

From DevOps Teams to Platform Teams

And what did we solve?

About me

Jacob Lärfors, Consultant & Founder @ Verifa

10+ years doing CI and DevOps

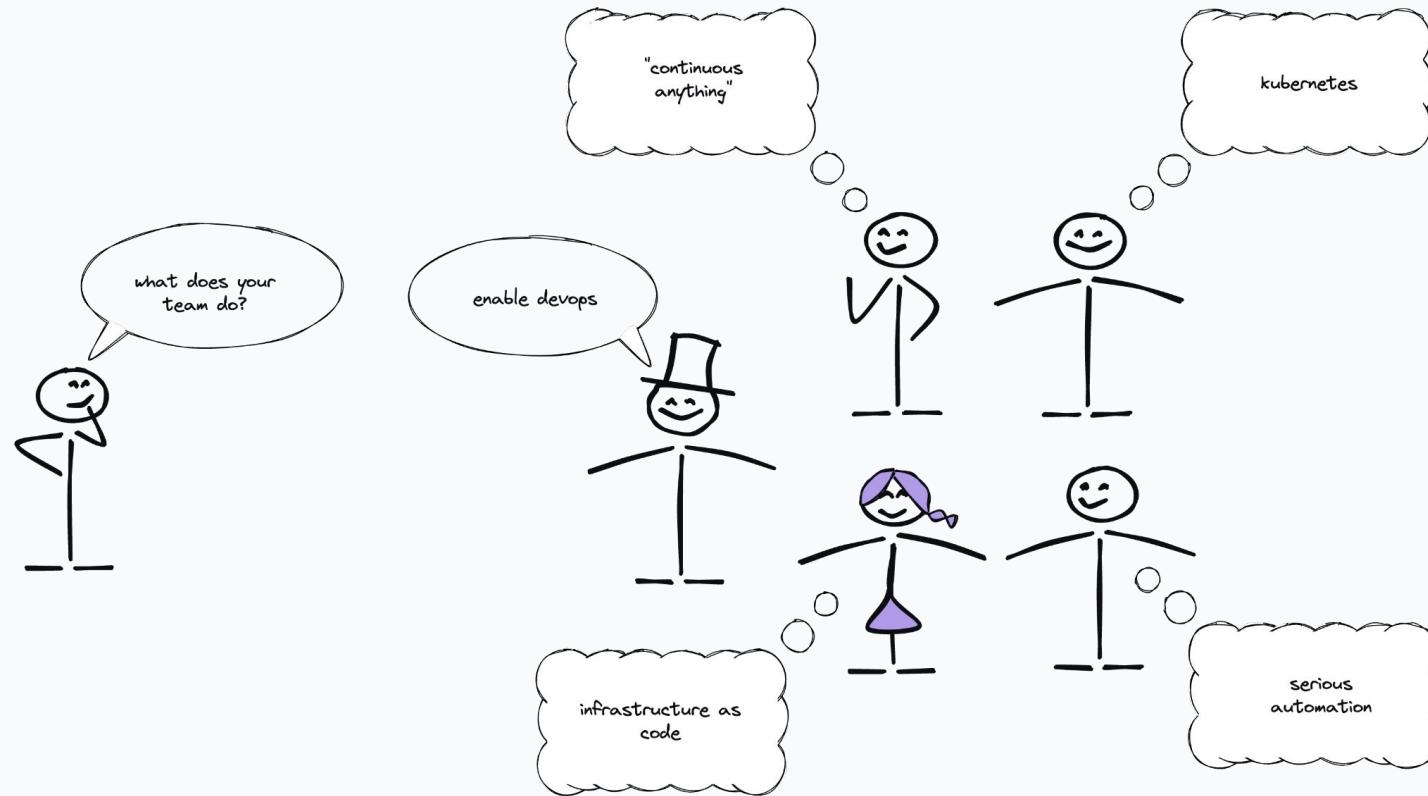
Father, hobby musician and juggler.

LinkedIn: [@jlarfors](#)

GitHub: [@jlarfors](#)

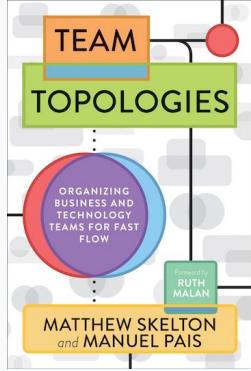


Your typical DevOps Team



Characteristics of a **DevOps Team**

- Unclear goals and roles
- Reactive
- Bad at saying “no”
- Unclear team boundaries



“provide a compelling internal product to accelerate delivery by Stream-aligned teams”

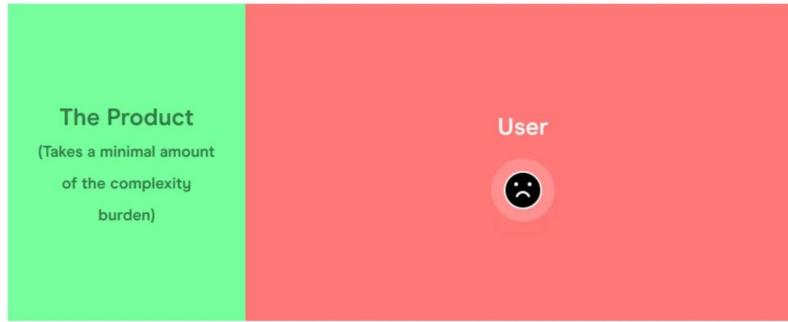
- Team Topologies

“Autonomous delivery teams can make use of the platform to deliver product features at a higher pace, with reduced coordination.”

