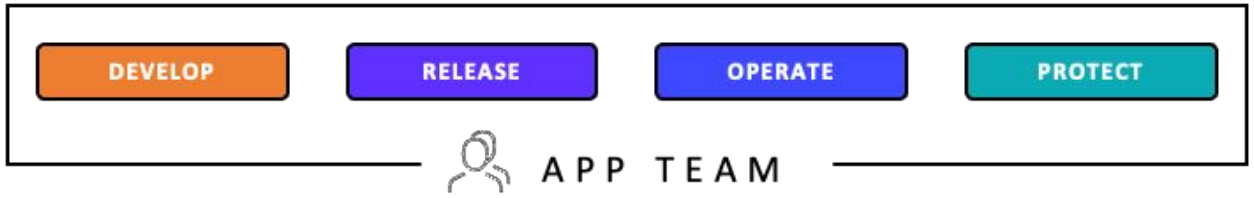
Infrastructure-First Thinking

Infrastructure-first thinking is an approach to platform development that focuses primarily on the underlying technology stack and infrastructure components before considering developer experiences or business outcomes. This mindset prioritizes technical architecture decisions, i implementation details as the starting point

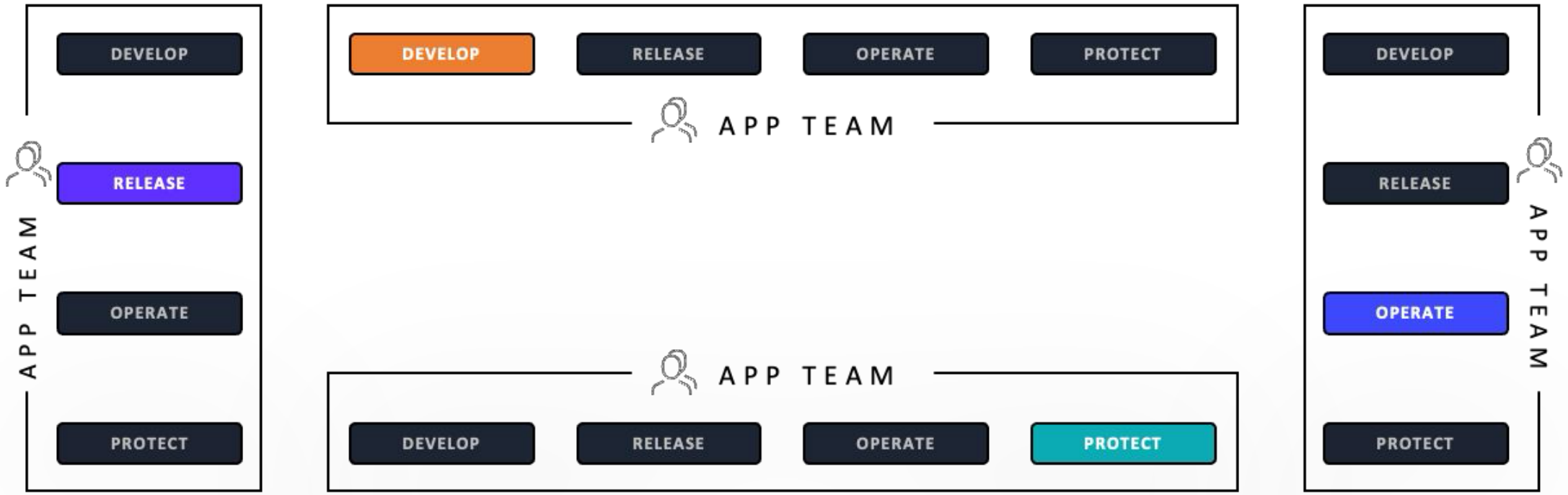
Characteristics of Infrastructure-First Thinking:

