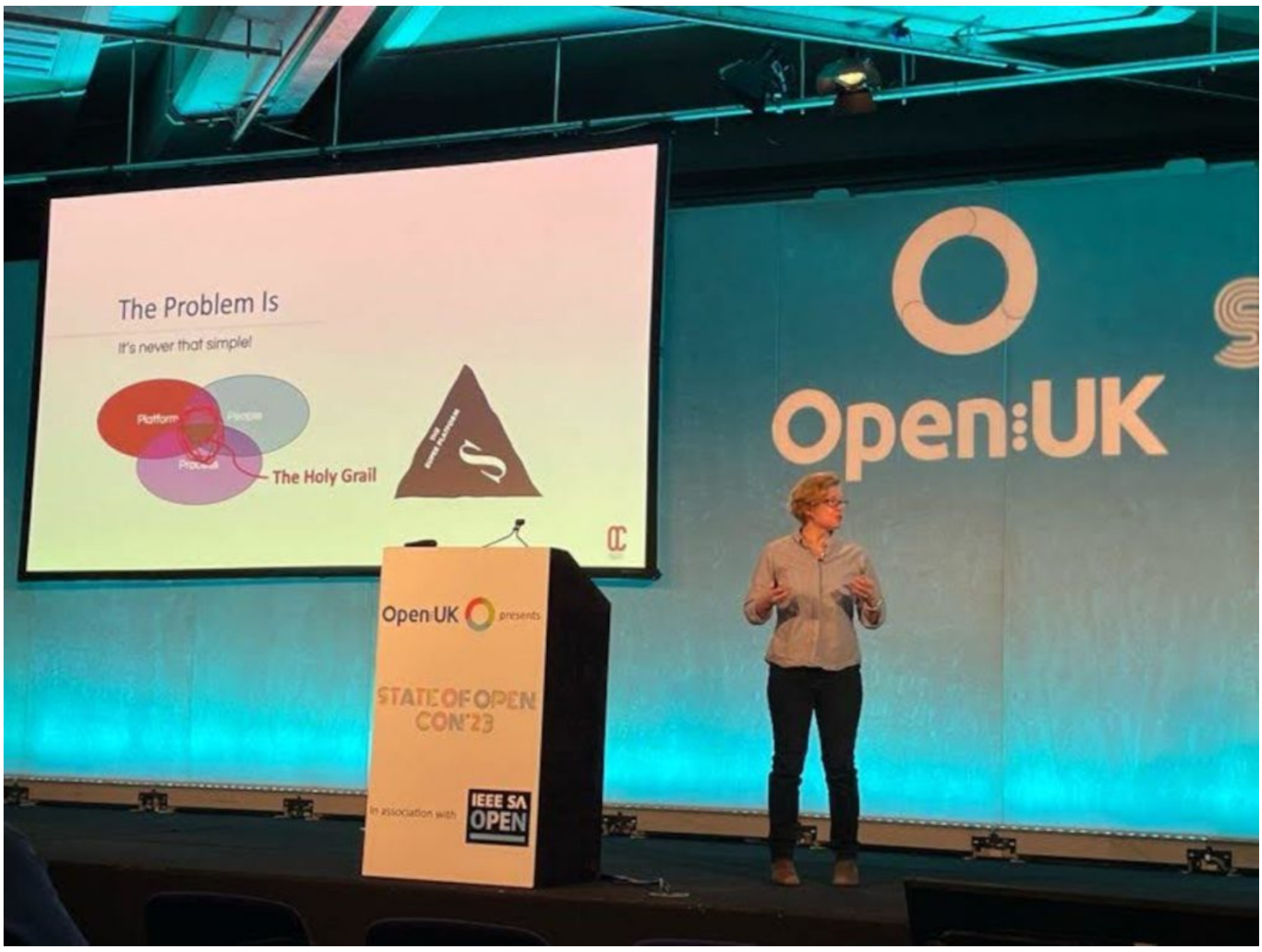
DEVOPS / PLATFORM ENGINEERING / TECH LIFE