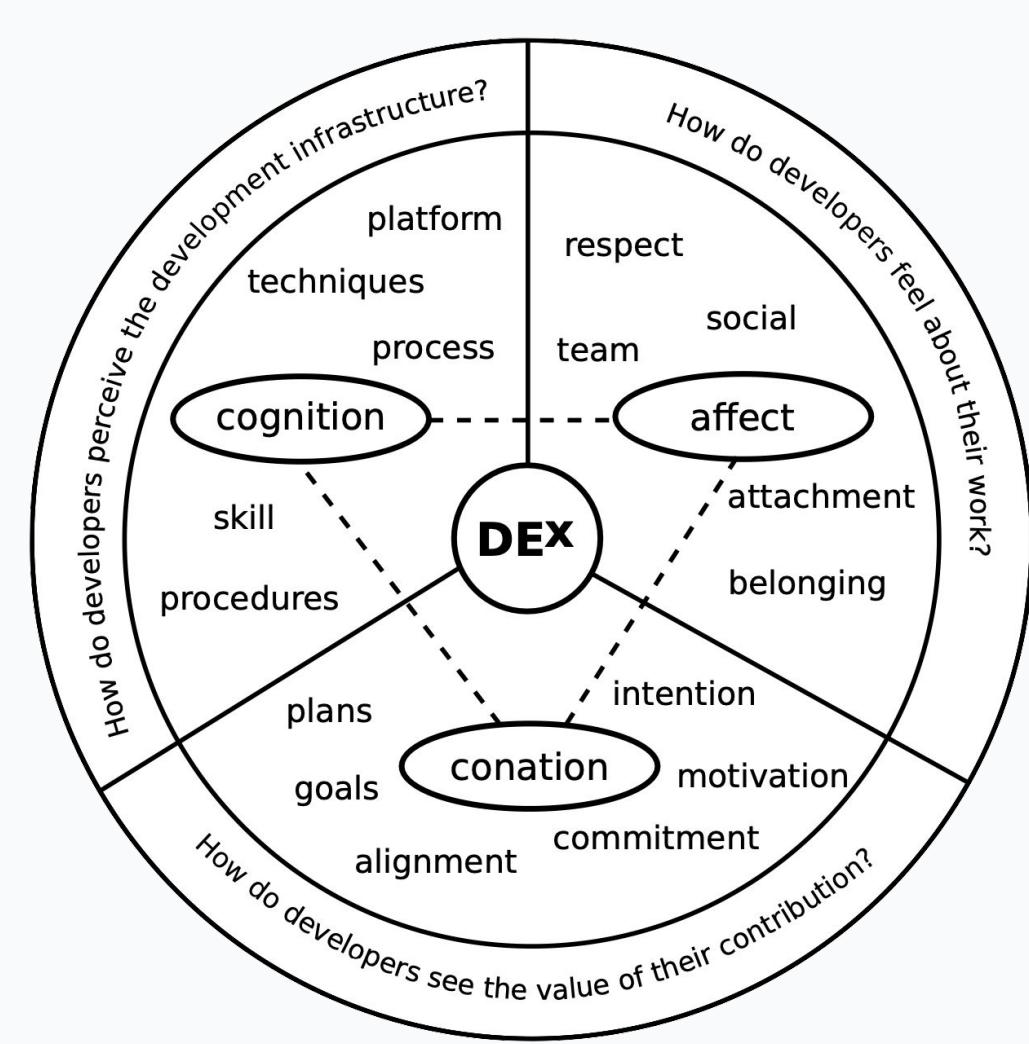
- [Evan Bottcher](#)

What is a
“Platform Team”?

The goal of a platform is to enable flow.

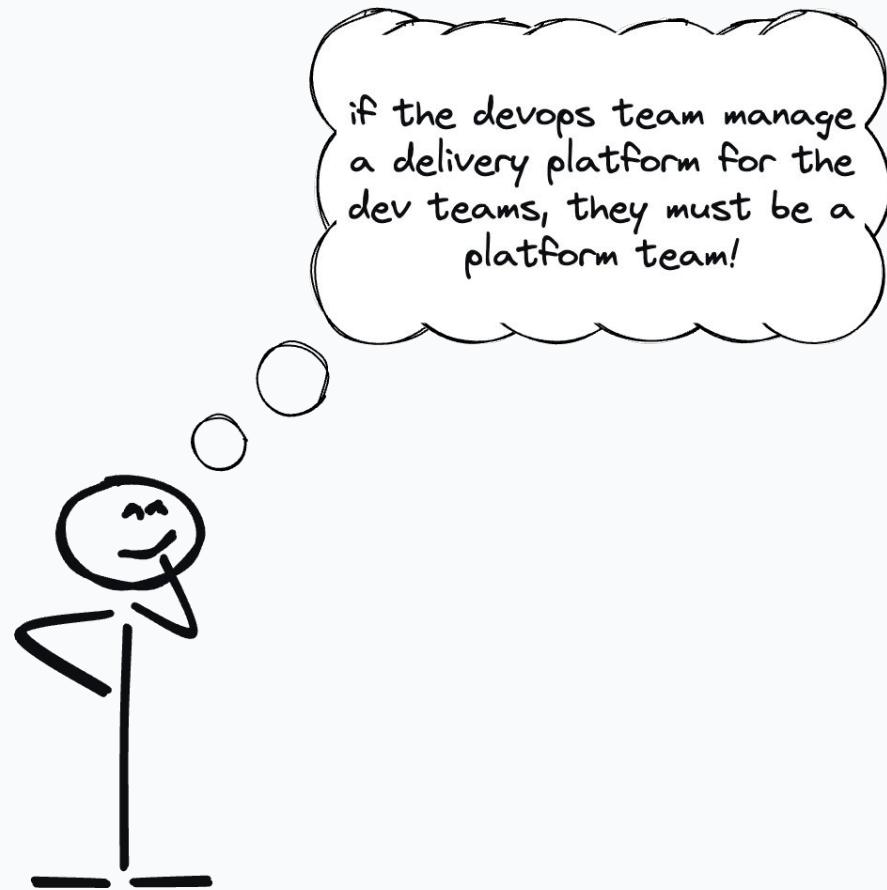


Why build a Platform?