- **Technology-Centered**: Places primary focus on selecting technologies, tools, and infrastructure components
- **Architecture-Driven**: Begins with architectural decisions rather than user needs
- **Tool-Oriented**: Emphasizes building a collection of tools rather than cohesive developer experiences
- **Bottom-Up Approach**: Designs from the infrastructure layer upward rather than from user needs downward
- **Solution Before Problem**: Often selects technologies before fully understanding the problems to solve

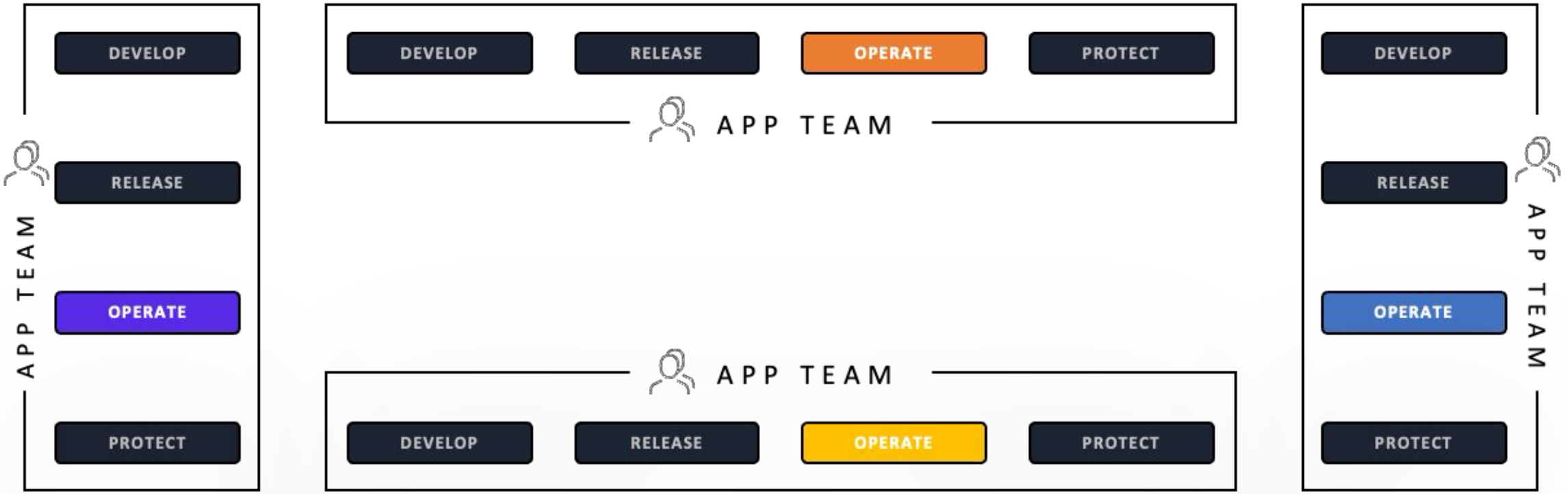
New Tech Stacks requires “Shifting Left”!
Teams taking on end-2-end responsibility



