LEARNING LOUNGE · KUBECON

What Platform Engineers Need to Know About Developer Experience

Building platforms that developers actually love to use

Max Körbächer

REPLY LIQUID

The Disconnect

Platform Team Thinks:		Developer Reality:
"We built a self-service portal!"		"Where do I even start?"
"Kubernetes handles it all"	≠	"Why did my deploy fail... again"
"Docs are in Confluence"		"I spent 2 days on YAML"
"We standardized on Helm"		"I just need to ship a feature"

The gap between 'built it' and 'easy to use' is where developer productivity dies.

What Is Developer Experience?

> *"The sum of all interactions and feelings a developer has while working with your platform; from first contact to production deployment."*

Cognitive Load

How much mental effort does the platform demand?

Every abstraction should reduce complexity, not add it.

Flow State

Can developers stay in a productive focus zone?

Interruptions, slow feedback, and friction kill flow.

Feedback Loops

How fast can a developer know if something worked?

Tight loops = faster iteration = happier devs.

Your Platform Is a Product

The Mindset Shift

Infrastructure	→	**Product**
Internal tool	→	**Developer-facing service**
"It works"	→	**"Delightful to use"**
Ticket-driven	→	**Self-service**

Product Thinking in Practice

- Treat devs as your primary users
- Run user research & interviews
- Define success metrics (DORA, etc)
- Maintain a product backlog
- Publish changelogs & release notes

Golden Paths: Paved Roads, with escape hatches

✓ Automated, opinionated, documented · Zero-config for happy path · Escape hatches exist

01

Opinionated defaults

Make the right way the easy way. Default choices should follow best practices: security, observability, resource limits baked in.

02

Escape hatches

Never trap developers. Golden paths guide, they don't cage. Power users must be able to go off-path without fighting the platform.

03

Progressive disclosure

Simple things stay simple.
Complex things are possible.
Don't show every knob upfront only reveal complexity as needed.

Golden Pat

xist

01

Opinionated defaults

Make the right way the ea
Default choices should fol
practices: security, observ
resource limits baked in.

e disclosure

s stay simple.
gs are possible.
very knob upfront only
exity as needed.

Measuring What Matters

DORA Metrics

platform health proxy

Deployment Frequency

How often devs ship

Lead Time for Changes

Commit → production speed

Change Failure Rate

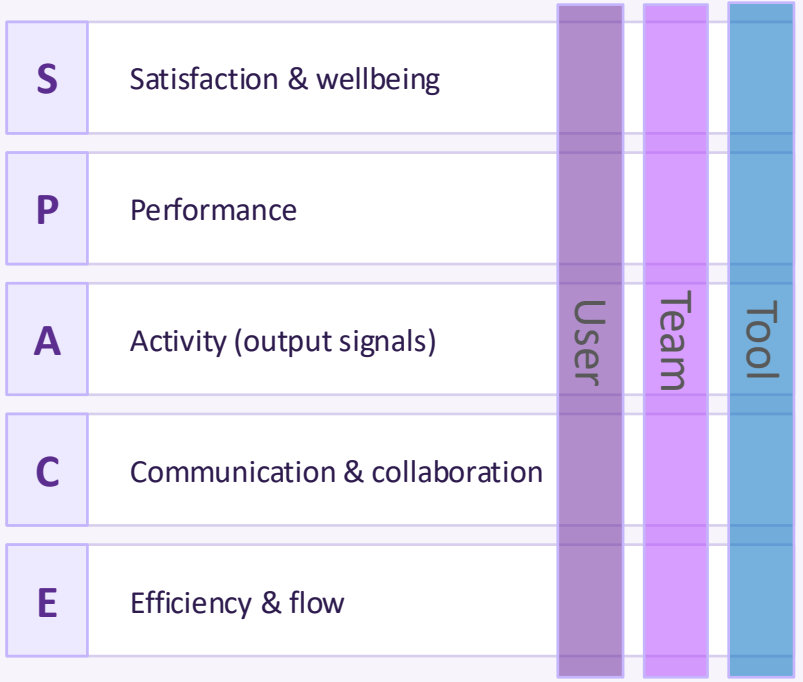
% deploys causing issues

Time to Restore

Recovery from failures

SPACE Framework

developer wellbeing



Top DevEx Killers on Kubernetes Platforms

1

Cryptic error messages

"Error: exit code 137" tells a dev nothing. Platform errors must be actionable, with context and next steps.

2

Slow CI/CD feedback loops

Waiting 30 min for a test failure is a creativity killer. Target < 5 min for the inner loop.