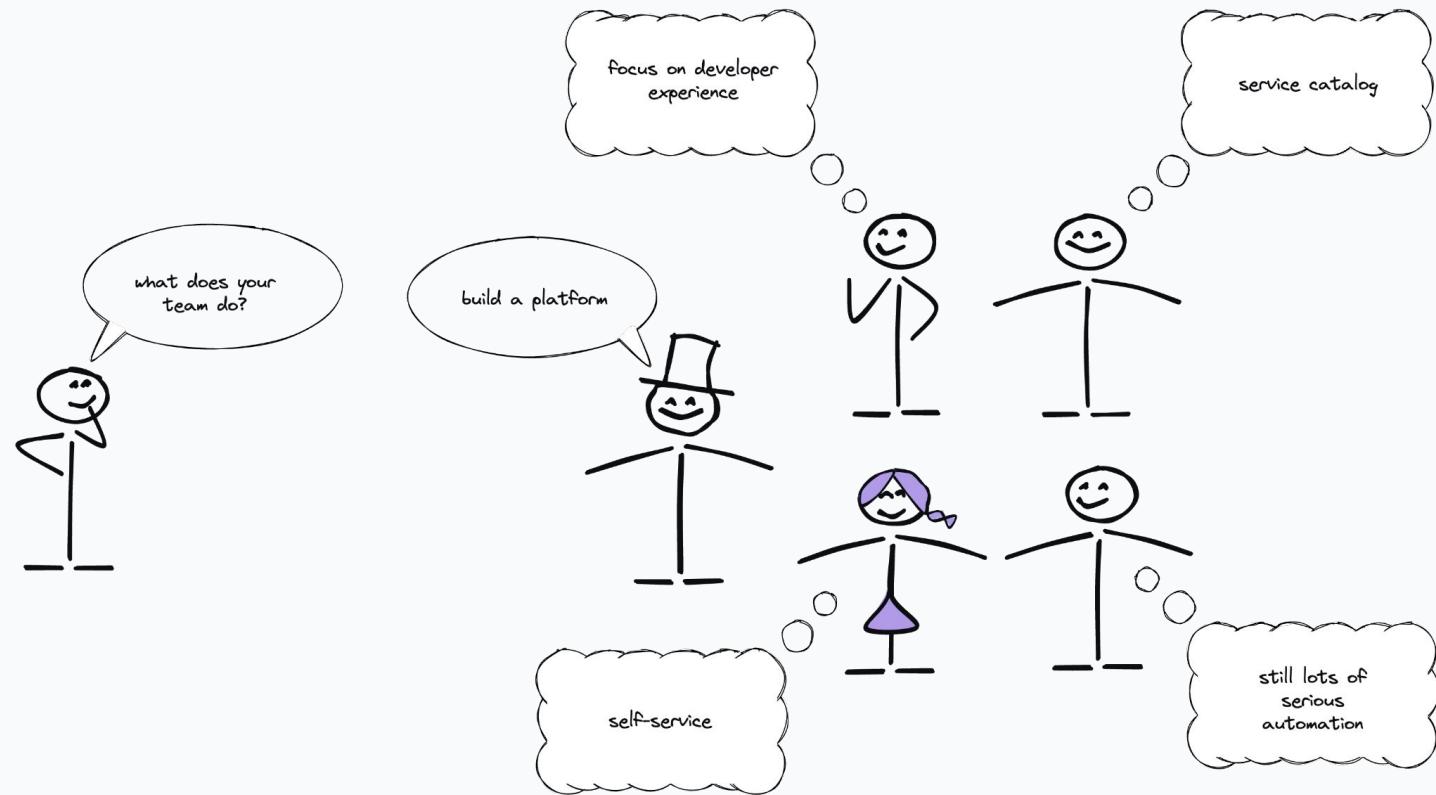
Developer Experience

Fagerholm, Fabian & Münch, Jürgen. (2012).
Developer Experience: Concept and Definition.



Back to our DevOps
team...