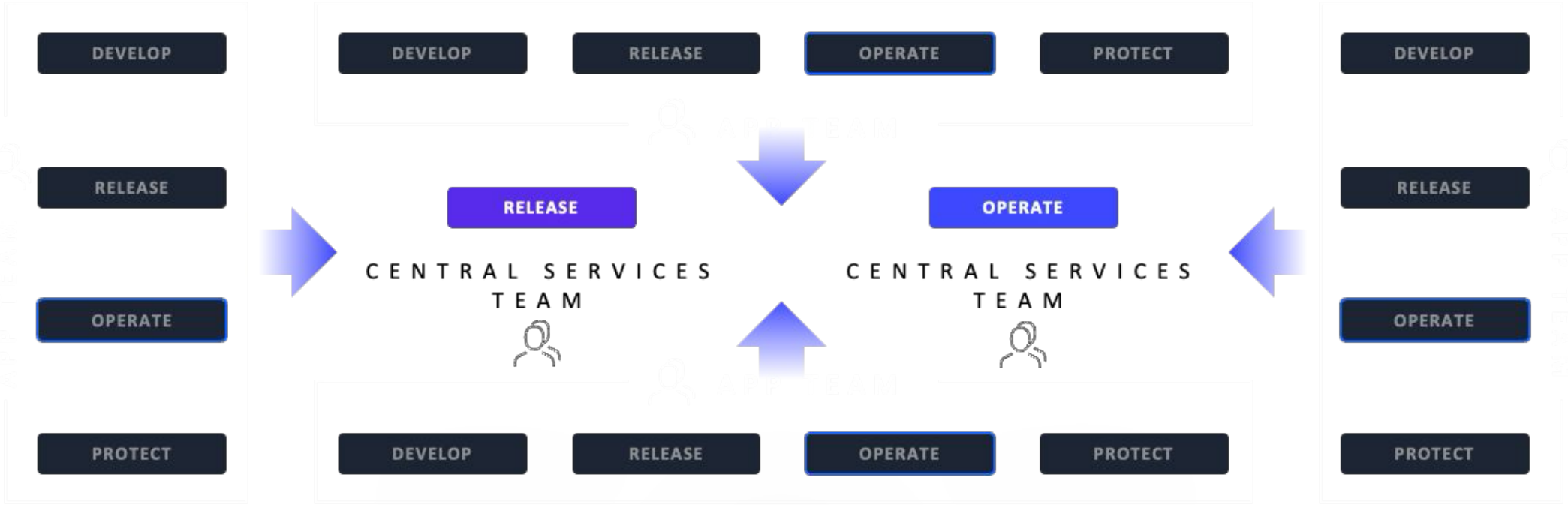
When new tech stack adoption grows - Teams start do things “their way”!



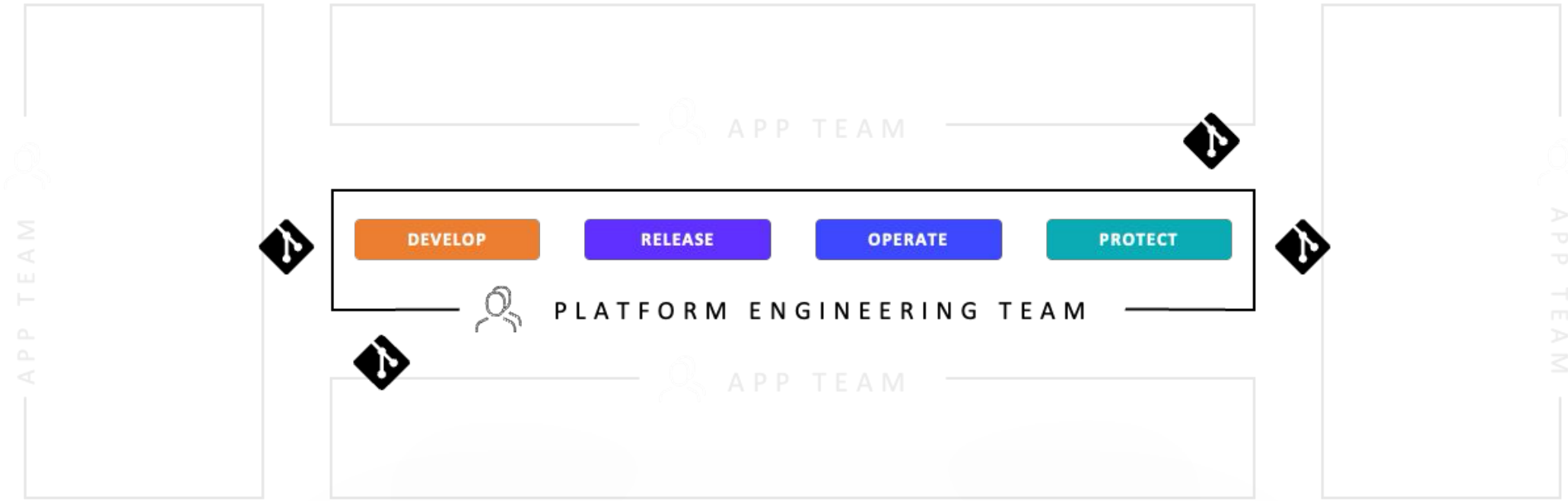
When new tech stack meets old stacks - “their way” becomes “duplicated effort!”



