Kris Buytaert @krisbuytaert

Lessons from 10 years of Platform Engineering

Devoppsdays Tokyo 2025

Lets start with some #definitions

What is #devops

A global movement to improve the quality of software delivery leveraging Open Source experience, started in Gent in 2009

What is #devops NOT ?

- developers doing operations
- operations people doing developers work

Who am I ?

Kris Buytaert:

- I used to be a developer
- Then I became an Ops person
- Organiser of #devopsdays, #cfgmgmtcamp, #loadays, …
- OpenSource consultant @ o11y.eu / inuits.eu
- Cofounder of all of the above
- Everything is a Freaking DNS Problem
- @krisbuytaert on mastodon.social /bsky/github/

What is (by Wikipedia)

What is a Platform ?

“A computing platform, digital platform, or software platform is the infrastructure on which software is executed. While the individual components of a computing platform may be obfuscated under layers of abstraction, the summation of the required components comprise the computing platform.”

What is Engineering ?

“Engineering is the practice of using natural science, mathematics, and the engineering design process to solve problems within technology, increase efficiency and productivity, and improve systems.”

What is Platform Engineering

Wikipedia says:

Platform engineering is a software engineering discipline focused on the development of self-service toolchains, services, and processes to create an internal developer platform

Team Topologies says :

Platform teams enable stream-aligned teams to deliver work with substantial autonomy. While the stream-aligned team maintains full ownership of building, running, and fixing an application in production, the platform team provides internal services that the stream-aligned team can use. ”

If Platform Operations was a response to devops failing,

You didn't understand devops

You were still doing dev, oops

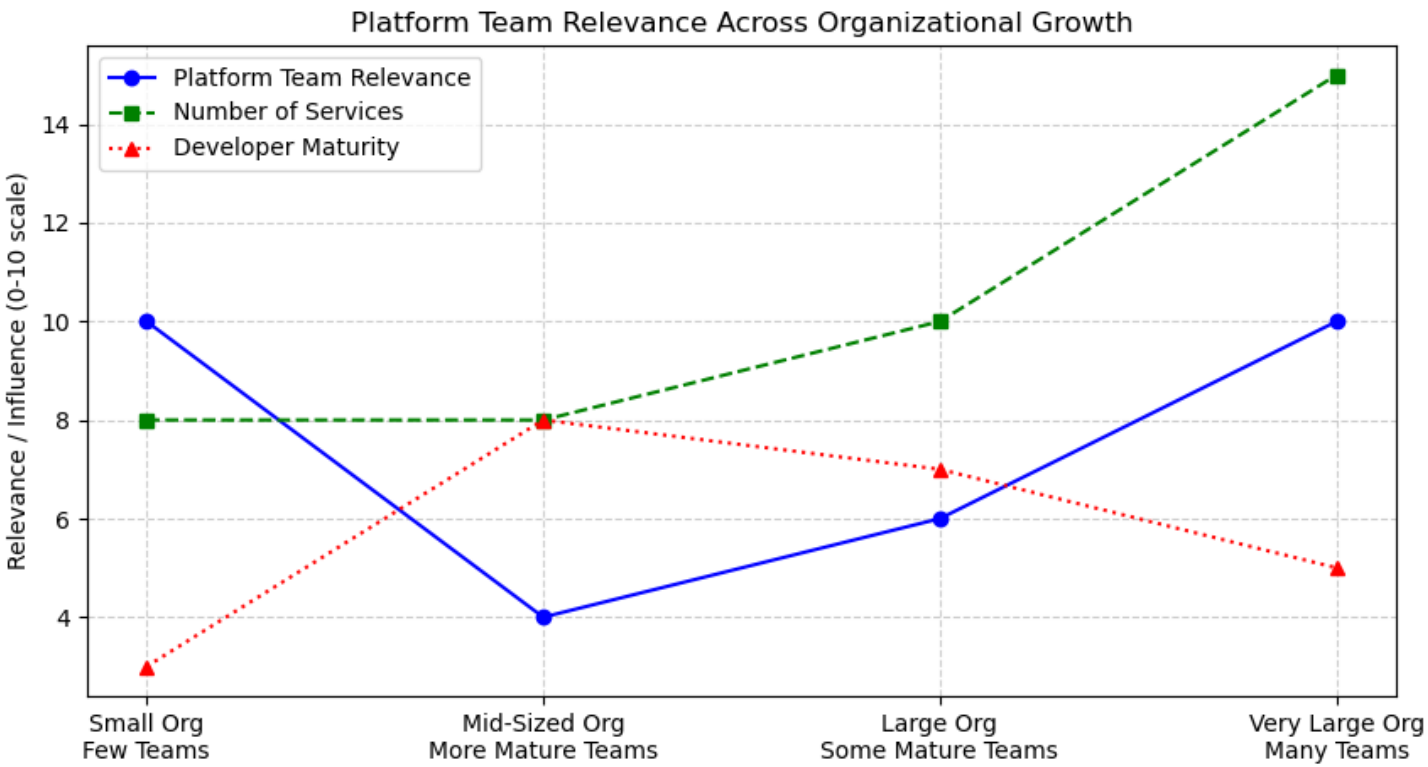
What is Plaform Engineering to me..