From DevOps to Platform team

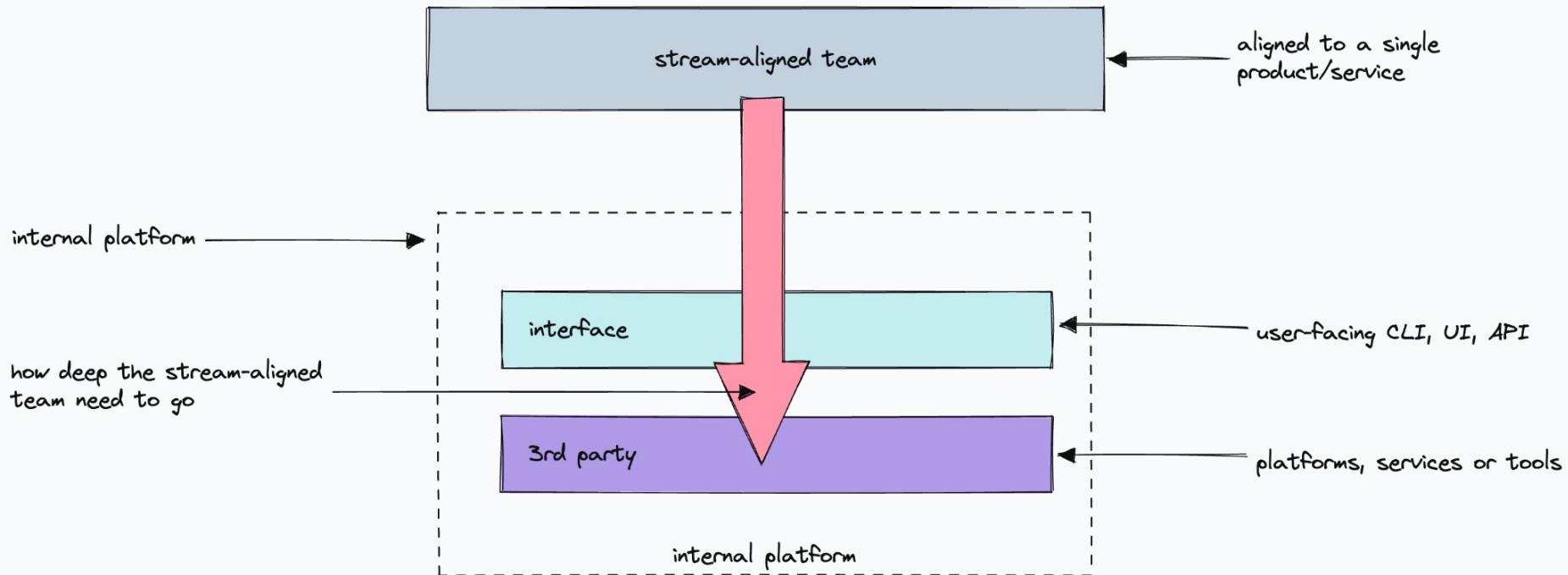


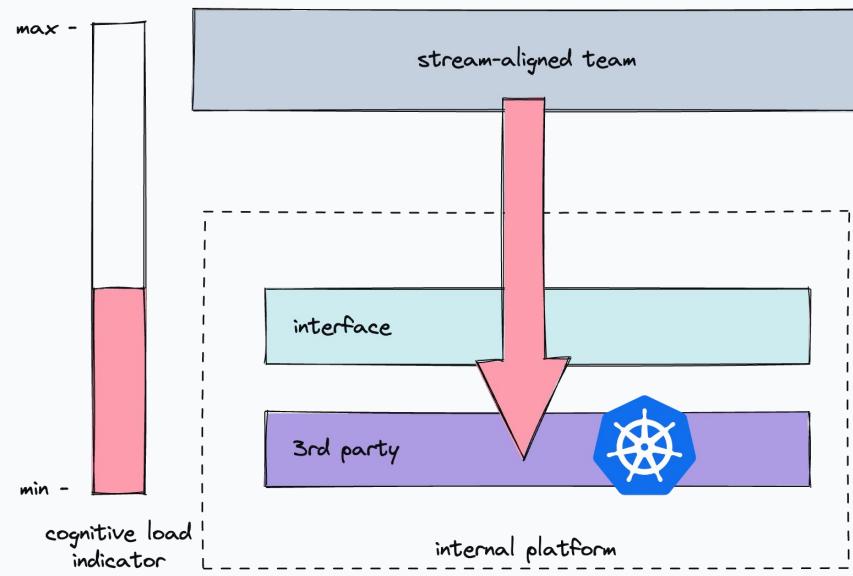
And thus,

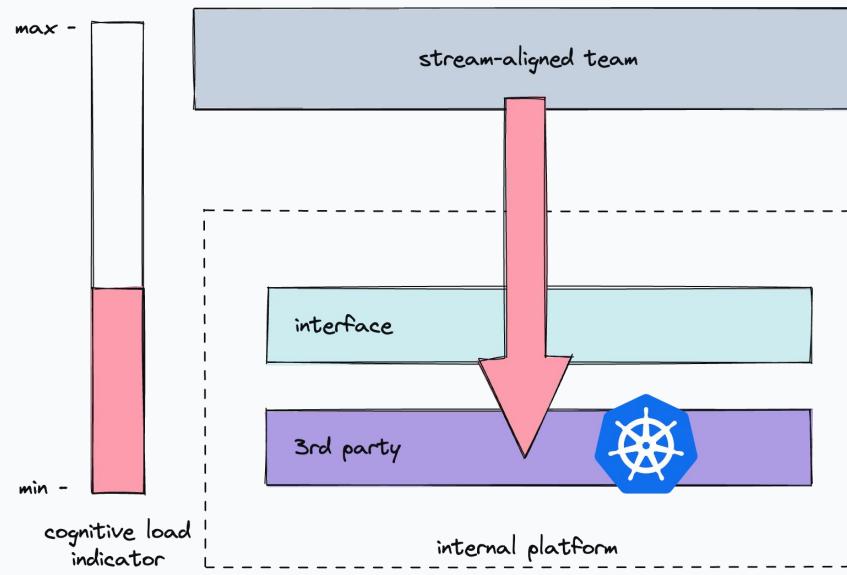
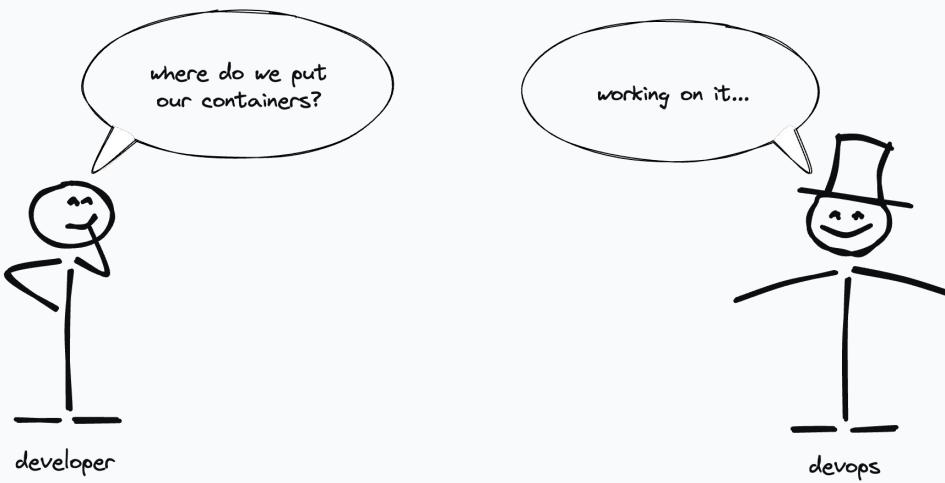


DevOps is Dead.
Long live Platform
Engineering

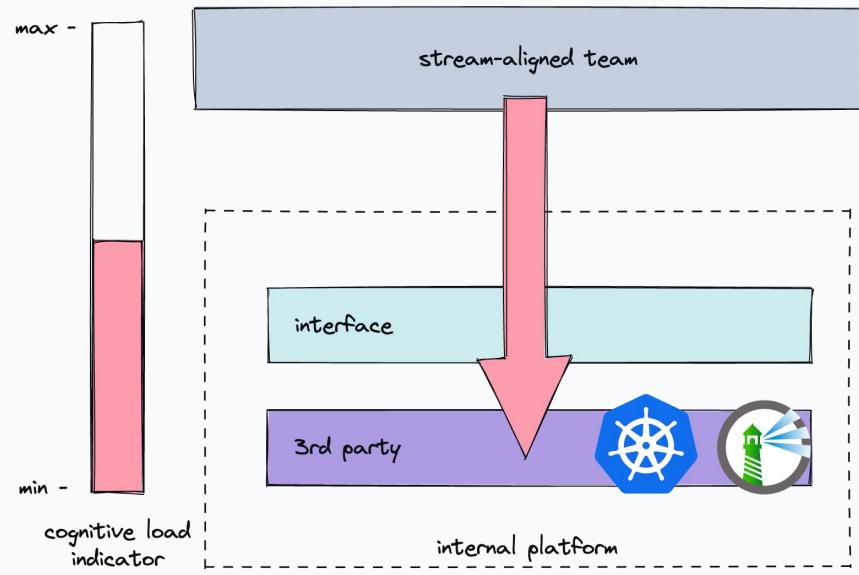
Let's take an
example...

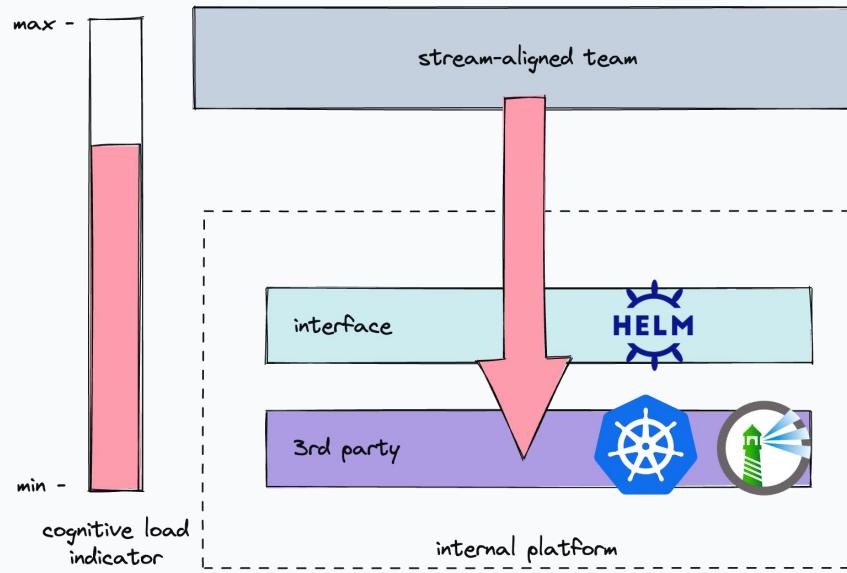
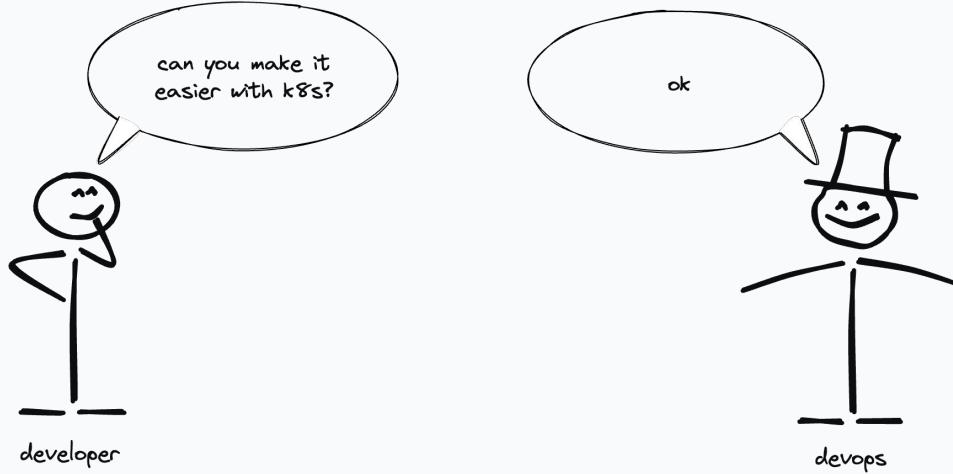