Central Services became the “Patch Solution”



Leading to Platform Engineering:
Git* & IDPs (Internal Development Platform)
for “Golden Path Self-Service”

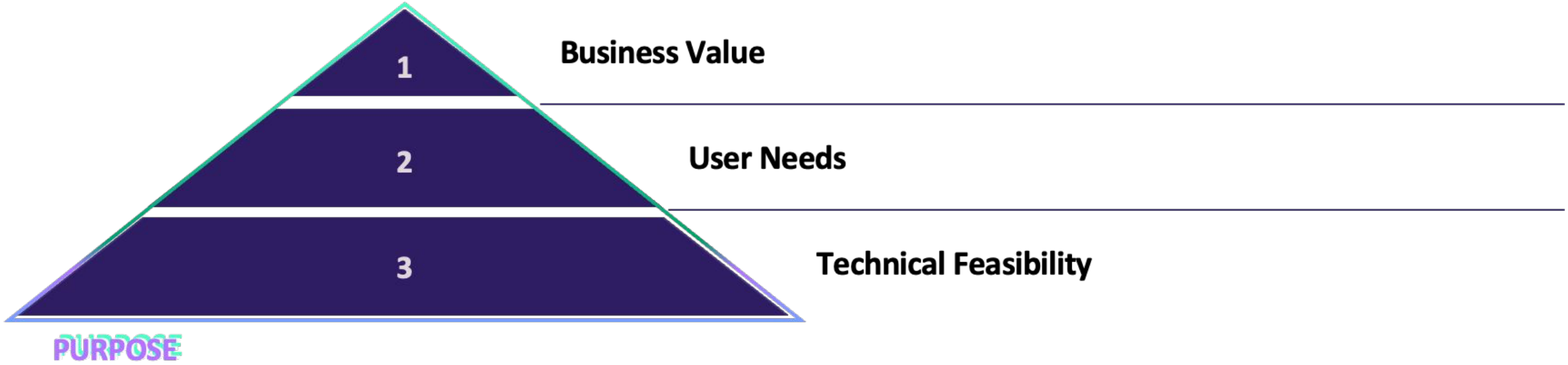


KubeCon | CloudNativeCon
Europe 2025

> *A platform without adoption is just infrastructure. A product mindset is what turns platforms into indispensable tools that developers want to use, not tools they're forced to use.*