A specific implementation of collaboration between development and operations also know as #devops where one team offers a platform to be consumed by multiple other teams.

Job titles

devops engineer - platform engineer

Pendulum of adopting Platforms



Benefits of Platform engineering

- Self Service
  - Less manual interactions
  - Less room for unneeded flavours
  - Faster onboarding
- Enhanced standardisation
  - Standard Workflow
  - Standard Tooling
  - Standard Integrations

More Benefits of Platform engineering

- More focus for developers on development
- Standardised Security
- Basic Monitoring and Observability builtin
- Scalable On Call team
- Team members can change teams easier

Risks of Platform Engineering

- Overengineered platform
- Poor stakeholder engagement
- High Initial Building cost
- High Maintenance Cost
- Resistance to Change from existing development teams
- Lack of clear ownership

Our story : the setting

- small organisation ..
- multiple small development teams working on unrelated applications
- small operations team
- 1st platform

  2012-05-09 14:09 +0200 Anonymized Colleague I Inital commit
- self service

  2016-06-08 21:55 +0200 Kris Buytaert I Initial commit, folder only

The self service platform

- 1 (yaml) file to generate the pipelines
  - pipeline as code
  - build for for 2-3 language patterns
  - Enforced limited steps for standardizing, packaging, deployment
  - Allowed for free flow integration with tests, build etc
  - Pipeline as code for the Infrastructure as Code

The platform

- Fully Automated :
  - 1 script to generated the IaC code tree
  - 2 repositories to spin up the infra. (code/data)
  - Multiple promotion targets could be chained
  - 1 hour to spin up the fully clustered dev/uat/prod infra (depending on the engineer that did it)
- Semi Automated :
  - DNS
  - Public Access
  - Cloud / Bare Metal Selection

Onboarding

- General required introduction to the organisation
- Specific required introduction to how the platform works.
- Example self service task in onboarding
- First couple of years we failed to inboard people
- Then onboarding became required
- Still, Different new team members never got onboarded
- Platform onboarded was deemed as “unneeded”

The usage

- 20-25 projects onboarded
- on multiple public and private cloud platforms
- the only way to get a project live in the organisation
- 24/7 oncall team

Why we build it ..

- It was the only option ..
- Too many teams/projects
- Not enough people to put in each service team
- More flavours were too expensive
    - standardize
    - standardize
    - standardize

Can u afford custom flavours ?

Where we succeeded

- Monitoring built in
- Security built in
- Scalability built in
- Metrics built in
- Resilience build in .
- Cloud agnostic
- Basic Guardrails in place

We did survive the 10th floor test. (aka the OVH fire)

Self Service

vs Self service adoption ..

- Most devs / PM's didn't care .
- Still asked the Platform team to spin up a new stack
- Only a limited number of devs actually used the self service.
- Standardisation agreements often were ignored,

Where we failed

- Teams didn’t add custom tests in the pipelines..
- Teams didn’t build custom Metrics
- Teams didn’t consume logs

Why we failed

- The teams wanted their own flavours
- They didn’t understand the cost of running their own flavours
- We didn’t evangelise why we build this …
- We didn’t prove the actual cost
- We didn’t have Product Management on the platform ..
- Teams wanted their own tools , not the tool we “forced” on to them.
- We were probably to early

Was the feedback loop missing ?

- For some teams probably ..
- Others were actively using the ecosystem.
- Others asked for antipattern features
- e.g
    - Support for Multiple Branches
    - Support to promote broken builds
- The guardrails still felt blocking

Find your champion

- Find a key consumer of the platform
- Let them contribute
- Let them Evangelise

Finding your champion can be hard.

- people aren’t allowed
- people have too much work / pressure
- people don’t have the skills
- different priorities

Developer Experience / Usability was missing

- Tooling did not exist yet.
- No internal developers to support our effort
- Yaml is NOT useable
- Yaml is NOT useable

The result :

- Shadow IT
- Bigger failure
- New inexperienced engineers created new (container) platform.
- Now it’s 5 different templates to be modified that generates a broken pipeline.
- After 5 years still not includes relevant monitoring
- It was Insecure by design
- The Costs have more than doubled.

If you build it ..

They won’t come .. They will object ..

They will create shadow IT They will create security problems

Lessons

- Automate , Automate
- Self Service is key
- Evangelise
- Find your champions
- Treat the platform as a product

Kris Buytaert

@krisbuytaert

kris@o11y.eu

Essensteenweg 31

2930 Brasschaat

Belgium

Contact: info@o11y.eu