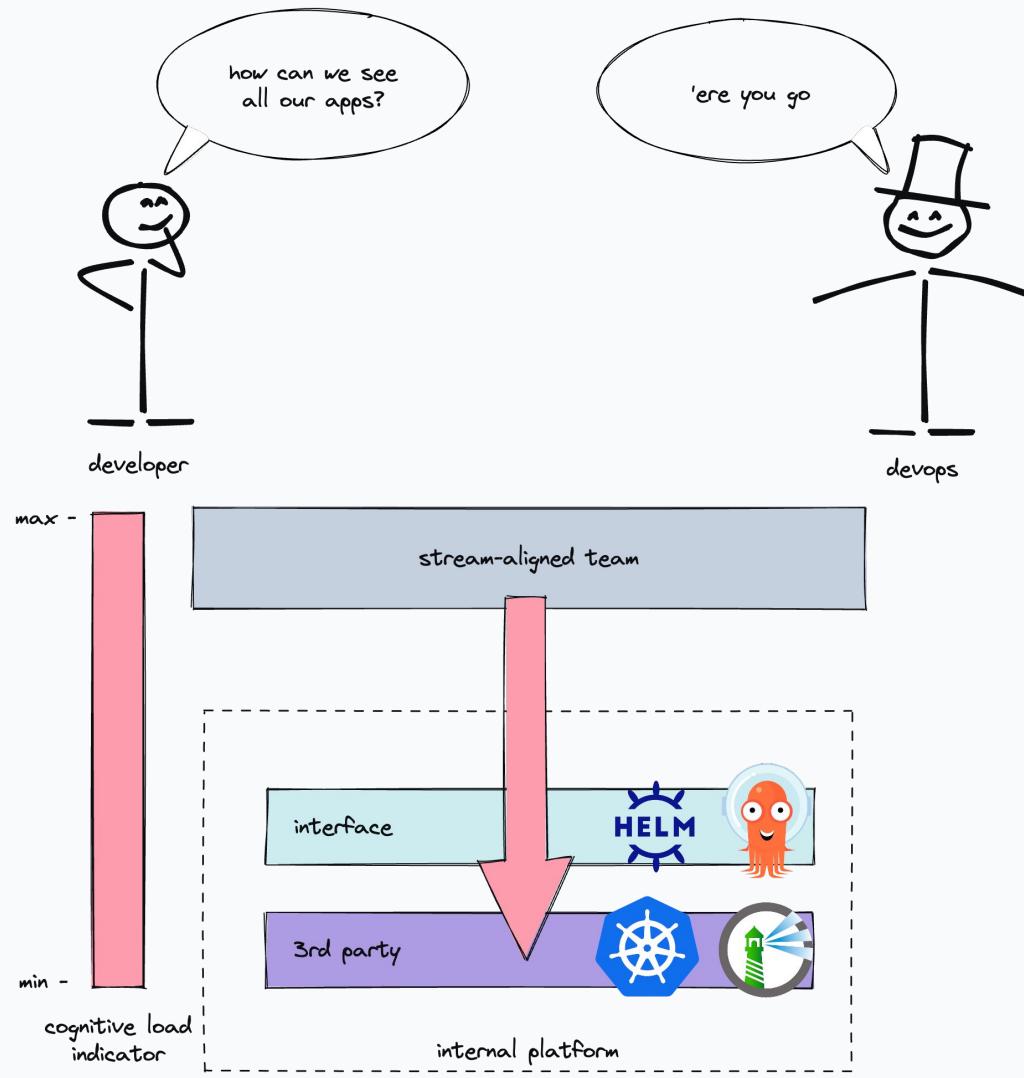


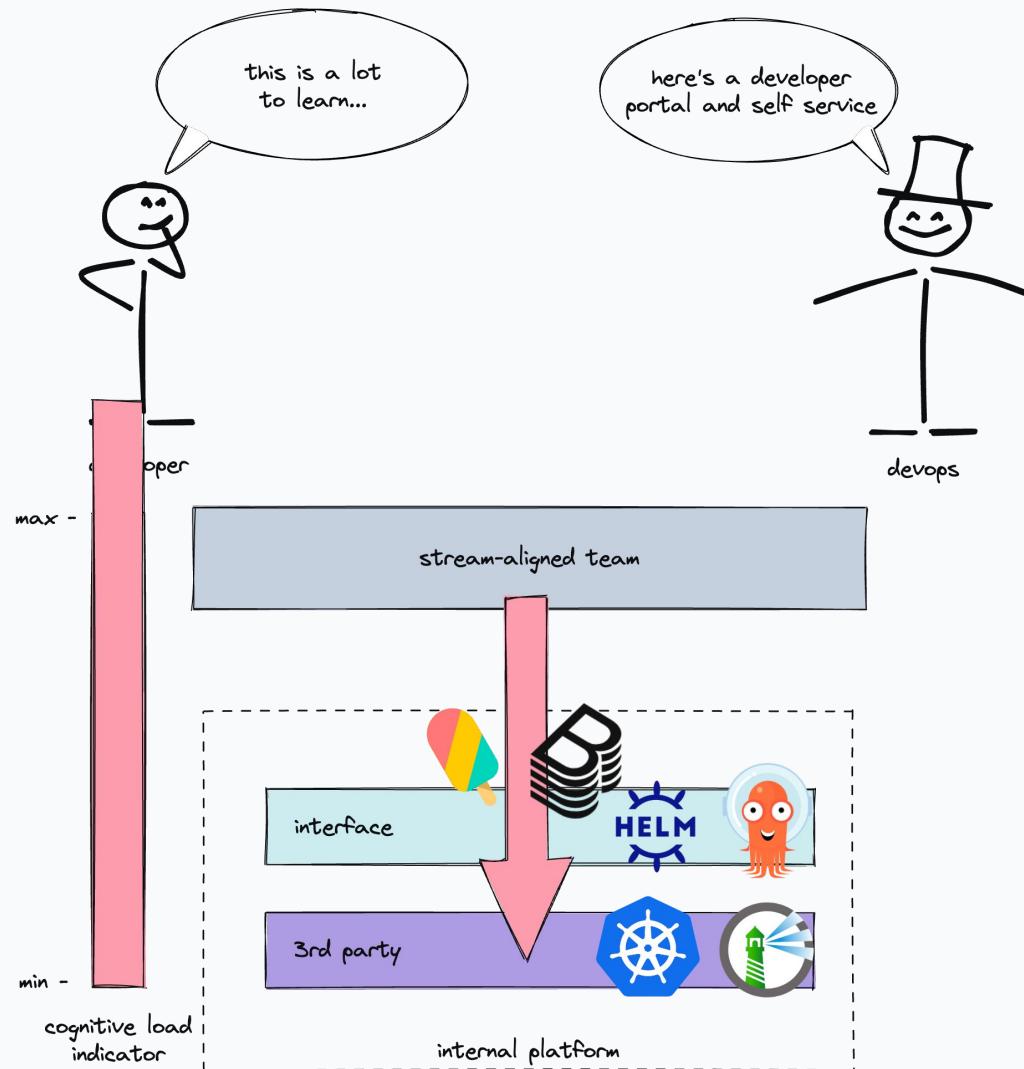


Meanwhile in the Dungeons of DevOps...