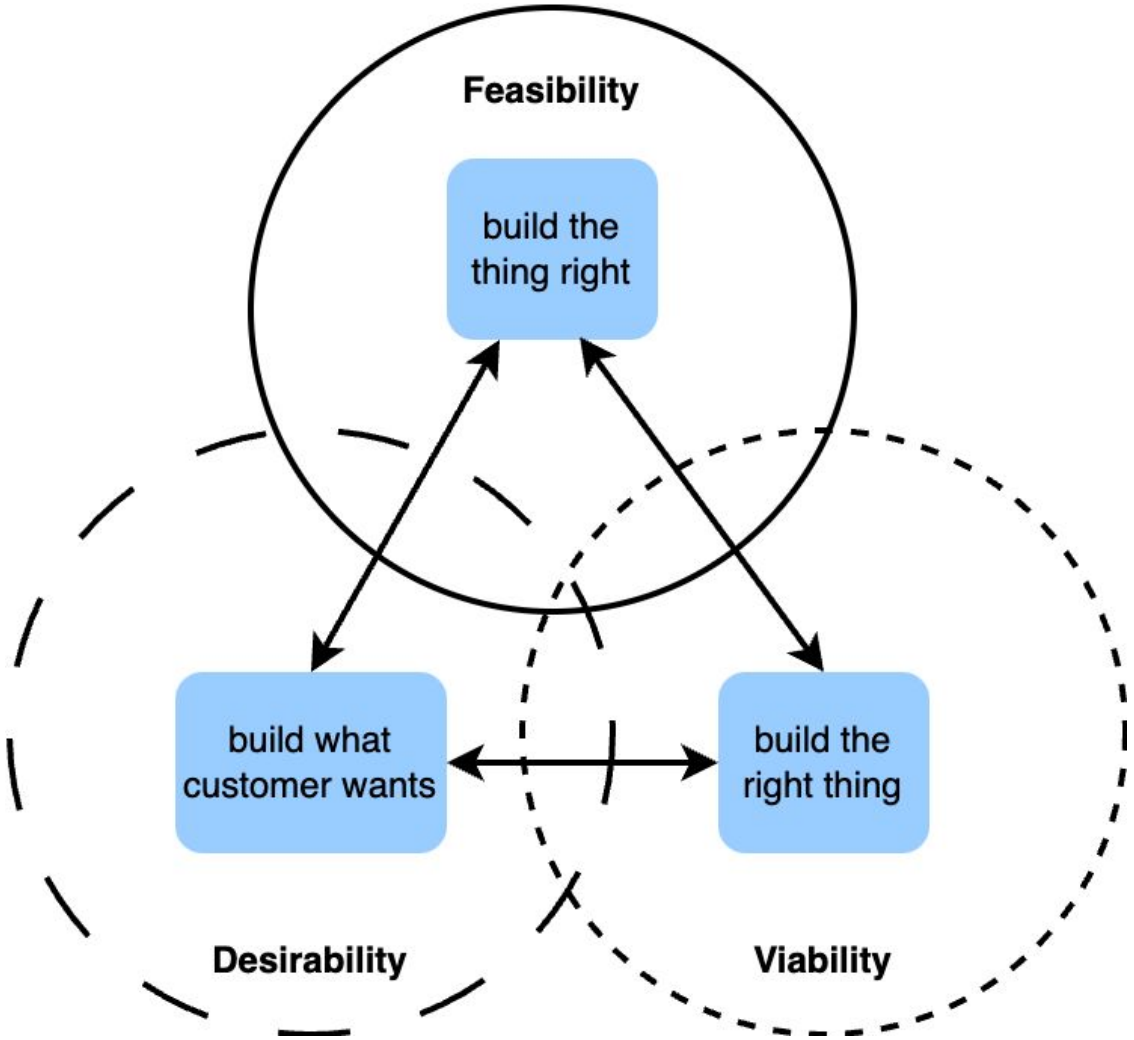
The Importance of Purpose

A well defined purpose builds the bracket around business and technical demands giving to it sense and direction.