3

Documentation debt

Outdated docs are worse than no docs. Treat docs as code, version them, test them, own them.

4

Secret sprawl & config overload

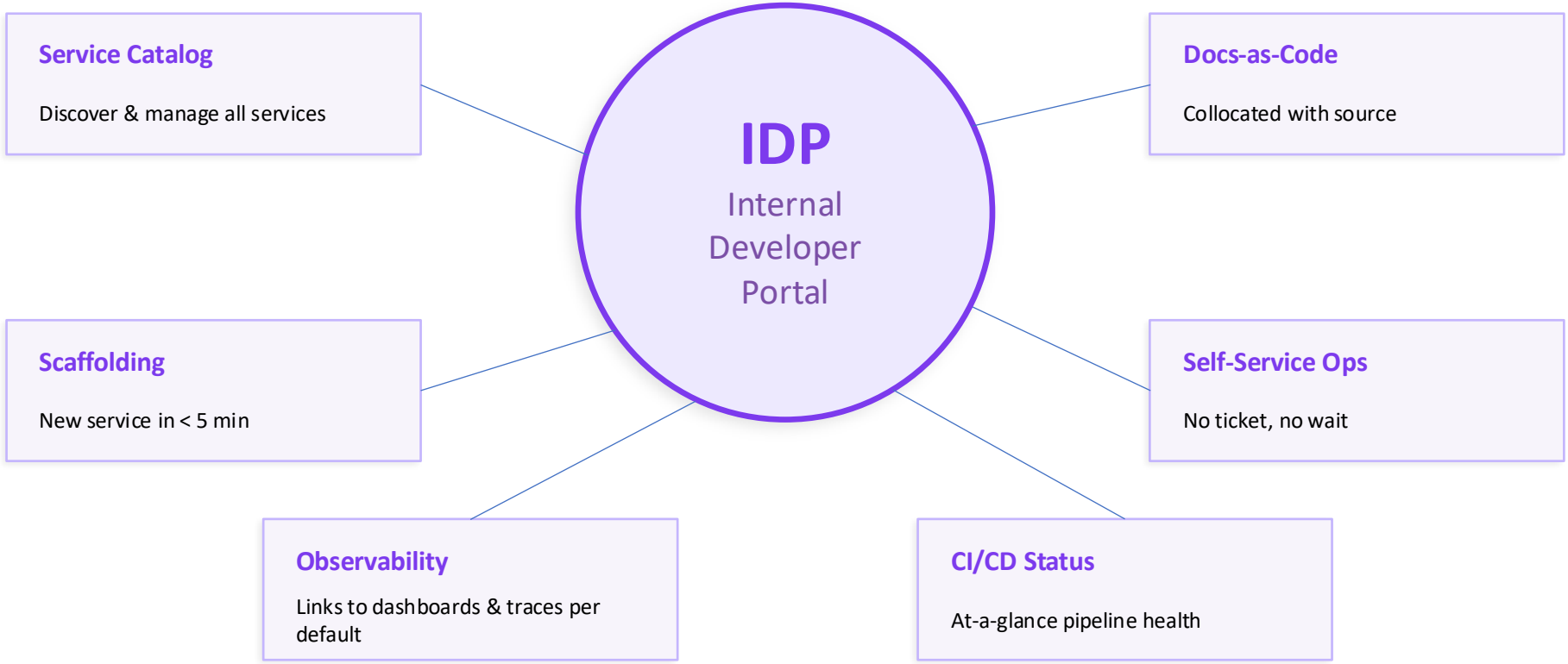
Developers shouldn't need 12 different config files and 3 secrets managers to deploy a service.

5

Invisible platform changes

Silent breaking changes destroy trust. Changelogs, migration guides, deprecation warnings are non-negotiable.

The IDP as DevEx Anchor



Build Empathy Into Your Practice

> *"You cannot design a good developer experience from a conference room. You have to ship with your users."*

Developer interviews

Monthly 30-min 1:1s with different teams. Ask: what slowed you down this week?

Friction logs

Ask devs to document every time they get stuck. Review weekly. Prioritize ruthlessly.

Shadow sessions

Sit with a dev trying to onboard or deploy. Watch. Don't help. Take notes.

Platform NPS

"How likely are you to recommend our platform?" Simple. Quarterly. Actionable.

Office hours

Weekly open session. Lower the barrier to giving feedback. Show you're listening.

Public roadmap

Show what's coming. Let developers vote and comment. Builds trust and buy-in.

The best platform is one developers forget they're using.

Now go talk to a developer. 👋

LinkedIn