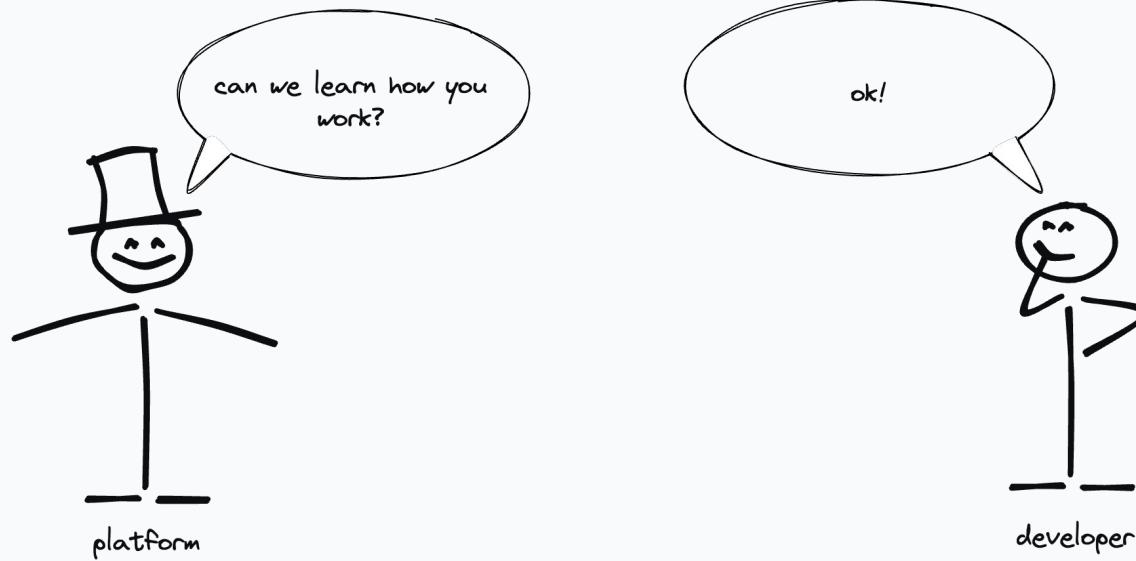


How was the platform built?

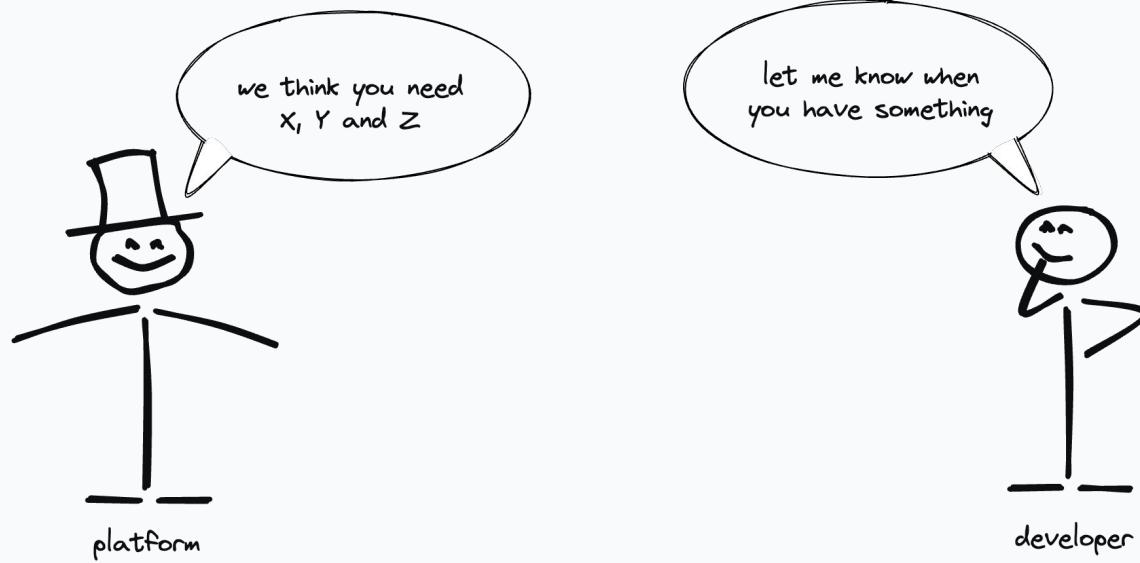
- “Design by accident”
- Reactively
- No real focus on Dev Experience
- Solving symptoms, not problems

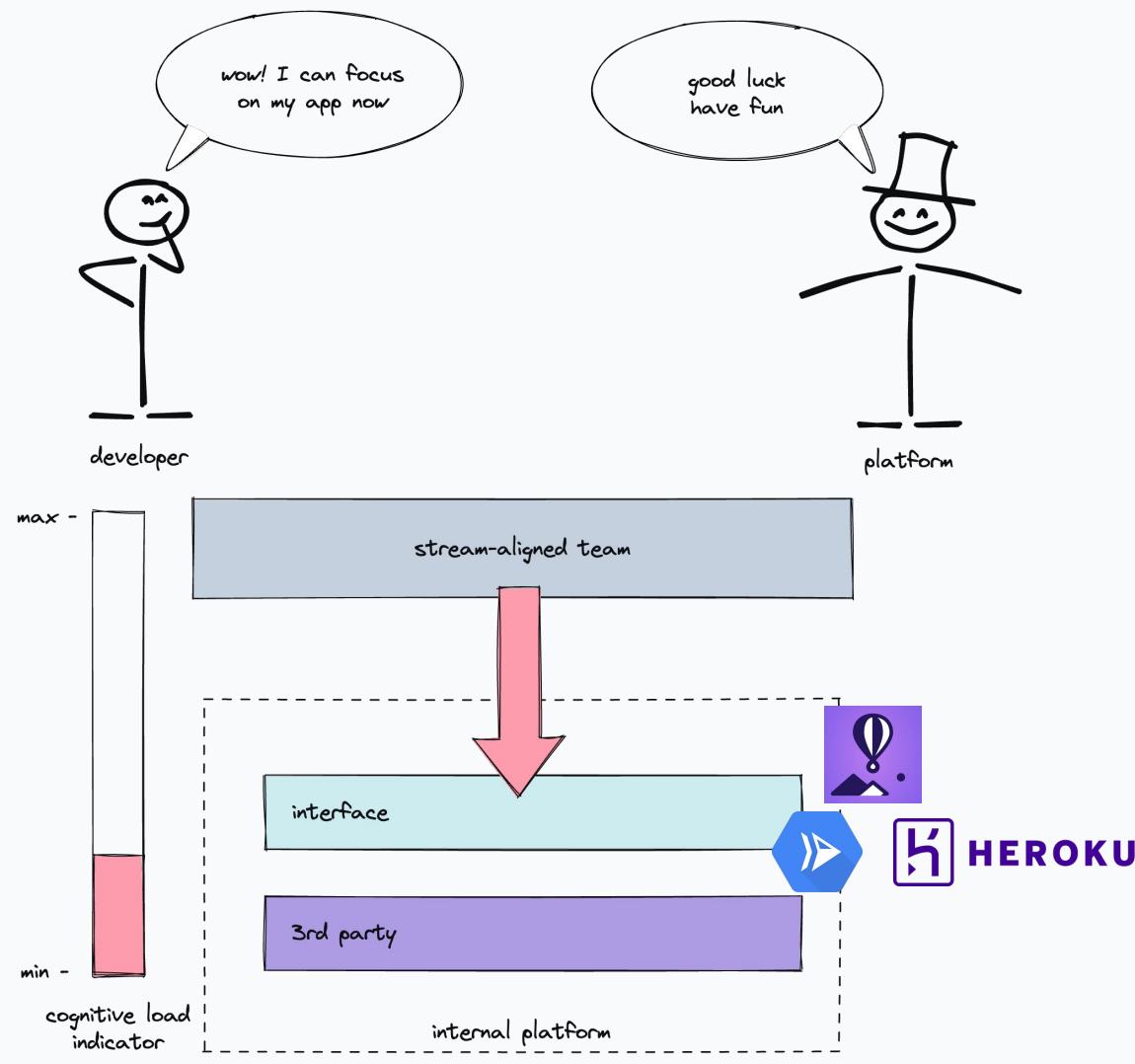
but we focused on
Dev Ex!