How to Foster a Good Internal Developer Platform Experience

Feb 13th, 2023 3:00am by Jennifer Riggins



LONDON — "App developers want to move fast, and operations engineers want to move safely," which is where the DevOps model arose, Abigail Bangser, Syntasso's principal engineer, said at this week's State of Open Con. DevOps brought all the skills needed to release into one team, which lessened some of the cognitive load on developers who no longer had to worry — too much — about the security, support and cost on top of their code because the ops engineers on their team had that covered. DevOps also

concerned with infrastructure, scaling, and security. These, she emphasized, are non-differentiating but not unimportant. Which is where the internal developer platform team comes in to reduce that cloud-heavy cognitive load so devs can focus on development and ops can focus on operations. While the platform engineers focus on that non-differential but important code that, cross-organizationally, teams need.

"So you have a team that produces and operationalizes tools to help other teams produce and operationalize their software applications to their end users," Bangser said, offering up the latest definition of platform engineering.

So now that we know what a platform engineering team is supposed to do, let's dive into how to foster a good internal developer platform experience — so your customer colleagues will actually want to use it.

Who Are You Building for, Anyway?

The app teams and maybe the app teams are about the shiny-shiny, but the platform team is about Old Reliable. "I've found people get fixated on the new innovative technologies but they lose track of the people they are building it for." Nicki Watt, as the CEO and CTO of OpenCredo distributed systems consultancy, has helped several organizations build internal developer platforms, allowing her to uncover some patterns and anti-patterns, which she shared with the State of Open Con crowd.

A lot of clients, she says, think Kubernetes is the platform, but "If you want to build a great platform, it's got to be more than just Kubernetes and the open source ecosystem around it." An internal developer platform is a mix of people, process and technology — and it's not a straightforward recipe — or a Venn diagram. It takes empathy for your internal developer community.

First and foremost, Watt reminded attendees that "Your audience is more diverse than you realize." Especially as an organization grows, platform engineers — mostly infrastructure specialists themselves — tend to not be aware of how many different levels of skills and experience are building the stack above them. This results in low-level YAML abstractions, she warned, and "There's a lot more communities that need to be taken into account than those."

Primarily that's the app developers, but also don't forget about data scientists and machine learning engineers that may need hardware or other different features. She also observed that the community of leadership and governance — including regulatory and finance — need to be thought of in the platform design. After all, they will want reports to see if and how they are gaining a return on this investment. Part of the platform engineering journey, she says, is educating execs on the value of it.

And then, "Adapting the platform itself for specific community requirements is good but not sufficient,"

"Some need more freedom to innovate and others will need a bit more guidance," she said. "If you want to build a really great platform engineering developer experience, this requires you to look at it as a holistic socio-technical challenge." Her definition of platform engineering comes down to building, maintaining and providing a "curated platform experience for all the communities using it," which factors in all the evolving technical, social and team structures.

A Good Platform Establishes Boundaries.

We've already spoken about how an internal developer platform sits at different levels at different organizations — and even at different levels within an organization. Watt says a good platform engineering experience establishes clear boundaries and responsibilities, answering both: How much does the platform do? versus How much does the engineer need to do? Otherwise you end up in a Blame Game, she explained.

These boundaries can be different per stakeholder but then remember to document those differences, including:

- What does the platform do?
- What is the platform team responsible for?
- What are the app teams responsible for?

Of course, as with any developer experience, documentation is an important part of encouraging self-service and automation, which, Watt says, is most sought after by users of the platform. "This self-service gives teams the ability to go as fast or as slow as they need to go." She emphasized that the aim is to make teams independent of and not dependent on you — "We're aiming for empowerment here" not platform engineers becoming overwhelmed with internal support tickets.

Platform teams are the builder of the golden pathway, which she explained, is flexible and evolvable, but, mainly, makes sure there's order. Promote freedom within boundaries over anything goes, she recommends, establishing ground rules upfront. This can be single cloud versus a multicloud setup, or only offering support for the standard stack, while the platform team doesn't offer support for the latest shiny experimentation outside the platform rails.