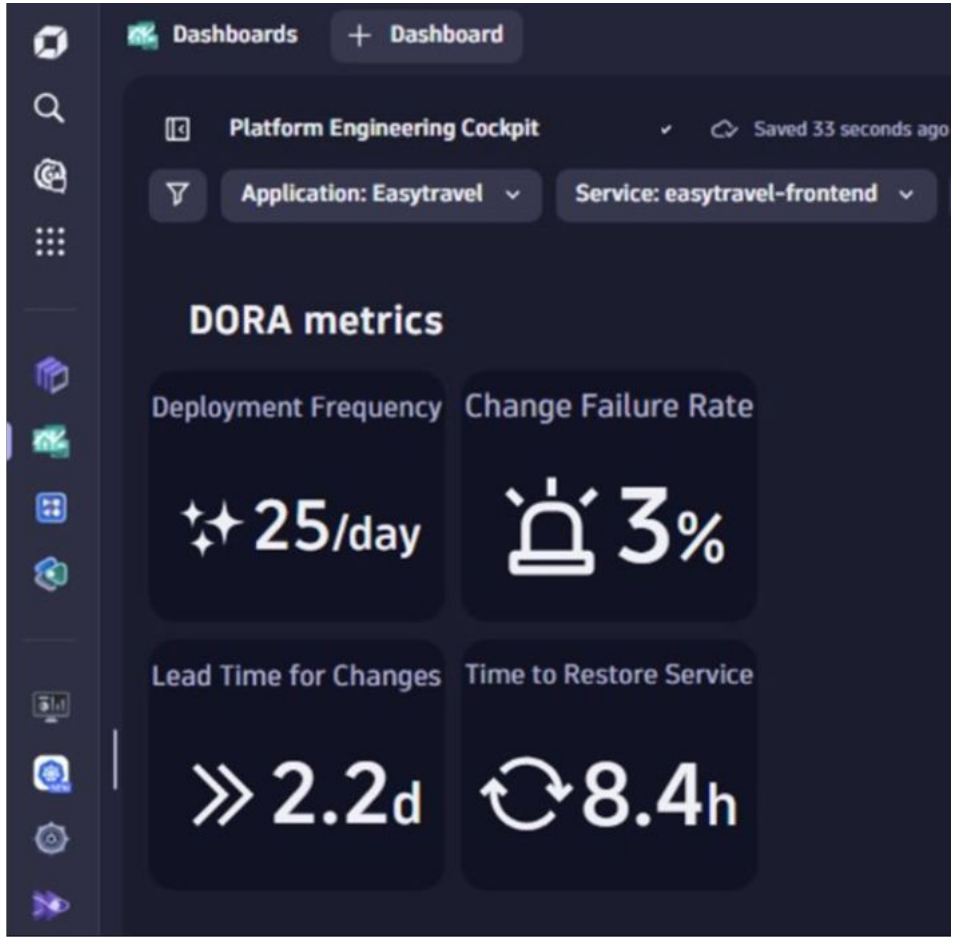


- Defining purpose ensures that the platform's realization stays on track.
- All features must ladder up to a clear purpose.
- It guides decision-making, as well as resource allocation.

Creating a platform for everyone is impossible



DORA Metrics: Measuring Development Performance



1

Deployment Frequency
How often code is deployed.

2

Lead Time
Time to production.

3

Change Failure Rate
Percentage of failed deployments.

4

Time to Restore
How long to recover from failures.

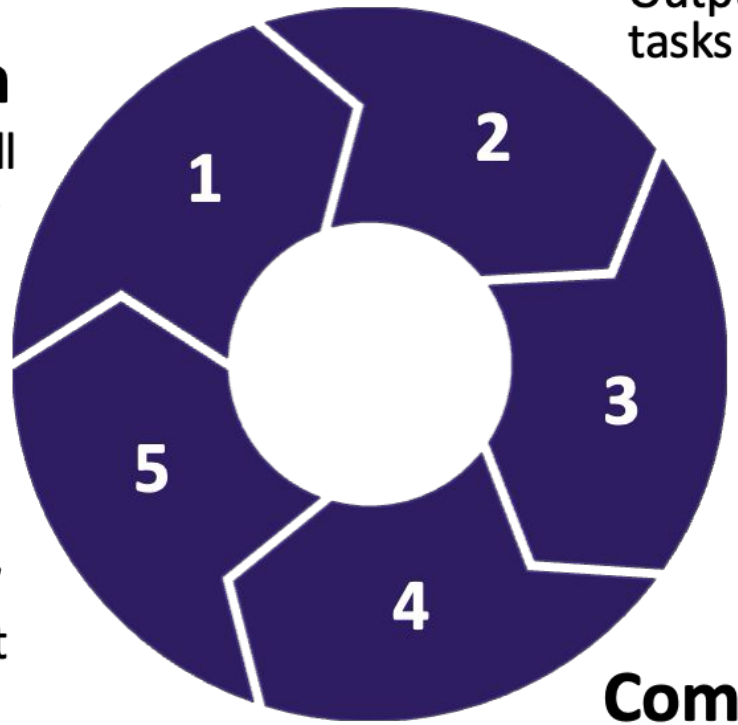
SPACE Metrics: Gauging Platform Adoption

Performance

Output-oriented metrics, such as tasks completed or releases published

Satisfaction

Collect information about overall fulfillment, happiness and work-releated mental health.