After some discovery, research and learning...





Characteristics of a Platform Team

- Well defined goal(s)
- Proactive
- Better at saying “no”
- Clear team boundaries

Practical tips

Users should choose
the platform

✗ Platform does not choose users

⁉ Why your platform over external?



“Platform as a Product”

 Goal is to reduce cognitive load

 Leaky abstractions

 Maintain backwards compatibility

Platform team must
“own” the interface

Generalised vs Opinionated



MVP, iterate fast



Difficult to remove APIs/features



Do as much as possible with as little
input as possible

~~Generalised~~ vs Opinionated



MVP, iterate fast



Difficult to remove APIs/features



Do as much as possible with as little input as possible



Manual is “better” for cognitive load



Scalability through numbers



!? Why force “GitOps” on our users?

Automation vs Manual

Layers of a Platform Interface

User-facing to
consume the API
and enhance UX.

Interface

Higher level
vocabulary
shielding users
from complexity.

API

Stable, evolvable
data-contract for
your abstraction.

Abstraction

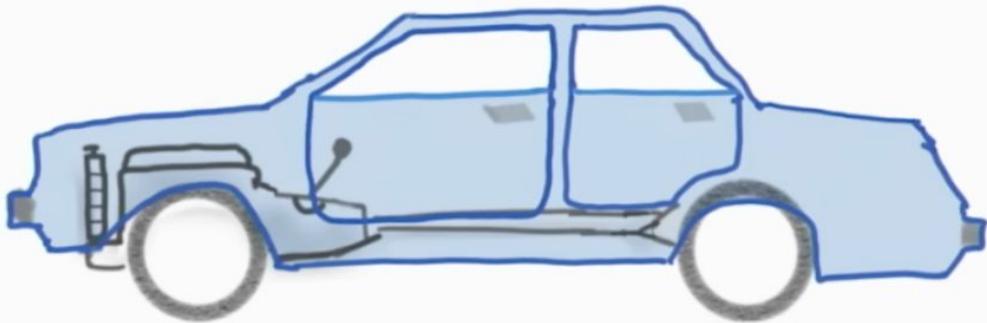
Abstraction:

Removing or generalizing details or attributes to focus attention on details of greater importance

Illusion:

Removing or generalizing important details, which cause the user to be misled

If software engineers had named the automobile...



...it'd be called `PistonCrankshaftGearWheelAssembly`

<https://twitter.com/ghohpe/status/1513362076990803969>

[Build abstractions not illusions - Gregor Hohpe](#)

Our investigation approaches

Value Stream Mapping

Learn what development teams actually do to deliver software. This helps with the abstraction.

Wardley Mapping

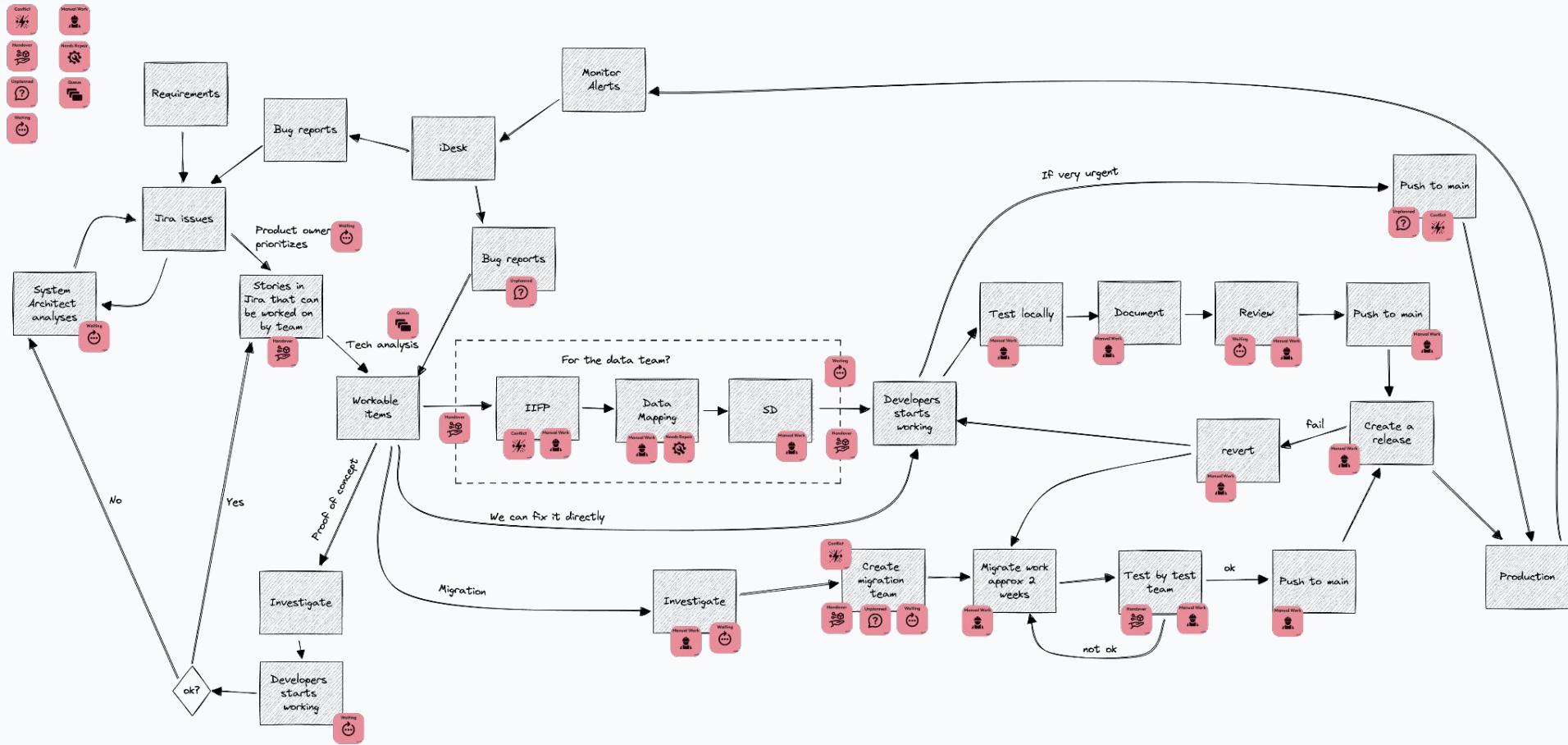
Visualise what you put in the face of developers and reason about it.