"By doing that you give people freedom to choose but you don't give a free-for-all which can lead to challenges later," Watt said. Choices are limited within ecosystems.

An API Becomes a Contract with Your Internal Developers

An API continuously comes up as the easiest way for developers to consume the internal platform in the way they want to. In part, that's because, as Postman's Chief Evangelist Kin Lane puts it, an API is a contract and can even act as your Service Level Agreement with your API customers. "An API contract is a shared understanding of what the capabilities of a digital interface are, allowing for applications to be programmed on top of," he first wrote back in 2019. An API mitigates and communicates change in both a human- and machine-readable way, and should provides some guarantees of reliability and stability, he continued.

Building your internal platform the be API-first allows it to be more self-serviceable. By extending that service contract to an internal platform, Bangser said, an API becomes:

- A discoverable catalog of offerings
- One or more user interface options
- On-demand offerings (although not necessarily instant delivery)
- An introduction to a "do nothing" script

This is why she recommends kicking off your platform engineering journey with the limited time and

API-first development.

“APIs allow for clean communication boundaries,” she said, and “creating an API provides the ability to iterate.”

Watt goes further to suggest defining an actual internal platform contract, a la Amazon Web Services‘ Shared Responsibility Model. Clarify those boundaries, like if an organization wants the platform team to handle all security checks or if your platform engineers just provide the tooling and each team is responsible to run the scan on its own containers. Whenever possible, she recommends, favor automation and API-driven interactions, especially for onboarding and infrastructure provisioning.

And, of course, documentation is essential, including creating templates and reference implementations, which further supports self-service.

“You want to pull your teams closer to the platform, to engage with the platform. A good way to do that is to provide documentation and the reference implementation they need to get going,” Watt said.

Don’t forget to provide the professional services side of the platform engineering experience, as well. Watt says this can include workshops, pair programming, embedding in DevOps teams, or, as we learned from CyberArk, you can borrow app developers for your platform team. As your platform team scales, you may need your own platform evangelists to engage your colleague customers.

Don’t forget, if your internal platform is a contract with your app teams, then reliability matters, and continuous testing must be built into your platform strategy from the start. A sure-fire way to get resiliency and reliability built in, Watt says, is to eat your own dogfood and use the platform — “Impose on yourself the restrictions you have on other people,” especially the “tooling to spin up infrastructure. As soon as you do this, you will find problems get fixed really quickly,” she said. If that’s not possible, at least test against your reference implementations as exemplar tenants.

In the end, platform engineering demands that you “adjust your thinking — aim not just to provide a service but rather to be a service and serve your communities,” Watt said. “We need to be open not just as in open source but open in the way we interact with people and open with the teams about what we need to be successful.”

Disclosure: The author of this article does some freelance content work for OpenUK, the presenter of State of Open Con.

TNS

Jennifer Riggins is a culture side of tech storyteller, journalist, writer, and event and podcast host, helping to share the stories where culture and technology collide and to translate the impact of the tech we are building. She has been...

Read more from Jennifer Riggins →

RELATED STORIES

Drive Platform Engineering Success with Humanitec and Port

FOLLOW TNS

TNS DAILY

ARCHITECTURE

Cloud Native Ecosystem
Containers
Edge Computing
Microservices
Networking
Serverless
Storage

ENGINEERING

AI
Frontend Development
Software Development
TypeScript
WebAssembly
Cloud Services
Data
Security

OPERATIONS

Platform Engineering
Operations
CI/CD
Tech Life
DevOps
Kubernetes
Observability
Service Mesh

CHANNELS

Podcasts
Ebooks
Events
Newsletter
TNS RSS Feeds

THE NEW STACK

About / Contact
Sponsors
Sponsorship
Contributions

FOLLOW TNS

roadmap.sh

Community created roadmaps, articles, resources and journeys for developers to help you choose your path and grow in your career.

Frontend Developer Roadmap
Backend Developer Roadmap
Devops Roadmap

FOLLOW TNS

TNS DAILY