Activity

Focusing on activities such as the number of commits, merge requests, or code reviews

Efficiency

How smooth the development and deployment process go

Communication

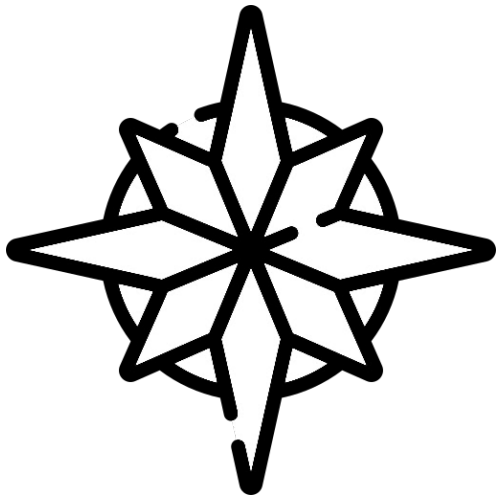
Examine a team's communication behavior and alignment

Following a north star requires many disciplines

Deliver a clear purpose

User Research

Measure and adept to the efficiency & adoption



Tech Debt Management

Establish Principles

The imperative of change

Challenge Conway’s Law

Build a Community

Manage Technical debt by building what matters

User Research

Conduct thorough user research. Understand their pain points.

Feedback Loops

Establish feedback loops. Continuously improve the platform.

Iterative Design

Use an iterative design approach. Adapt to changing needs.

User Research:

Conduct User Interview:

- Ask about current tooling
- What is the current workflow with the tool?
- What would be an ideal workflow?
- What are the existing gaps?
- What do they think a migration effort would look like?

Feedback Loops:

- What works and what doesn’t?
- Did anything you thought you wanted not work out?
- Have your requirements changed?

Manage Technical debt, by knowing when to change directions.

- Why was a decision taken? What would need to happen to trigger a new decision?
- Adjust to shifting requirements
  - As products change, developer workflows may change as well.
- Iterative design. Implement the Thinnest Viable Platform first, then incrementally add improvement, running the evaluation loop after each new implementation
- Avoid the Sunk Cost Fallacy

… and then there are the Depreciation Criterias

Platform Engineering is NOT

The latest K8s & tech

Backstage

GitOps

Rebranded DevOps

Infra departments believing to know what someone else needs

Platform Engineering and its success is defined by

People

Culture

Purpose

And anything else between the tech layers

Open Questions?

Free book give away:

- 2nd of April - Dynatrace S240 during booth crawl
- 3rd of April - Syntasso S641 during lunch break

SPACE Metrics: Examples

	Satisfaction & Well-being	Performance	Activity	Communication & Collaboration	Efficiency & Flow
Individual	• Developer Satisfaction • Retention	• Code Review Velocity	• Focus Time • # Commits • # Issues / PBIs • Lines of Code	• Code Review Score (quality) • PR Merge Times	• Knowledge Sharing • X-Team Reviews
Team	• Developer Satisfaction • Retention	• Velocity (shipped) • Delivery Lead Time	• Cycle Time • Velocity (done) • # Issues / PBIs	• Code Review Engagement • PR Merge Times • Meeting Quality	• Code Review Stale Time • Handoffs
System	• Satisfaction with Engineering System	• Velocity • Lead Time • Customer Satisfaction • MTTR	• Deployment Frequency	• Knowledge Sharing • X-Team Reviews	• Lead Time • Velocity

Source: Michael Kaufmann, Packt, Accelerate DevOps with GitHub