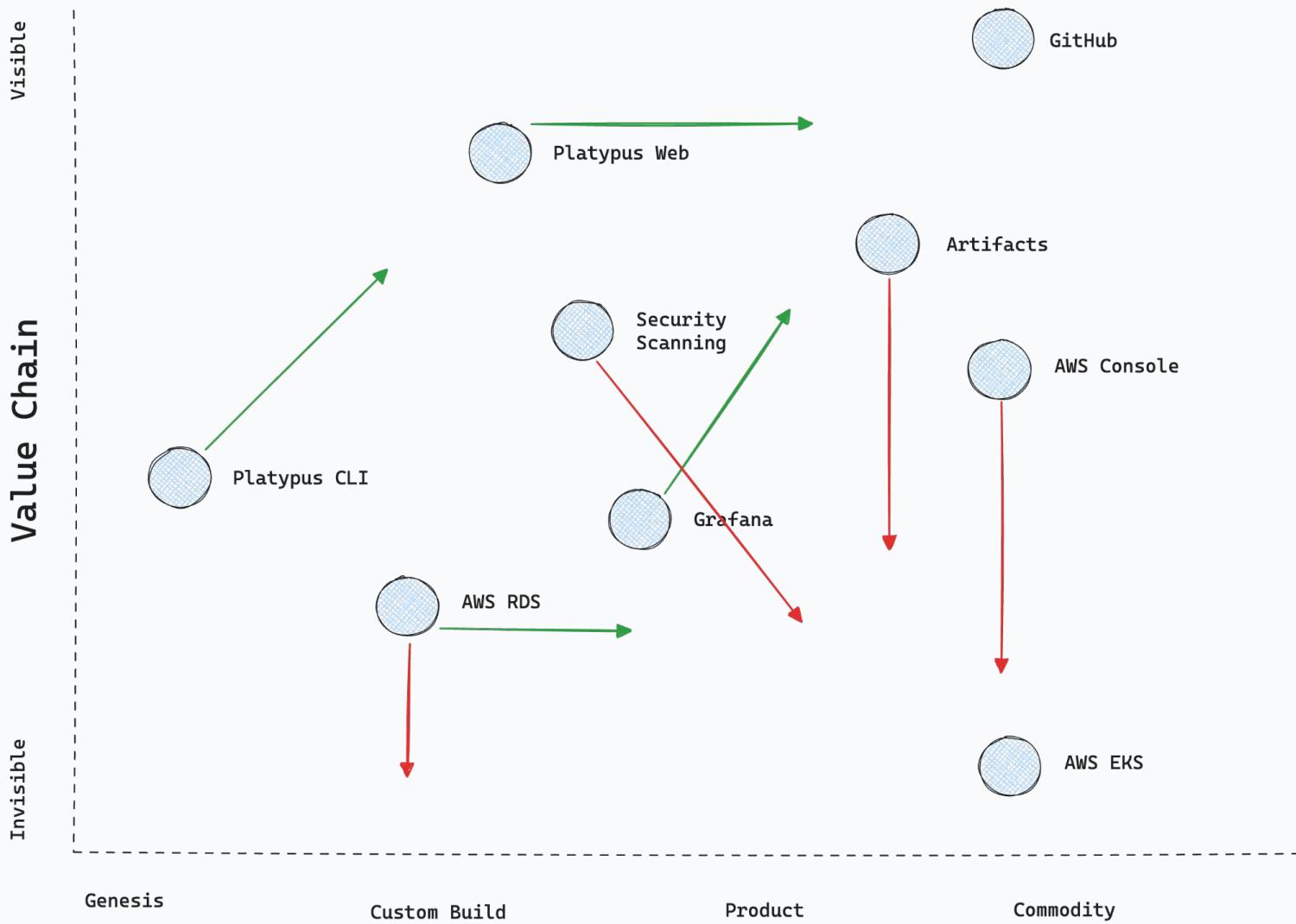
Gemba Walks

Observe development teams and learn about their every day work.

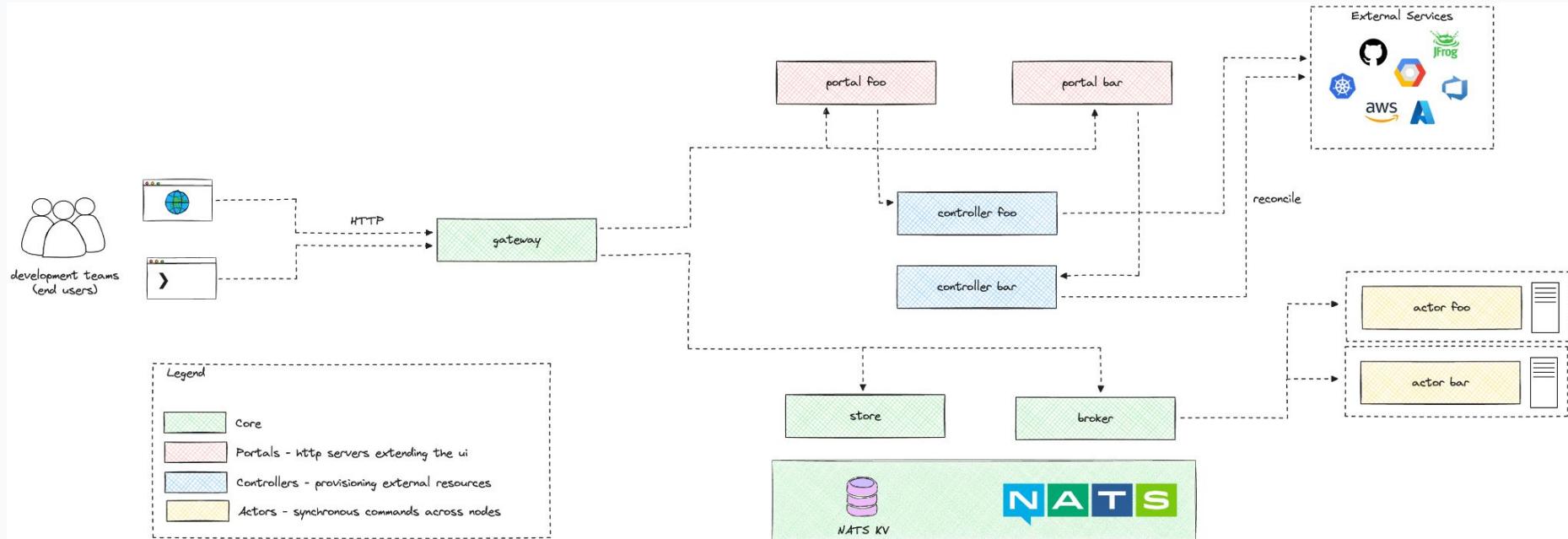
Recurring Surveys

Gain an overview of teams and those that are succeeding and/or struggling in areas.





BYOP with Horizon (a little side project)



When to use iterative development?

You should use iterative development only on projects that you want to succeed.

– Martin Fowler

Summary

- DevOps is not dead, but maybe DevOps Teams should be
- Platform Team should focus on users and enabling flow
- Developer Experience should be built iteratively, like a product

Questions?